УДК 004.043

Разработка моделей представления данных сложных производственных проектов в современных информационных системах

Черновалова Маргарита Витальевна
Филиал «Национальный исследовательский университет «МЭИ»
в г. Смоленске
Магистрант

Булыгина Ольга Валентиновна
Филиал «Национальный исследовательский университет «МЭИ»
в г. Смоленске
К.э.н., доцент кафедры менеджмента и информационных технологий

Аннотация
Рассмотрена архитектура системы поддержки принятия решений по управлению производственными проектами, предусматривающая использование нескольких моделей данных и их взаимное преобразование. Проведен анализ способов прямого и обратного отображения объектных и реляционных представлений данных. Приведен пример практической реализации одного из вариантов преобразований.
Ключевые слова: системы поддержки принятия решения, модель данных, производственный проект, объектно-ориентированное программирование.

Development of data representation models of complex productive projects in modern information systems

Chernovalova Margarita Vitalievna,
The Branch of National Research University "MPEI" in Smolensk
Master student

Bulygina Olga Valentinovna
The Branch of National Research University "MPEI" in Smolensk
PhD, associate professor of department of management and information technology

Abstract
The article describes an architecture of the project management decision-making support system, that includes multiple convertible data models. The analysis of the ways of forward and reverse mapping object and relational views of data is given. An example of the practical implementation conversion is considered.
Keywords: decision-making support system, data model, industrial project, object-oriented programming.

В настоящее время системы поддержки принятия решений получили широкое распространение. Это связано с тем, что они позволяют принимать решения в условиях неопределенности при наличии неструктурированной или частично структурированной информации, позволяя при этом достичь результатов, не уступающих тем, которые могли бы быть получены при работе с экспертами. Само возникновение таких систем связано с применением и развитием методов искусственного интеллекта, а также совокупности научных дисциплин, изучающих методы решения задач творческого характера с использованием вычислительных систем. Таким образом, системы поддержки принятия представляет собой системы, которые оперируют знаниями специалистов в заданной предметной области и формируют в зависимости от ситуации некоторые конкретные решения.

В связи с тем, что производственные проекты отличаются высокой сложностью и длительностью, традиционное представление данных информации о них с помощью реляционной модели данных становится неэффективным вследствие того, что необходимо хранить большие объемы трудно формализуемой информации. [1, 2]. Так в процессе управления сложными производственными проектами необходимо иметь информацию о целях, событиях, мероприятиях, работах и их результатах, сценариях, изменениях, ресурсах и т.д. Задача обеспечения полноты, актуальности и непротиворечивости данной информации осложняется наличием нескольких представлений: реляционного – для хранения в базе данных, объектного – для обработки. В связи с этим возникает потребность в инструментах, обеспечивающих преобразование данных из одного представления в другое с учетом варианта реализации каждого из них [3].

Рассматриваемый тип проектов предполагает наличие больших объемов плановой и оперативной информации, которая должна храниться в базе данных автоматизированной информационной системы. В связи с этим в процессе проектировании структуры такой базы данных необходимо принимать во внимание тип СУБД, используемой на предприятии.

В настоящее время наиболее широкими возможностями по хранению и обработке информации обладают объектные СУБД, однако, наиболее распространенными на отечественных предприятиях до сих пор остаются реляционные [4] .

Процесс разработки структуры реляционной базы данных начинается с построение логической и физической моделей. В данном случае первая модель предназначена для отражения понятий предметной области и взаимосвязей между ними (рисунок 1).

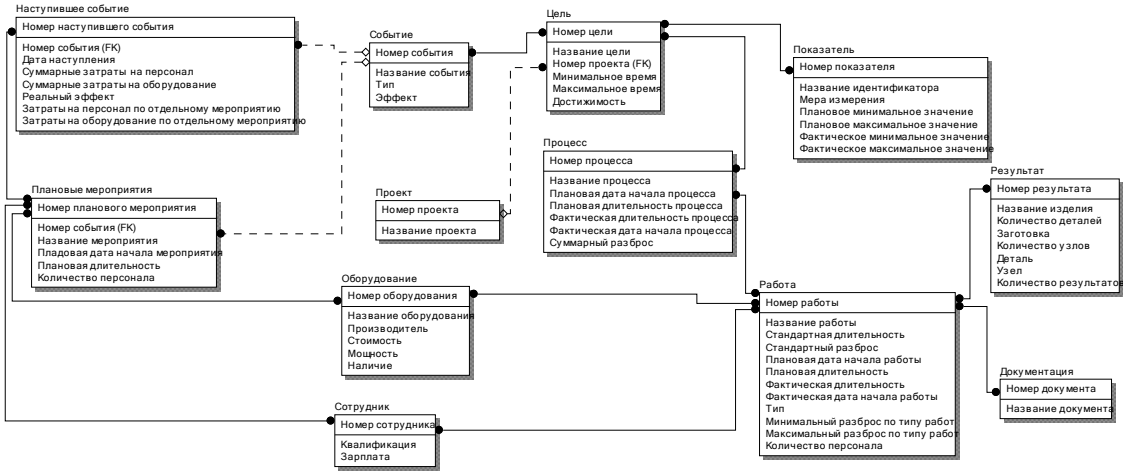

Рисунок 1 – Логическая модель производственного проекта

В результате анализа предметной области было установлено, что между многими выделенными сущностями имеется связь «многие-ко-многим», что также следует из рисунка 1. Это является естественным для производственных проектов. В связи с тем, что отражение таких связей в реляционной модели невозможно, физическая модель БД в значительной степени отличается от логической (рисунок 2). В данной модели для отражения связей «многие-ко-многим» были сформированы дополнительные таблицы, например, REAL_EVENT_MER, PROC_WORK, WORK_PERSON, и т.д. Появление справочных таблиц (например, EVENT_TYPE) в свою очередь, направлено на повышение эффективности работы с БД.

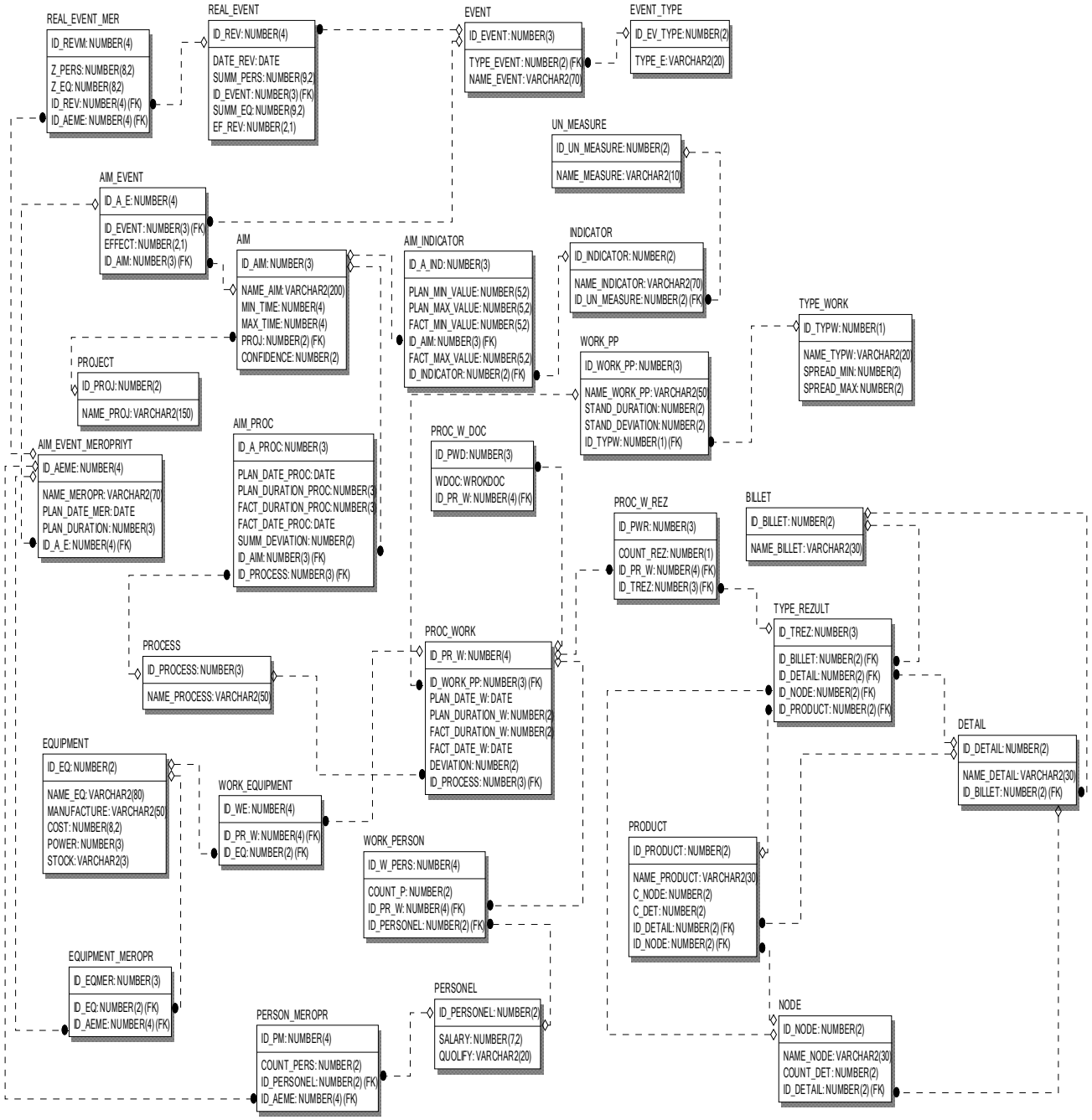

Рисунок 2 – Физическая модель производственного проекта

Как видно из физической модели на рисунке 3, представление проекта в сфере производства наукоемкой продукции в реляционной БД довольно затруднительно. Это обусловлено, прежде всего, наличием большого количества таблиц и записей, которые необходимо хранить, многие из которых служат лишь для хранения первичных ключей и предназначены для физической реализации связей типа «многие-ко-многим».

Объектная модель данных производственного процесса представлена на рисунке 3.

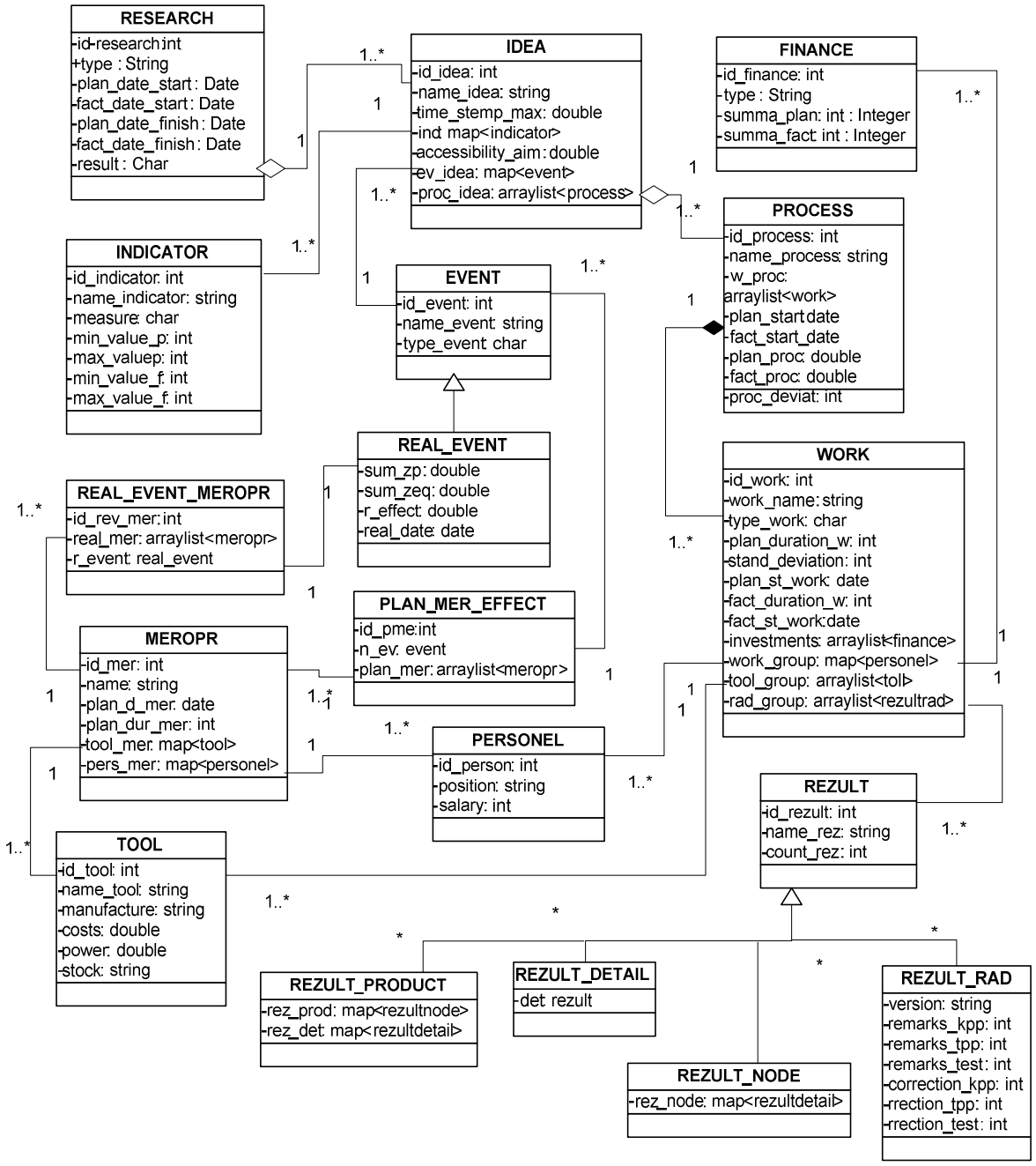

Рисунок 3 - Диаграмма классов процесса подготовки производства машиностроительной продукции

Из рисунка 3 видно, что количество классов в данном случае меньше, чем количество таблиц в БД, что в значительной мере сокращает объем хранимой и обрабатываемой информации, повышая тем самым скорость проводимых манипуляций с данными. Кроме этого к достоинству данного подхода к представлению информации можно отнести ее наглядность и понятность для человека. Так каждый класс имеет свою смысловую нагрузку и отражает сущности реальных объектов, принимающих участие в данном процессе.

Кроме этого, в современных условиях реализация многих приложений осуществляется с помощью объектно-ориентированных языков

программирования, наиболее популярным из которых на сегодняшний момент является Java (*Oracle Corporation*). Отличительной особенностью Java в сравнении с другими языками программирования является обеспечение высокой продуктивности программирования и кроссплатформенности создаваемых приложений.

Взаимосвязь между реляционными базами данных, хранящими информацию о производственных проектах, и объектами прикладного решения на языке Java, то она может быть достигнута разными способами. Результаты анализа различных технологий (спецификаций, фреймворков) для разработки схемы обмена данными (маппинга) между объектами Java и таблицами реляционной базы данных приведены в таблице 1 [5,6].

Таблица 1 – Технологии хранения и преобразования данных

	Сериализация	JDBC	ORM	ODB	EJB2	JDO	JPA
Простота	+	+	+	+	-	-	+
ООП	+	-	+	+	-	+	+
Java объекты	+	-	+	+	+	+	+
Параллелизм	-	+	+	+	+	+	+
Транзакции	-	+	+	+	+	+	+
Схема данных	-	+	+	-	+	+	+
Работа с наборами данных	-	+	+	+	+	+	+
Хранение данных в разных форматах	-	-	-	-	+	+	-
Организация запросов к базе данных	-	+	+	+	+	+	+

Сериализация представляет собой встроенный механизм, который предназначен для хранения и передачи объектов в Java. Данный подход используется крайне редко, так как в этом случае потребуется хранить и использовать граф объектов целиком, что довольно трудоемко, особенно при организации работы с большими объемами данных.

JDBC (Java database connectivity) – это стандарт, разработанный специально для работы с реляционными базами данных. Данный подход имеет один главный недостаток – это отсутствие возможности проецировать реляционные данные на объекты, что приводит к увеличению текста программы для организации данного преобразования.

ORM (object/relational mapping) – это фреймворки для маппига объектов на реляционные таблицы от разных поставщиков. Использование данного подхода затруднено отсутствием стандартов, что привело к

большому числу несовместимых реализаций. В результате код написанной программы становится непереносимым.

ODB (object databases) обеспечивает объектные реализации баз данных. Основной недостаток такой же, как и у ORM.

EJB2 (Enterprise Java Beans) используется на платформе Java уровня предприятия. Обеспечивается работа с данными на уровне объектов, при этом способы физического хранения данных не ограничиваются только реляционными базами данных. Имеются ограничения на сам объектно-ориентированный подход, которые проявляются в отсутствии (сложности) реализации таких функций как наследование, полиморфизм и внешние связи объектов. Еще одним недостатком является потребность в высокопроизводительных серверах.

JDO (Java data objects) разрабатывался как Java ORM standard, который предназначен для работы с различными источниками данных.

JPA (Java persistence API) обеспечивает работу на уровне объектно-ориентированной модели. Существует возможность комбинирования доступа к данным на уровне реляционных данных, что обеспечивает более высокую производительность и гибкость по сравнению с JDO [6].

Наиболее эффективным способом организации взаимосвязи Java с реляционной БД является технология JPA. Стоит при этом отметить, что на сегодняшний момент разработано достаточно большое количество реализаций данной спецификации, например, Hibernate JPA, EclipseLink, TopLink, MyBatis и т.д.

Процесс взаимодействия приложения Java и реляционной базы данных посредством Hibernate проиллюстрирован рисунком 4.

Из рисунка видно, что в процессе преобразования используется описание связи классов приложения с таблицами базы данных в формате xml.

Работа с механизмом Hibernate начинается с установки параметров соединения с базой данных, которое осуществляется по средствам файла hibernate.cfg.xml (рисунок 5). В данном файле содержатся все необходимые сведения и параметры для подключения к БД, а также описываются связанные с базой классы. При осуществлении соединения применяется технология Java DataBase Connectivity (JDBC), для которой необходимый драйвер загружается из подключаемой библиотеки ojdbc7.jar. После этого переходят к инициализации механизма Hibernate.

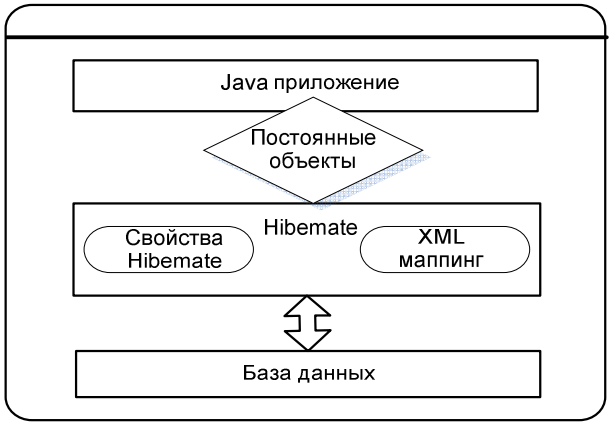

Рисунок 4 – Схема преобразования данных

Рисунок 5 – Листинг кода файла конфигурации

Стоит отметить, что работа с БД в данном случае осуществляется на базе транзакций в пределах сессии. В связи с чем, возникает потребность в создании фабрики сессий (рисунок 6). Сам механизм активируется по средствам служебного класса HibernateUtil.java (рисунок 7), в процессе загрузки которого происходит создание специального объекта sessionFactory, предназначенного для считывания данных из файла hibernate.cfg.xml и осуществления подключения к базе данных.

```
baza ▸ tppn ▸ util ▸ HibernateUtil ▸
package util;

import org.hibernate.SessionFactory;


@SuppressWarnings("deprecation")
public class HibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from hibernate.cfg.xml
              sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Make sure you log the exception, as it might be swallowed
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Рисунок 6 – Листинг программы для организации фабрики сессий

```
*Event.hbm.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="tppn">
    <class name="Event" table="EVENT">
        <id name="id_event" column="ID_EVENT">
            <generator class="native"/>
        </id>
        <property name="name_event" column="NAME_EVENT"/>
        <many-to-one name="Type_event" column="ID_EV_TYPE" class="tppn.baza.Event_type"/>
    </class>
</hibernate-mapping>
```

Рисунок 7 - XML – файл ассоциации класса приложения и таблицы БД

Процесс маппинга классов на таблицы БД, а также отношения между классами отражаются в файлах вида *name_class*.hbm.xml. Пример отображения класса EVENT (события) на таблицы базы данных представлен на рисунке 8 .

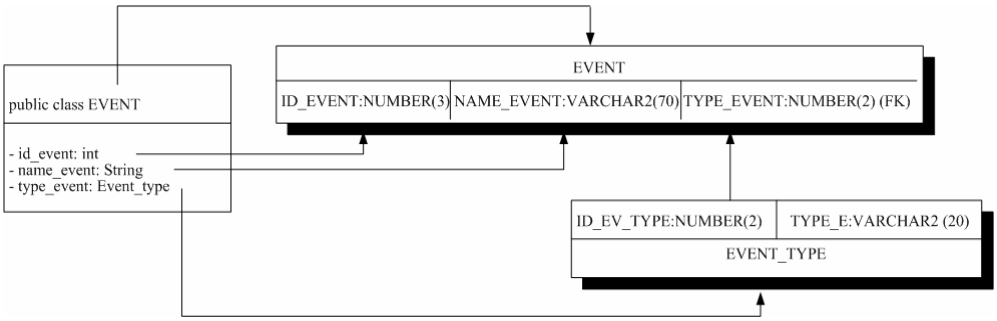

Рисунок 8 - Пример отображения класса на таблицы базы данных

В процессе построения описаний взаимосвязей классов и реляционных таблиц в формате xml основной проблемой является реализация отображений для атрибутов сложных типов (коллекций). Разработка шаблонов таких отображений для основных случаев является направлением дальнейших исследований. В целом, опыт работы с библиотекой Hibernate JPA позволяет

охарактеризовать ее как удобный и эффективный инструмент для организации преобразования объектного представления данных Java-приложений в структуру данных реляционных СУБД.

Библиографический список

1. Дли М.И., Стоянова О.В. Способы представления экспертных данных в системах поддержки принятия решений по управлению сложными проектами // Нейрокомпьютеры: разработка, применение. 2016. № 7. С. 21-28.
2. Стоянова О.В., Дли М.И., Белозерский А.Ю. Модели представления данных сложных производственных проектов в автоматизированных информационных системах промышленных предприятий // Программные продукты и системы. 2015. № 4. С. 210-218.
3. Стоянова О.В., Дли М.И., Васицына А.И. Анализ современных подходов к решению задачи построения моделей сложных проектов // Вестник Саратовского государственного технического университета. 2012. Т. 1. № 2 (64). С. 374-378.
4. Стоянова О.В., Дли М.И. Информационно-аналитическая система управления производственными проектами машиностроения в условиях неопределенности // Программные продукты и системы. 2015. № 3 (111). С. 49-56.
5. Литвинюк А. Введение в Hibernate и основы ORM // Компьютерная газета AZ [Электронный ресурс]. – Режим доступа: http://www.nestor.minsk.by/kg/2005/32/kg53209.html
6. Meyer B. Himbernate ORM & JPA Overview [Электронный ресурс] – Режим доступа: http://www.slideshare.net/brmeyer/orm-jpa-hibernate-overview