DOI 10.37882/2223-2966.2024.12.26

УНИВЕРСАЛЬНАЯ МОДЕЛЬ ДАННЫХ НА ОСНОВЕ ФОРМАТА JSON

A UNIVERSAL DATA MODEL BASED ON THE JSON FORMAT

O. Minkov

Summary. Today, we deal with a huge amount of different data, so there is a need to create universal models for such data in order to optimize and simplify interaction with it. The purpose of this article is to describe a universal data model based on the JSON format, and to compare the proposed model with existing analogues. The proposed model has a number of advantages, the main of which are: flexibility, ease of understanding, economy in terms of memory occupancy.

Keywords: universal data model, data, metadata, JSON, classifier, object, attribute, field.

Минков Олег Александрович
Аспирант, ИАТЭ НИЯУ МИФИ, Обнинск
oaminkov@gmail.com

Аннотация. Сегодня наблюдается колоссальный объём разнородных данных, поэтому возникает необходимость в создании универсальных моделей для таких данных с целью оптимизации и упрощения взаимодействия с ними. Целью статьи является описание универсальной модели данных на основе формата JSON, и сравнение предложенной модели с уже существующими аналогами. Предлагаемая модель обладает рядом преимуществ, основными из которых являются: гибкость, простота понимания, экономность с точки зрения занимаемой памяти.

Ключевые слова: универсальная модель данных, данные, метаданные, JSON, классификатор, объект, атрибут, поле.

Введение

На сегодняшний день человечество взаимодействует с колоссальным объёмом разнородных данных, которые необходимо обрабатывать, хранить, получать и передавать. Данные могут быть структурированными и неструктурированными, иметь различные форматы, относиться к разным областям знаний. В связи с этим возникает необходимость в разработке моделей, которые бы описывали те или иные данные, их формат, способ хранения и обработки. Существует множество моделей, которые основаны на реляционных, нереляционных базах данных (графовые, NoSQL), многомерных кубах, форматах на основе языка XML и т.д.

Разнообразие используемых моделей требует создания универсальной модели данных, которая была бы достаточно гибкой, легко понимаемой как машиной, так и человеком, а также простой при использовании. Уже имеются попытки создания универсальной модели на основе формата XML, но есть трудности обработки данных в XML при объемах данных больше 1 Мбайта. Есть также модели на основе так называемых триплов, когда все данные хранятся в виде набора из трёх полей: идентификатор, название атрибута и значение атрибута. Но в такой модели есть ряд минусов: её тяжело поддерживать, трудно читать человеку или требуется эффективное решение по постоянному преобразованию из плоской структуры в трипл и обратно.

Цель данной работы — предложить универсальную модель данных на основе формата JSON (JavaScript Object Notation), так как этот формат является сейчас очень популярным и востребованным. Многие языки программирования имеют удобные инструменты для работы с ним, а также данные в JSON намного легче воспринимаются человеком, чем в XML.

Обзор существующих моделей данных

Рассмотрим некоторые уже существующие универсальные модели данных, базирующиеся на реляционных базах данных и языке XML.

Первая такая модель доступна только на базе СУБД (Система Управления Базами Данных) Oracle и представляет собой структуру иерархически связанных таблиц [1]. Структура такой модели показана на рисунке 1.

Как мы можем наблюдать на рисунке 1, универсальная схема данных на базе Oracle имеет достаточно сложную структуру, состоящую из восьми таблиц, что достаточно много. Это обстоятельство и то, что Oracle — платная СУБД, являются серьёзными недостатками.

Еще одна попытка создания универсальной модели данных была сделана во ВНИИГМИ МЦД [2]. В этой модели выделена совокупность существенных признаков УМД:

1. Использование единого словаря описаний унифицированных атрибутов, в том числе атрибутов метаданных для описания объектов и событий.
2. Применение минимальной атомарной единицы хранения — значения атрибута данных с помощью многомерной схемы представления как для данных, так и метаданных. Многомерная модель

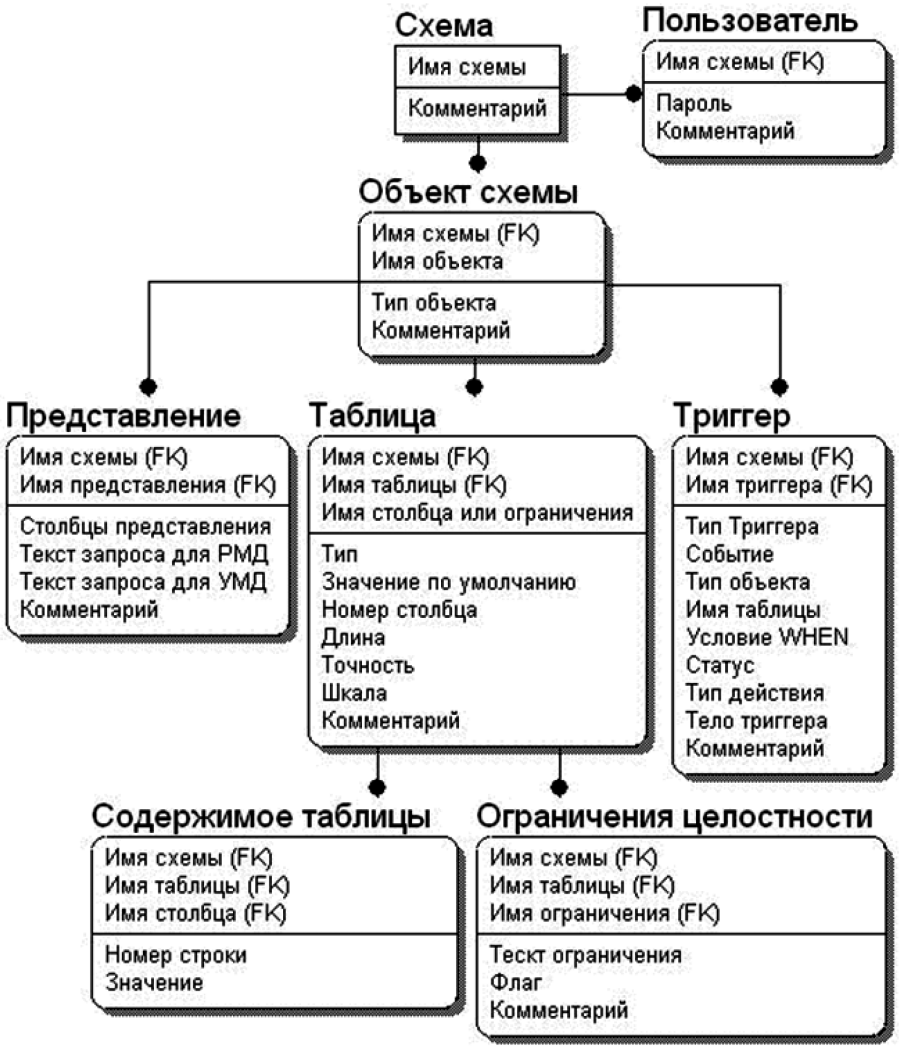


Рис. 1. Универсальная схема данных в СУБД Oracle [1]

данных позволяет хранить данные в виде трипла (идентификатор записи, имя атрибута, значение атрибута).

3. Создание единой схемы хранения всех классификаторов в одной таблице (ID, код, значение), что позволяет легко добавлять новые значения кодов в существующие классификаторы и новые классификаторы при настройке на предметную область.
4. Выделение простых сущностей (объекты метаданных, товары и т.п.) в виде отдельных объектов УМД.
5. Определение этапов жизненного цикла для всех выделяемых сущностей (объектов), предназначенных для хранения в УМД;
6. Выделение статической справочной (Каталог) и динамической (Факты) информации для каждого объекта в виде отдельных таблиц, связанных между собой и описывающих один объект.
7. Сохранение связей между таблицами «Каталог» и «Факты», экземплярами различных объектов в виде отдельной таблицы.
8. Хранение информации об объектах БД в виде «технологической информации» и их связях как один из объектов БД — «Сведения об состоянии объектов предметной области в БД».

На рисунке 2 показана предложенная схема универсальной модели данных.

Как видно из рисунка 2, предложенная схема состоит из меньшего числа таблиц и связей, чем схема в СУБД Oracle, рассмотренная ранее. На рисунке 3 продемонстрирован пример заполнения таблиц предложенной схемы.

Подобная структура не имеет строгой иерархической схемы, зависимости обнаруживаются с трудом, кроме того, присутствует некоторая избыточность универсальной модели, основанной на данной структуре.

Существует ещё одна УМД [2, 6]. Структура этой УМД переняла некоторые основы своей предшественницы,

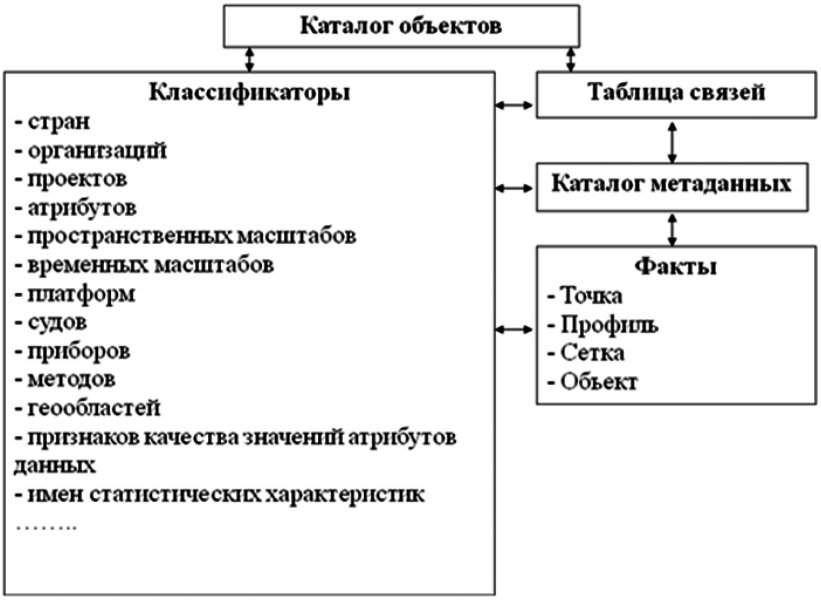


Рис. 2. Универсальная модель данных [2]

а) Метаданные б) Факты в) Связи г) Классификаторы

ID_Ekz	ID_Attr	Value		ID_Ekz	ID_Attr	Value		ID_Ekz	ID_Attr	Value		ID	ID_Attr	Value
10101	A1	**БД1**		10101	F1	**01-01-08**		101	ID_Obj_in	БД1		1	1	Ааааа
10102	A2	Name1		10102	F2	**01-12-08**		102	ID_Attr	Name1		1	2	Ббббб
10103	A3	**аааа**		10103	F3	**Иванов**		103	ID_EkzObj	1		3	ГОПН	Ввввв
10104	A4	**100**		10201	F1	**01-12-07**		104	ID_Obj_out	Param		3	НСА	Гггггг
20201	A1	**БД2**		10202	F2	**01-10-08**		105	ID_EkzObj	Par1		3	НЭМ	Ддддд
20202	A2	**Name2**		10203	F3	**Петров**		201	ID_Obj_in	БД2		2	А	Анализ
20203	A5	**НСА**		10301	F1	**15-12-07**		202	ID_Attr	Geo		2	К	Климат
…..	…….	….		…..	…..	….		…..	……..	….		…	…..	……

Рис. 3. Пример реализации предложенной схемы УМД [2]

ID_o	Name	Value
1	#Class	1
2		3
…	…	…
N		

ID_	Name	Value
1	#Object	1
1	#Object	2
1	#Object	4
3	#Object	18
3	#Object	23
3	#Object	50
…	…	…
N		

ID	Name	Value
1	Atr1	a
1	Atr2	b
1	Atr3	c
2	Atr1	d
2	Atr2	e
2	Atr3	f
18	Atr1	g
…	…	…
N		

Рис. 4. Пример структуры УМД [2, 6]

в основе всё также лежит понятие трипла, однако, вводится несколько видов триплов, под каждый из которых выделяется своя таблица. Новая схема вносит в универсальную модель данных строгую иерархию видов триплов. Иерархия основана на принципах объектно-ориентированного программирования. Пример предложенной структуры УМД продемонстрирован на рисунке 4.

В первой таблице содержатся триплы, обозначающие отдельный класс объектов, в поле «name» они содержат служебное слово #class. Во второй таблице содержатся триплы, обозначающие объекты классов, в поле «name» они содержат служебное слово #object. Третья таблица нужна для хранения атрибутов. Любые дополнительных параметры, такие как имя класса, пакет, в котором класс находится и т.п. заносятся в четвертую таблицу с идентификатором, который содержится в таблице класса в поле «value». Также с данным идентификатором будут заноситься объекты соответствующего класса во вторую таблицу по мере их появления. Т.е. таблицы связаны типом один-ко-многим по цепочке от первой к третьей, что наглядно показано на рисунке 4. Предложенная структура обеспечивает пользователя возможностью добавлять или удалять какой-либо атрибут в объекте без изменения схемы базы данных. Так как между классами, объектами и их свойствами существует однозначная связь, то можно образовать хеш этих связей для конкретных экземпляров триплов, фактически добавить индексы, улучшив тем самым производительность модели.

Помимо основанных на базах данных УМД, рассмотренных выше, существует множество моделей, которые базируются на языке XML. Всех их объединяет одно: данные хранятся в файлах на языке XML, а для обработки этих данных используется XML-схема. Одним из примеров использования языка XML является формат XSIL (Extensible Scientific Interchange Language), разработанный Центром компьютерных исследований (Center for Advanced Computing Research, CACR) [3, 4]. Это гибкий, иерархический, расширяемый язык для описания научных данных. Как всякий документ XML, файл XSIL является иерархией элементов, каждый из которых представлен тэгом, несколькими атрибутами, текстом и может содержать другие элементы. XSIL предоставляет синтаксическую структуру для распространенных научных данных (таблиц, массивов и т. п.) и механизм для расширения XML и объектов Java. В XSIL входит также расширяемая платформа для визуализации Xlook. Кроме того, файл XSIL может быть использован и для непосредственного просмотра в веб-браузере. Библиотека может быть расширена классами, написанными пользователем для специфичных типов данных [7].

Все рассмотренные УМД имеют как достоинства, так и недостатки. Чтобы побороть некоторые недостатки, было принято решение создать новую УМД.

Описание предлагаемой модели данных

В основе предлагаемой модели данных лежит JSON — популярный формат данных, широко используемый в REST-сервисах для обмена данными между приложениями. Большинство REST-сервисов для взаимодействия друг с другом и с клиентскими приложениями используют JSON.

Предлагаемая модель состоит из двух типов файлов. Первый тип — схема (или классификатор), в котором хранится общая информация (метаданные), связи с другими схемами и набор полей с атрибутами, характерными для объектов этой схемы.

Основные поля классификатора включают:

- id — уникальный в рамках всех схем идентификатор в формате GUID (Globally Unique Identifier);
- name — название схемы, предлагается сделать его уникальным для каждой схемы;
- created — дата и время создания схемы;
- description — текстовое описание схемы;
- relations — массив, содержащий связи текущей схемы с другими схемами. Одна связь выражается двумя полями: id и name связанной схемы;
- fields — массив полей, которые могут содержать конкретные объекты, относящиеся к рассматриваемой схеме. Одно поле выражается тремя атрибутами: id, name и type поля. Type может быть численным, строковым, булевым и т.д.

Пример общей структуры схемы представлен на рисунке 5.

Помимо предложенных полей схемы, модель можно дополнить другими полями, к примеру, owner, которое содержало бы информацию о владельце схемы.

Второй тип — объект (или сущность), в котором хранится конкретный набор данных, например, какие-либо измерения, данные наблюдений и т.д.

Как и схема, объект имеет набор обязательных полей. Предлагаются следующие поля:

- id — уникальный в рамках объектов одной схемы идентификатор объекта в формате GUID;
- name — уникальное в рамках объектов одной схемы имя объекта в строковом формате;
- class_id — идентификатор схемы, к которому относится объект;
- class_name — название схемы, к которому относится объект. Данное поле предлагается для лучшего восприятия объекта человеком;
- created — дата и время создания объекта;
- fields — массив полей, которые содержит объект. Могут содержать любые данные, однако, для

```
{
    "id": "GUID классификатора",
    "name": "Название классификатора",
    "created": "Дата и время создания классификатора",
    "description": "Описание классификатора",
    // набор полей, содержащих метаданные (источник, владелец и т.д.)
    "relations": [
        {
            "id": "GUID связанного классификатора",
            "name": "Название связанного классификатора"
        }
    ],
    "fields": [
        {
            "id": "GUID поля",
            "name": "Название поля"
            "type": "Тип поля"
        }
    ]
}
```

Рис. 5. Общая структура схемы

```
{
    "id": "GUID сущности",
    "name": "Название сущности",
    "class_id": "GUID классификатора, который описывает данную сущность",
    "class_name": "Название классификатора",
    "created": "Дата и время создания сущности",
    // дополнительные метаданные (владелец и т.д.)
    "fields": [
        {
            "id": "GUID поля",
            "name": "Название поля",
            "type": "Тип поля",
            "value": "Значение поля"
        }
    ]
}
```

Рис. 6. Общая структура объекта

лучшей совместимости предлагается, чтобы поля объекта соответствовали полям схемы, к которому относится данный объект.

Массив field в свою очередь состоит из набора json-объектов, каждый из которых содержит следующие поля:

- id — уникальный в рамках полей сущности идентификатор поля в формате GUID;
- name — уникальное в рамках полей сущности название поля;
- type — тип поля. Может быть строковым, числовым, булевым, отражать связь с другими объектами и т.д.;
- value — значение поля. В зависимости от типа поля может быть простым значением, json-объектом, массивом.

Пример общей структуры объекта представлен на рисунке 6.

Так же, как и схема, объект можно дополнить новыми полями, если они необходимы.

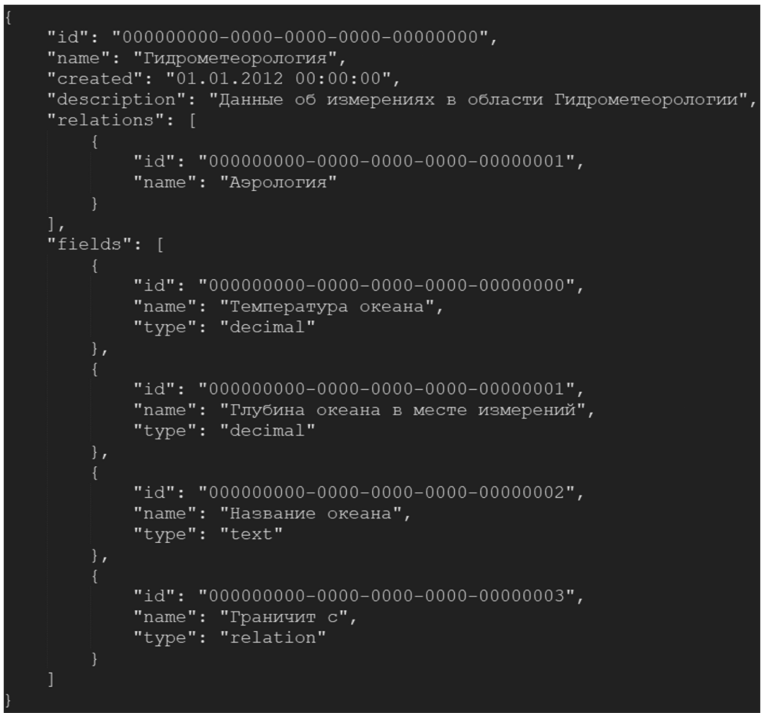
Представленные структуры позволяют описать любой набор структурированных данных, которые затем будут легко восприниматься как машинами, так и людьми. Пример конкретной схемы представлен на рисунке 7. Пример конкретного объекта представлен на рисунке 8.

Сравнение предлагаемой модели с существующими моделями

Краткое сравнение рассмотренных ранее универсальных моделей данных приведено в таблице 1.

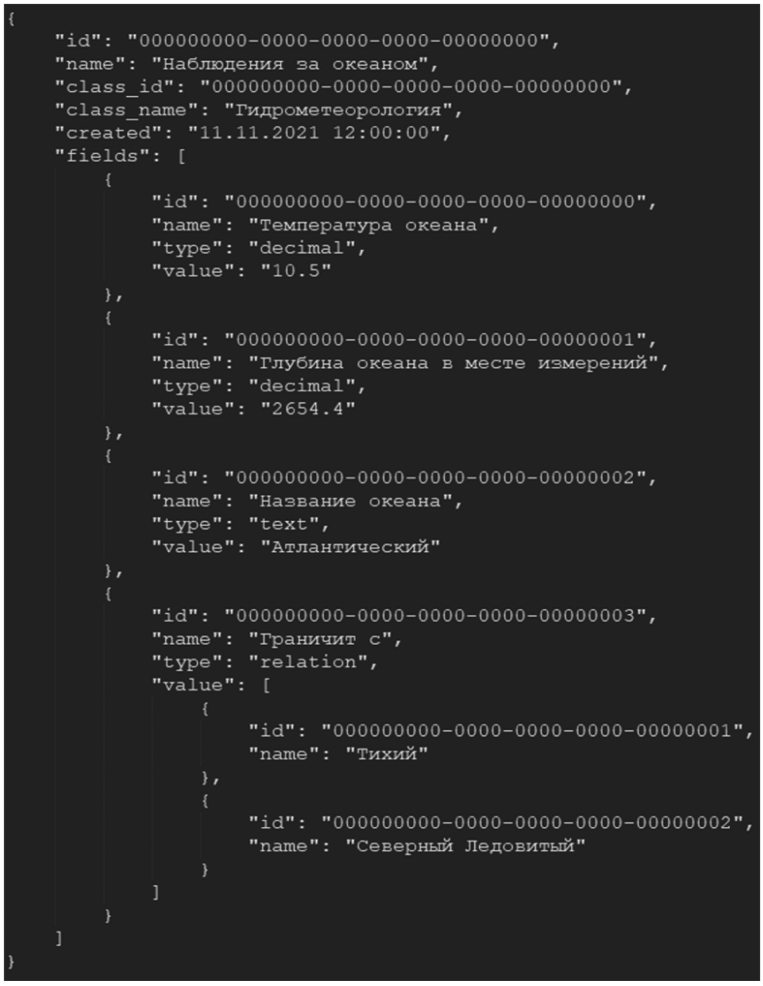
Предлагаемая модель имеет ряд преимуществ перед существующими универсальными моделями данных. Для демонстрации достоинств модели данных на основе JSON проведём её сравнение с двумя популярными моделями на основе XML и триплов.

Модель данных на основе XML в целом имеет схожую структуру с предложенной моделью на основе JSON. В ней тоже есть схема, которая представляет собой файл формата XSD (XML Schema definition) и хранит описание структуры данных. Сами данные представляют собой XML файлы, что аналогично объектам в предлагаемой модели на основе JSON. Однако, основные преимущества JSON модели перед XML состоят в её лаконичности и простоте понимания человеком. Если мы посмотрим



```
{
    "id": "000000000-0000-0000-0000-00000000",
    "name": "Гидрометеорология",
    "created": "01.01.2012 00:00:00",
    "description": "Данные об измерениях в области Гидрометеорологии",
    "relations": [
        {
            "id": "000000000-0000-0000-0000-00000001",
            "name": "Аэрология"
        }
    ],
    "fields": [
        {
            "id": "000000000-0000-0000-0000-00000000",
            "name": "Температура океана",
            "type": "decimal"
        },
        {
            "id": "000000000-0000-0000-0000-00000001",
            "name": "Глубина океана в месте измерений",
            "type": "decimal"
        },
        {
            "id": "000000000-0000-0000-0000-00000002",
            "name": "Название океана",
            "type": "text"
        },
        {
            "id": "000000000-0000-0000-0000-00000003",
            "name": "Граничит с",
            "type": "relation"
        }
    ]
}
```


Рис. 7. Пример схемы



```
{
    "id": "000000000-0000-0000-0000-00000000",
    "name": "Наблюдения за океаном",
    "class_id": "000000000-0000-0000-0000-00000000",
    "class_name": "Гидрометеорология",
    "created": "11.11.2021 12:00:00",
    "fields": [
        {
            "id": "000000000-0000-0000-0000-00000000",
            "name": "Температура океана",
            "type": "decimal",
            "value": "10.5"
        },
        {
            "id": "000000000-0000-0000-0000-00000001",
            "name": "Глубина океана в месте измерений",
            "type": "decimal",
            "value": "2654.4"
        },
        {
            "id": "000000000-0000-0000-0000-00000002",
            "name": "Название океана",
            "type": "text",
            "value": "Атлантический"
        },
        {
            "id": "000000000-0000-0000-0000-00000003",
            "name": "Граничит с",
            "type": "relation",
            "value": [
                {
                    "id": "000000000-0000-0000-0000-00000001",
                    "name": "Тихий"
                },
                {
                    "id": "000000000-0000-0000-0000-00000002",
                    "name": "Северный Ледовитый"
                }
            ]
        }
    ]
}
```


Рис. 8. Пример объекта

Таблица 1.

Сравнение рассмотренных моделей данных

Модель	Универсальность	Единица хранения	Сложность	Достоинства	Недостатки
Модель на основе XML	Широкая	Объект	Средняя	Можно описать любую предметную область. Предназначена для обмена данными	Многословность, накладные расходы на память
Модель на основе триплов	Широкая	Атрибут в виде трипла (ИД, код атрибута, значение)	Сложная	Позволяет представить любой объект в виде 4 таблиц (метаданные, факты, классификаторы, связи)	Высокая сложность реализации постоянного конвертирования из УМД в плоскую структуру и обратно. Трудна для понимания
Модель на основе JSON	Широкая	Объект в виде записи	Средняя	Можно описать любую предметную область. Предназначена для обмена данными	Сложности с реализацией конвертации в модель на основе JSON и обратно

на структуру XML, мы увидим, что в ней содержится множество громоздких тегов, которые тяжело читать, и к тому же из-за этих тегов увеличивается объём хранимой информации. Если для малого набора данных это не критично, то при большом объёме это более чем заметно. JSON не имеет таких проблем, так как его намного легче воспринимать человеку за счёт простого и понятного синтаксиса, к тому же данные, хранимые в формате JSON занимают меньше места, что позволяет экономить память при хранении и трафик при обмене этими данными через Интернет. Помимо прочего, JSON сейчас более популярный формат, нежели XML.

Модель данных на основе триплов является отличным решением, позволяющим эффективно хранить любые данные. Однако, у неё имеются серьезные недостатки. Основной недостаток кроется в том, что для работы такой модели необходима СУБД, установленная на устройстве, на котором необходимо с данной моделью работать. Второй важный недостаток — плохая читаемость человеком, так как все данные представляют собой набор записей из трёх полей. Предложенная модель на основе JSON качественно выделяется на данном фоне благодаря своей простоте и понятности. Данные в предложенной модели представляют собой набор текстовых файлов, которые поддерживаются по умолчанию большинством операционных систем. А это значит, что мы можем без проблем и без установки дополнительного программного обеспечения просматривать и даже вручную редактировать эти файлы. Про удобную и понятную читаемость такой модели уже было сказано ранее.

Заключение

В рамках данной работы была впервые предложена и описана универсальная модель в формате JSON для структурированных данных. Проведено сравнение предложенной модели с двумя другими моделями, в формате XML и триплов. Доказано, что модель в формате JSON обладает преимуществами — лаконичность и простота понимания человеком.

В дальнейшем планируется программно реализовать конвертацию данных в предложенную модель и из неё.

ЛИТЕРАТУРА

1. Муса-Оглы Е., Бессарабов Н. Универсальная модель данных в Oracle. [Электронный ресурс]. — Режим доступа: https://www.interface.ru/home.asp?artId=24052
2. Вязилов Е.Д., Федорцов А.А. Универсальная модель хранения данных с учетом жизненного цикла объектов // Шестая Всероссийская Открытая конференция «Современные проблемы дистанционного зондирования земли из космоса (Физические основы, методы и технологи и мониторинга окружающей среды, потенциально опасных явлений и объектов). — Москва, ИКИ РАН, 10–14 ноября 2008.
3. Williams, R. 2000, Extensible Scientific Interchange Language (XSIL),http://www.cacr.caltech.edu/XSIL.
4. Dick, D. 2020. Office open XML. Available at http://officeopenxml.com/anatomyofOOXML.php
5. Пергаменцев Ю. Проектирование БД на основе универсальной модели данных // «Вега-ЛТД». Материалы конференции «Корпоративные БД 2002». [Электронный ресурс]. — Режим доступа: http://www.citforum.ru/seminars/cbd2002/111.shtml, свободный. — Загл. с экрана.
6. Вязилов Е.Д. Создание и использование баз данных. Palmarium Academic Publishing, Germany. 2018. 545 с.
7. Melzer, S., Thiemann, S., Schiff, S. and Möller, R., 2023. Implementation of a Federated Information System by Means of Reuse of Research Data Archived in Research Data Repositories. Data Science Journal, 22(1), p.39.DOI: https://doi.org/10.5334/dsj-2023-039

Журнал «Современная наука: актуальные проблемы теории и практики»