УДК 004.942

В.В. Моисеев, Н.Г. Ярушкина

НАЧАЛЬНАЯ МОДЕЛЬ ДАННЫХ ПРЕДМЕТНОЙ ОБЛАСТИ НА ОСНОВЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

Моисеев Владислав Валерьевич, *аспирант кафедры «Информационные системы» Ульяновского государственного технического университета, окончил факультет информационных систем и технологий УлГТУ. Преподаватель Колледжа экономики и информатики УлГТУ. Имеет работы в области интеграции данных. [e-mail: v.v.moiseev@ulstu.ru].*

Ярушкина Надежда Глебовна, *доктор технических наук, профессор, окончила Ульяновский политехнический институт. Исполняющая обязанности ректора УлГТУ, заведующая кафедрой «Информационные системы» УлГТУ. Имеет более 300 работ в области мягких вычислений, нечеткой логики, гибридных систем. [e-mail: jng@ulstu.ru].*

Аннотация

В данной статье приводится описание модели данных для получения информации о структуре реляционной базы данных. Данная модель полезна для задачи уменьшения трудозатрат при проектировании и перепроектировании существующих программных продуктов. В текущий момент данные процессы зачастую сопровождаются анализом излишне крупных реляционных схем, что в конечном итоге увеличивает время разработки.

Модель, представленная в работе, позволяет получить из реляционной схемы данные, которые впоследствии можно применить для составления других более гибких формализаций. Также модель является расширяемой дополнительными параметрами, которые не удаётся получить из реляционной схемы, однако которые будут полезны для задачи проектирования.

В работе приводятся примеры запросов на языке SQL по получению информации о структуре реляционной базы данных на примере системы управления базами данных SQL Server, а также проводится сравнительный анализ предложенной модели по объёму и скорости формирования с популярными форматами: DBML и EDMX.

Ключевые слова: реляционная схема данных, модель данных, SQL Server, EDM, DBML.

doi: 10.35752/1991-2927-2019-4-58-51-56

INITIAL DOMAIN DATA MODEL BASED ON RELATIONAL DATABASE

Vladislav Valerevich Moiseev, *Postgraduate Student at the Department of Information Systems of Ulyanovsk State Technical University; graduated from the Faculty of Information Systems and Technologies of UlSTU; a lecturer at the College of Economics and Informatics of UlSTU; an author of articles in the field of data integration. e-mail: v.v.moiseev@ulstu.ru.*

Nadezhda Glebovna Yarushkina, *Doctor of Sciences in Engineering, Professor; graduated from Ulyanovsk Polytechnic Institute; Acting Rector of UlSTU, Head of the Department of Information Systems at UlSTU; an author of more than 300 publications in the field of soft computing, fuzzy logic, and hybrid systems. e-mail: jng@ulstu.ru.*

Abstract

This article describes the data model for obtaining information about the structure of a relational database. This model is useful for the task of reducing labor costs when designing and redesigning the existing software products. Currently, these processes are often accompanied by analysis of excessively large relational schemes, which ultimately increases development time.

The model presented in the paper allows one to obtain data from a relational scheme that can subsequently be used to compile other more flexible formalizations. In addition, the model is expandable with extra parameters that cannot be obtained from the relational scheme, but which will be useful for the design task.

The paper presents queries in SQL language for obtaining data on the structure of a relational database using the example of the SQL Server database management system and provides a comparative analysis of the proposed scheme in terms of volume and speed of formation with popular formats: DBML and EDMX.

Key words: relational data schema, data model, SQL Server, EDM, DBML.

Введение

В связи с возросшей сложностью информационных систем и программных комплексов необходимо разрабатывать новые методы по их проектированию и перепроектированию, которые должны быть более оптимальными и менее трудозатратными по сравнению с существующими.

В конце XX века в компьютерных науках и математической логике возникает идея формального описания предметных областей [1]. Для этого используют онтологии – специальные формализации, с помощью которых можно искать противоречия и зависимости между концептами и другими элементами предметной области. Такая формализация является настолько гибкой, что на её основе можно строить другие формальные модели предметных областей, например: реляционную модель данных; объектно-ориентированную схему классов на языке UML или любом другом языке программирования, поддерживающем принципы объектно-ориентированного программирования (ООП). Также с помощью онтологий можно выстраивать правила интеграции данных между различными программными продуктами. Уже сейчас существует множество стандартов передачи структурированных данных. Например, одним из самых популярных является протокол SOAP, при котором схема данных обычно описывается в виде XSD-документов [2].

Обладая такими свойствами, применение онтологий и элементов онтологического анализа крайне полезно для проектирования и перепроектирования программных продуктов ввиду присутствия аксиоматики при описании предметной области.

Однако в текущий момент существует множество крупных информационных систем и программных продуктов, для которых онтология предметной области не составлена. С другой стороны, во множестве программных продуктов существует реляционное хранилище данных, в котором уже есть начальные сведения для построения такой формализации. Таким образом, для более оптимального проектирования и перепроектирования программных продуктов необходимо каким-либо образом преобразовать данное реляционное хранилище в онтологический ресурс. Из-за различной природы исходной и результирующей модели данный процесс является нетривиальным [3]. Например, прямое преобразование реляционной модели в онтологию (например, при использовании подходов из [4] и [5]) может породить некачественную модель ввиду следующих причин:

• не все элементы реляционной модели требуют преобразования;

• не все данные о предметной области можно извлечь из реляционной модели;

• существует множество нестандартизированных особенностей в различных физических реализациях реляционных моделей.

Дополнительной трудностью при извлечении метаданных из реляционного хранилища крупных программных продуктов являются время выполнения данного процесса и объём занимаемого дискового пространства результирующей структуры данных.

Таким образом, для решения задачи упрощения процессов проектирования и перепроектирования программных продуктов путём внедрения элементов онтологического анализа требуется создание специальной модели, которая должна отвечать следующим характеристикам:

• возможность исключать некоторые элементы реляционной схемы без потери согласованности модели;

• возможность включать дополнительные свойства, которые могут быть полезны, но отсутствуют в реляционной схеме;

• независимость от конкретной реализации реляционного хранилища данных;

• время получения данных для формирования модели должно быть линейно по отношению к количеству элементов модели;

• объем сериализованных структурированных данных модели должен быть сопоставим с аналогичными моделями, применяемыми в настоящее время.

Назовём предлагаемую модель начальной, потому что она будет инкапсулировать в себе особенности конкретных реализаций реляционных хранилищ данных. Также модель можно будет использовать для построения других формализаций.

1 Выбор инструментальных средств для предлагаемой модели

Существует множество реализаций реляционных хранилищ в виде систем управления базами данных (СУБД). Одной из таких СУБД, популярной в промышленной эксплуатации, является Microsoft SQL Server (MSSQL). Данная СУБД выполнена по стандарту ANSI/ISO SQL:2003. В соответствии с этим, MSSQL поддерживает информационную базу данных (БД) *INFORMATION_SCHEMA*, в которой содержится информация о структуре и схеме других БД. Дополнительно с этим MSSQL поддерживает собственную схему *sys* для более детального доступа к настройкам и метаданным БД.

Данная СУБД поддерживает свойства расширения, которые помогают указывать любые текстовые метаданные для схем, таблиц, полей. Наиболее популярным свойством является *MS_Description*, в котором можно задать описание любых элементов модели БД.

MSSQL поддерживает процедуры в виде хранимых процедур и функций. Для формирования онтологии информация о них также может быть полезна. Доступ к процедурам, их параметрам и полям результирующей таблицы также указывается в информационной БД *INFORMATION_SCHEMA*.

Таким образом, инструментария MSSQL по извлечению метаданных достаточно для формирования начальной модели, которая впоследствии может быть преобразована в онтологию.

В качестве записи начальной модели решено использовать язык разметки XML (eXtended Markup Language) из-за его широкой поддержки, стандартных механизмов схематизации в виде XML Schema Definition (XSD), различных возможностей использования и лёгкости в поддержке при командной разработке.

2 Описание начальной модели данных для формирования онтологии

Пусть начальная модель данных для формирования онтологии из реляционной схемы обозначается как M. Тогда модель можно представить в виде выражения:

$$M = \langle E, R \rangle, \quad (1)$$

где $E = \{E_1, E_2, ..., E_n\}$ – множество сущностей реляционной схемы;

$R = \{R_1, R_2, ..., R_m\}$ – множество отношений между сущностями;

n – количество сущностей;

m – количество отношений.

Сущность можно описать следующим образом:

$$E_i = (Name, A, C), \quad (2)$$

где $Name$ – название сущности i;

$A = \{A_1, A_2, ..., A_n\}$ – множество атрибутов сущности i;

$C = \{C_1, C_2, ..., C_n\}$ – множество ограничений сущности i;

n – количество атрибутов у сущности i.

Атрибут сущности описывается следующим образом:

$$A_i = (Num, Name, Type, P), \quad (3)$$

где Num – порядковый номер атрибута i в кортеже атрибутов сущности;

$Name$ – название атрибута, уникальное в рамках сущности;

$Type = [\text{Int, Float, Double, Char, Decimal, Bit}]$ – тип данных;

$P = \{P_1, P_2, ..., P_n\}$ – множество дополнительных свойств атрибута.

Дополнительные свойства атрибута могут быть следующими:

- максимальная длина (для строковых полей);
- максимальное количество цифр (для целых и десятичных чисел);
- количество знаков после запятой (для десятичных чисел);
- текстовое описание.

Ограничение сущности P может быть записано следующим образом:

$$P_i = F(C), \quad (4)$$

где C – множество атрибутов, к которым применимо ограничение;

F – функция, представляющая вид ограничения.

Ограничения могут принимать следующий вид:

- F_{PK} – ограничение первичного ключа (primary key constraint);
- F_{NN} – ограничение обязательности (not null constraint);
- F_{UQ} – ограничение уникальности (unique constraint);
- F_{FK} – ограничение внешнего ключа (foreign key constraint).

Отношение между сущностями можно описать следующим образом:

$$R = D(F_{FK,i}, F_{PK,j}, RP), \quad (5)$$

где D – функция, описывающая вид отношения;

$F_{FK,i}$ – ограничение внешнего ключа сущности i;

$F_{PK,j}$ – ограничение первичного ключа сущности j;

$RP = \{RP_1, RP_2, ..., RP_n\}$ – множество дополнительных параметров отношения.

На основании (2), (4) и (5) можно сделать вывод, что определение отношения R можно упростить до:

$$R = E_i o_R E_2 : \{E_i, E_j\} \subseteq E, \quad (6)$$

где o_R – операция связывания сущностей E_i и E_j.

Запись полученной модели предлагается производить при помощи языка разметки XML с использованием следующих правил:

- ключевой элемент модели – *Schema*;
- элементы описания сущностей – *Entity*;
- элементы описания атрибутов – *Column*;
- названия сущностей, атрибутов и ограничений указываются при помощи атрибутов *Schema* и *Name*;
- ограничения первичного ключа определяются атрибутом *IsPrimaryKey*;
- ограничения обязательности определяются атрибутом *IsNullable*;
- ограничения внешнего ключа определяются элементами *ForeignKey*.

3 Получение данных для модели из СУБД SQL Server

Для заполнения предложенной модели необходимо выполнить запросы в БД для получения сведений о ней.

Как упоминалось ранее, Microsoft SQL Server реализует стандарт ANSI/ISO SQL:2003 и добавляет собственную схему *sys*, которая может пригодиться для рефлексии, поэтому запросы к БД могут выполняться с ошибками на других СУБД.

Изначально необходимо получить список сущностей: таблиц и табличных представлений. Для каждой сущности также следует найти текстовое описание из свойства расширения *MS_Description*. Для выполнения этих действий используются таблицы *sys.tables*, *sys.views*, *sys.schemas*, а также функция *sys.fn_listextendedproperty*. Представление запроса в нотации Transact-SQL (здесь и далее будет использоваться нотация Transact-SQL):

```
SELECT
    s.name AS [SchemaName],
    t.name AS [Name],
    'T' AS [Type],
    ep.value AS [Description]
FROM sys.tables t
    JOIN sys.schemas s ON s.schema_id = t.schema_id
    OUTER APPLY sys.fn_listextendedproperty(N'MS_Description', N'SCHEMA', s.name, N'TABLE', t.name, NULL, NULL) ep
UNION ALL
SELECT
    s.name AS [SchemaName],
    v.name AS [Name],
    'V' AS [Type],
    ep.value AS [Description]
FROM sys.views v
    JOIN sys.schemas s ON s.schema_id = v.schema_id
    OUTER APPLY sys.fn_listextendedproperty(N'MS_Description', N'SCHEMA', s.name, N'TABLE', v.name, NULL, NULL) ep
ORDER BY SchemaName, Type, Name;
```

Запрос на получение списка атрибутов сущностей с их описаниями является относительно сложным, так как дополнительно требуется искать их вхождение в первичных ключах. Запрос в нотации Transact-SQL:

```
SELECT
    c.TABLE_NAME AS [Table],
    c.COLUMN_NAME AS [Name],
    c.DATA_TYPE AS [DataType],
    c.ORDINAL_POSITION AS [OrdinalPosition],
    ISNULL(CAST(CASE WHEN c.IS_NULLABLE = 'YES' THEN 1 ELSE 0 END AS BIT), 0) AS [IsNullable],
    ISNULL(CAST(CASE WHEN pkc.CONSTRAINT_NAME IS NOT NULL THEN 1 ELSE 0 END AS BIT), 0) AS [IsPrimaryKey],
    sc.is_identity [IsIdentity],
    sc.is_computed [IsComputed],
    ep.value AS [Description],
    ept.value AS [TableDescription]
FROM INFORMATION_SCHEMA.COLUMNS c
    JOIN sys.columns sc ON sc.object_id = OBJECT_ID(c.TABLE_SCHEMA+'.'+c.TABLE_NAME) AND sc.name = c.COLUMN_NAME
    OUTER APPLY sys.fn_listextendedproperty(N'MS_Description', N'SCHEMA', c.TABLE_SCHEMA, N'TABLE', c.TABLE_NAME, N'COLUMN', c.COLUMN_NAME) ep
    OUTER APPLY sys.fn_listextendedproperty(N'MS_Description', N'SCHEMA', c.TABLE_SCHEMA, N'TABLE', c.TABLE_NAME, NULL, NULL) ept
    LEFT JOIN (
      SELECT cu.TABLE_CATALOG, cu.TABLE_SCHEMA, cu.TABLE_NAME, cu.COLUMN_NAME, tc.CONSTRAINT_NAME
      FROM INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE cu
        JOIN INFORMATION_SCHEMA.TABLE_CONSTRAINTS tc ON tc.CONSTRAINT_CATALOG = cu.CONSTRAINT_CATALOG AND tc.CONSTRAINT_NAME = cu.CONSTRAINT_NAME AND tc.CONSTRAINT_SCHEMA = cu.CONSTRAINT_SCHEMA
        WHERE tc.CONSTRAINT_TYPE = 'PRIMARY KEY'
    ) pkc ON pkc.TABLE_SCHEMA = c.TABLE_SCHEMA AND pkc.TABLE_NAME = c.TABLE_NAME AND pkc.COLUMN_NAME = c.COLUMN_NAME AND pkc.TABLE_CATALOG = c.TABLE_CATALOG
WHERE c.TABLE_SCHEMA = @schema AND c.TABLE_NAME IN ({names})
ORDER BY c.TABLE_SCHEMA, c.TABLE_NAME, c.ORDINAL_POSITION;
```

Стоит отметить использование параметра *@schema* и заменителя {*names*}. Данная часть запроса служит для ускорения получения сведений из реляционной БД. Такая конструкция будет встречаться и в других запросах.

Получение внешних ключей описывается стандартной информационной БД *INFORMATION_SCHEMA*. Текст запроса SQL:

```
SELECT
    KCU1.CONSTRAINT_NAME AS FkConstraintName
   ,KCU1.TABLE_SCHEMA AS FkTableSchema
   ,KCU1.TABLE_NAME AS FkTableName
   ,KCU1.COLUMN_NAME AS FkColumnName
   ,KCU1.ORDINAL_POSITION AS FkOrdinalPosition
   ,KCU2.CONSTRAINT_NAME AS ReferencedConstraintName
   ,KCU2.TABLE_SCHEMA AS ReferencedTableSchema
   ,KCU2.TABLE_NAME AS ReferencedTableName
   ,KCU2.COLUMN_NAME AS ReferencedColumnName
   ,KCU2.ORDINAL_POSITION AS ReferencedOrdinalPosition
FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS AS RC

INNER JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE AS KCU1
   ON KCU1.CONSTRAINT_CATALOG = RC.CONSTRAINT_CATALOG
   AND KCU1.CONSTRAINT_SCHEMA = RC.CONSTRAINT_SCHEMA
   AND KCU1.CONSTRAINT_NAME = RC.CONSTRAINT_NAME

INNER JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE AS KCU2
   ON KCU2.CONSTRAINT_CATALOG = RC.UNIQUE_CONSTRAINT_CATALOG
   AND KCU2.CONSTRAINT_SCHEMA = RC.UNIQUE_CONSTRAINT_SCHEMA
   AND KCU2.CONSTRAINT_NAME = RC.UNIQUE_CONSTRAINT_NAME
   AND KCU2.ORDINAL_POSITION = KCU1.ORDINAL_POSITION
WHERE KCU1.TABLE_SCHEMA = @schema AND KCU1.TABLE_NAME IN ({names});
```

Следует обратить внимание на сортировку результирующих данных в запросах (оператор *ORDER BY*). Именно такую сортировку следует использовать, чтобы изменение предложенной модели данных не вызывало конфликтов при командной работе при помощи систем управления версиями (например, git или subversion). В противном случае при одновременном добавлении сведений о БД разными специалистами изменение модели данных может записаться в одно и то же место сериализованного XML-документа, что приведёт к конфликту.

4 Сравнение с другими моделями

Существует множество аналогичных моделей данных, получаемых из различных реляционных СУБД.

Например, в технологии Linq2Sql используется нотация DBML (Database Markup Language) [6].

Данная модель похожа на описанную ранее, однако обладает рядом недостатков:

- отсутствует описание сущностей и атрибутов;
- наличие суррогатных идентификаторов у некоторых процедур;
- отсутствие сортировки. Таким образом, ухудшается работа над моделью при командной разработке;
- излишне усложнённый механизм настройки сущностей;
- отсутствие открытых и поддерживаемых средств для синхронизации схемы БД с DBML.

Более современной и универсальной технологией построения модели данных является EDM (Entity Data Model) с форматом данных EDMX (подробности в [7]).

В этой модели идёт разделение на следующие составляющие:

- модель хранилища;
- концептуальную модель;
- сопоставления этих моделей;
- другие элементы и диаграммы.

Благодаря указанному разделению можно гибко настроить отображение хранилища данных на объектную модель. Однако это является избыточным для задачи формирования первичной модели данных предметной области.

У технологии есть стандартный дизайнер, встроенный в IDE Visual Studio, который поддерживает синхронизацию модели с БД.

Недостатки модели EDM для текущей задачи:

- излишняя сложность;
- большой объём данных (также используется язык разметки XML);
- медленная синхронизация модели с БД.

Как упоминалось ранее, предложенная модель должна обладать в том числе следующими характеристиками:

- время получения данных для формирования модели должно быть линейно по отношению к количеству элементов модели;
- объем сериализованных структурированных данных модели должен быть сопоставим с аналогичными моделями, применяемыми в настоящее время.

4.1 Сравнение моделей данных по объёму

Для проведения сравнения различных моделей данных возьмём две базы данных с различными схемами. Обозначим их как M1 и M2. В первой модели данных содержится 20 сущностей и 11 процедур, во второй схеме данных содержится 959 сущностей и 1289 процедур.

Далее необходимо сформировать при помощи стандартных средств сериализованные структуры, отображающие модели данных, и изменить объём занимаемого ими дискового пространства. Так как эта величина не является темпоральной, измерение проводилось один раз.

Для схемы данных M1 объём сериализованного представления составил:

- предлагаемая модель: 52 297 байт;
- DBML: 33 746 байт (-35% от предложенной модели);
- EDM: 146 539 байт (+180% от предложенной модели).

Для схемы данных M2 объём сериализованного представления составил:

- предлагаемая модель: 3 817 494 байт;
- DBML: 2 566 259 байт (-33% от предложенной модели);
- EDM: 8 632 101 байт (+126% от предложенной модели).

Визуально данные сравнения представлены на рисунке.

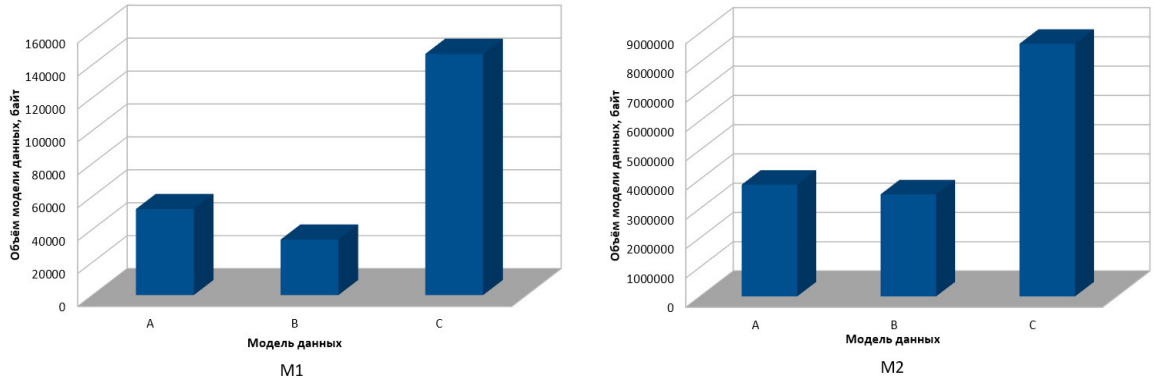

Рис. Сравнение объёма моделей данных схем M1 (31 элемент) и M2 (2248 элементов), где A – предлагаемая модель, B – DBML, C – EDM

4.2 Сравнение моделей по времени формирования

В данном эксперименте сравнивается время синхронизации моделей данных с БД. В качестве исходных реляционных схем данных также были взяты M1 и M2 из предыдущего эксперимента. Так как для DBML отсутствует встроенный механизм синхронизации с БД, проверяться будут только дизайнеры для предлагаемой схемы и EDM.

Было проведено 6 замеров времени синхронизации для каждой модели и схемы данных. В расчёт среднего времени брались все результаты, кроме первого, т. к. он считается подготовительным.

Результаты представлены в таблице.

Таблица

Сравнение времени синхронизации схем БД с моделями данных

Схема данных	Предлагаемая модель		Модель EDM	
	Общее время	На 1 элемент	Общее время	На 1 элемент
M1 (31 элемент)	119 мс	3,8 мс	1 072 мс	34,6 мс
M2 (2248 элементов)	7 420 мс	3,3 мс (-13%)	135 352 мс	60,2 мс (+74%)

Как можно отметить, формирование предлагаемой модели происходит от 10 до 20 раз быстрее, чем стандартными средствам для EDM. Также не была замечена деградация по времени синхронизации с увеличением количества элементов БД.

Заключение

Предложенная модель данных позволяет получать и хранить информацию о структуре реляционной БД. Время заполнения модели превосходит популярные решения для SQL Server: нотации DBML и EDM. Объём дискового пространства, необходимый для записи предложенной модели, сопоставим с аналогичными решениями, применяемыми для доступа к данным программных продуктов.

Описанная в работе модель данных полезна для процессов проектирования и перепроектирования программных продуктов, так как её расширяемость и независимость от физической реализации хранилища данных позволяют унифицировать процессы анализа и разработки.

СПИСОК ЛИТЕРАТУРЫ

1. Guarino N. Formal ontology in information systems // Proceedings of the first international conference (FOIS'98). Trento, Italy. IOS press, 1998. Т. 46. pp. 3–15.

2. Simple Object Access Protocol (SOAP) 1.1. W3C Note 08 May 2000. – URL: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/ (дата обращения: 25.11.2019).

3. Yarushkina N., Moiseev V. Analytical review of data transformation for the task of integrating various representations on the example of ontologies and relational databases // CEUR Workshop Proceedings. 2019. Vol. 2413. pp. 191–197.

4. Bumans G. Mapping between Relational Databases and OWL Ontologies: an Example. – URL: https://www.lu.lv/materiali/apgads/raksti/756_pp_99-117.pdf (дата обращения: 25.11.2019).

5. Astrova A., Kalja A. Mapping of SQL Relational Schemata to OWL Ontologies. – URL: http://wseas.us/e-library/conferences/2006elounda1/papers/537-193.pdf (дата обращения: 25.11.2019).

6. Freitas R. A. P. Relational databases digital preservation : дис. – 2013. – URL: https://repositorium.sdum.uminho.pt/bitstream/1822/25655/1/Ricardo%20Andr%C3%A9%20Pereira%20Freitas.pdf (дата обращения: 25.11.2019).

7. .edmx File Overview (Entity Framework). 2011. – URL: https://docs.microsoft.com/en-US/previous-versions/dotnet/netframework-4.0/cc982042(v=vs.100) (дата обращения: 25.11.2019).

REFERENCES

1. Guarino N. Formal Ontology in Information Systems. *Proceedings of the First International Conference (FOIS'98)*. Trento, Italy, IOS Press, 1998, vol. 46, pp. 3–15.

2. *Simple Object Access Protocol (SOAP) 1.1. W3C Note 08 May 2000.* Available at: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/ (accessed: 25.11.2019).

3. Yarushkina N., Moiseev V. Analytical Review of Data Transformation for the Task of Integrating Various Representations on the Example of Ontologies and Relational Databases. *CEUR Workshop Proceedings*, 2019, vol. 2413, pp. 191–197.

4. Bumans G. *Mapping between Relational Databases and OWL Ontologies: an Example.* Available at: https://www.lu.lv/materiali/apgads/raksti/756_pp_99-117.pdf (accessed: 25.11.2019).

5. Astrova A., Kalja A. *Mapping of SQL Relational Schemata to OWL Ontologies.* Available at: http://wseas.us/e-library/conferences/2006elounda1/papers/537-193.pdf (accessed: 25.11.2019).

6. Freitas R.A.P. *Relational Databases Digital Preservation. Diss.* 2013. Available at: https://repositorium.sdum.uminho.pt/bitstream/1822/25655/1/Ricardo%20Andr%C3%A9%20Pereira%20Freitas.pdf (accessed: 25.11.2019).

7. *.edmx File Overview (Entity Framework).* 2011. Available at: https://docs.microsoft.com/en-US/previous-versions/dotnet/netframework-4.0/cc982042(v=vs.100) (accessed: 25.11.2019).