УДК 004.91
DOI: 10.17586/0021-3454-2016-59-5-355-363

УСОВЕРШЕНСТВОВАНИЕ МОДЕЛИ СИТУАЦИОННО-ОРИЕНТИРОВАННОЙ БАЗЫ ДАННЫХ ДЛЯ ВЗАИМОДЕЙСТВИЯ С MySQL

А. С. Гусаренко

Уфимский государственный авиационный технический университет, 450000, Уфа, Россия
E-mail: artyomgusarenko@gmail.com

Рассматриваются вопросы оснащения модели ситуационно-ориентированной базы данных (СОБД) специализированными средствами для взаимодействия с системой управления баз данных MySQL. Представлена усовершенствованная модель СОБД, расширяющая возможности использования XML-данных, хранимых в реляционной СУБД MySQL. В результате модель СОБД получает средства для соединения с базой данных и управления ее объектами. Взаимодействие заключается в создании баз данных и таблиц, а также в выборке данных и их обновлении. Функции по организации взаимодействия реализованы на платформе интерпретатора динамической модели СОБД, разработанной с использованием технологий PHP. Модель имеет XML-представление и эквивалентное представление в виде диаграмм.

Ключевые слова: *веб-приложение, база данных, динамическая модель, NoSQL, XML, DOM, PHP*

Введение. На современном этапе развития NoSQL баз данных документоориентированные базы работают с различными типами данных, которые основаны на открытых стандартах XML и JSON, поддерживаемых также в реляционных базах данных. Ситуационно-ориентированные базы данных (СОБД), относимые [1—6] к типу документоориентированных, работают с различными так называемыми гетерогенными источниками данных [7—13]. В качестве источника данных могут выступать XML-данные, но хранящиеся не в привычном виде во внешней памяти, а в популярных СУБД, например MySQL. Доступ к таким XML-данным осуществляется опосредованно через принятые в MySQL запросы по стандарту SQL. В настоящее время СОБД не обладают инструментарием для получения доступа к данным через динамическую модель. Получение данных в приложении на основе СОБД осуществляется традиционными средствами с помощью различных функций на высокоуровневом языке программирования.

Предлагается создать модель более высокого уровня абстракции на основе динамической модели СОБД путем обеспечения взаимодействия с MySQL.

Соединение с сервером СУБД. В условиях, когда базы данных имеют различные направления развития и множество успешно реализованных видов систем управления, модель СОБД следует ориентировать на подключение к различным типам баз данных. Таким образом, в модели указывается тип базы данных, например `type = MySQL` в элементе `doc` (рис. 1), что важно, так как лингвистические средства СУБД отличаются. Элемент `doc`, функциональное назначение которого — соединение с базой данных, выделяется в отдельный тип документов — `conndoc`. На doc-соединения этого типа ссылаются другие элементы и документы модели из источников DPO, если требуется подключение к базе данных, в которой хранятся или собираются из одной или нескольких таблиц XML-документы, предварительно снабжаемые тегами XML, интерпретатором или функциями UDF, встроенными в СУБД. В зависимости от типа СУБД интерпретатор [13] задействует требуемый инструментарий для взаимодействия с хостом, на котором установлена СУБД. Для создания соединения с хостом СУБД в модели используется атрибут `action`, в котором декларируется

нужное действие `Connect` — открытие соединения — или `Disconnect` — закрытие соединения (см. рис. 1). Как только соединение инициируется, создается идентификатор соединения в памяти; ссылаясь на него, получать доступ к базе данных из источников можно при любом состоянии модели.

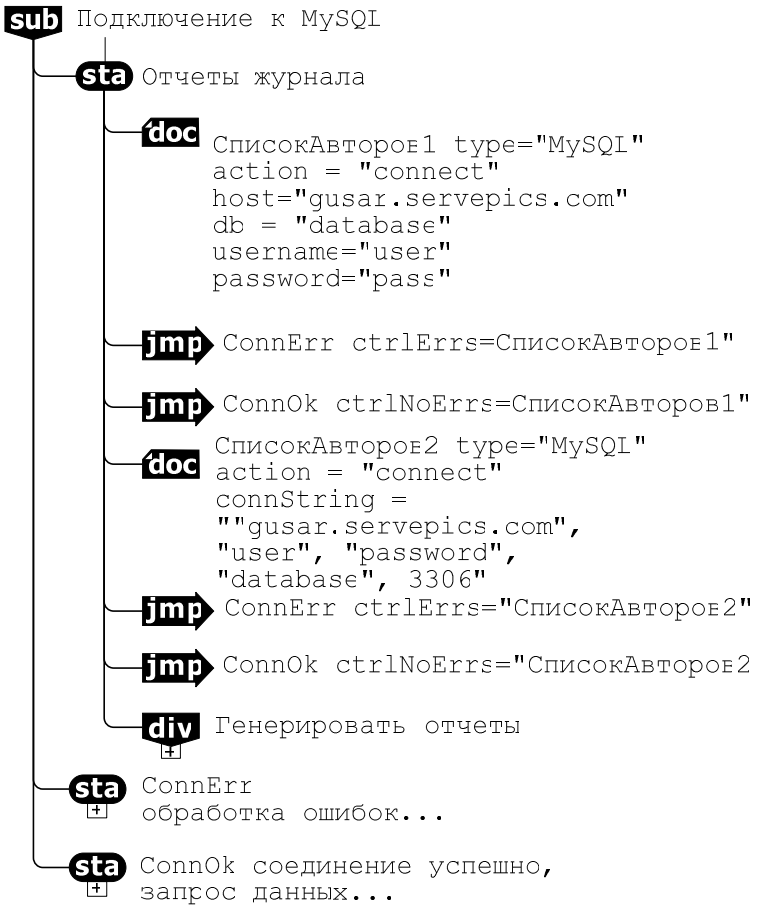

Рис. 1

Выбор хоста и базы данных, атрибуты соединения с базой данных. Перед тем как выбирать базу данных, требуется указать хост, на котором работает СУБД, в модели это делается через директиву `host`, где указывается URL-адрес сервера (в диаграмме на рис. 1 указан адрес сервера MySQL `gusar.servepics.com`). Выбрать базу, из которой приложение получает данные, можно по имени в атрибуте элемента `doc` модели `db` с указанием логина в атрибуте `username` и пароля в атрибуте `password`, принимаемых в интерпретаторе внутри программного кода `mysql_select_db`. Если база данных не создана в модели, ее можно создавать с помощью SQL-запроса, а затем выполнять выборку данных с помощью select-запросов, специфицируемых в источниках DPO [13]. После того как строка задана, интерпретатор собирает ее и использует в инструментарии MySQLi для работы с MySQL. В качестве примера doc-элемента подключения на рис. 1 представлена ситуация `Sta Отчеты журнала`, где в форме параметров-атрибутов `doc` с именем `СписокАвторов1` указаны подробности подключения. Элементы, которые имеют символ „плюс“ в рамке, предполагают свернутое содержимое, т.е. содержат дополнительное динамическое вложение, например DPO, или элементы-функции обработки ошибок. Такое представление характерно для динамических веб-приложений.

Строка соединения `connString`. Если СУБД требует подключения в форме строки, в модели предусматривается атрибут `connString`, в котором параметры подключения указываются не в разнородных самостоятельных атрибутах, а полностью строкой. Это позволяет

принять строку и сразу использовать ее в расширении взаимодействия с СУБД. Примером такой строки подключения служит ситуация `Отчеты журнала`, с doc-элементом соединения `СписокАвторов2`. В данном примере СУБД MySQL обеспечивает соединение и с помощью самостоятельных атрибутов, и с помощью целостной строки.

Обработка ошибок соединения с СУБД. Если соединение не может быть установлено, это приводит к ошибкам, обработку которых следует предусмотреть в модели [14—20]. На диаграмме (см. рис. 1) показаны специальные `jmp`-переходы, в которых задан атрибут `ctrlErrs=СписокАвторов1`, указывающий, что следует обрабатывать ошибки соединения. Элемент `jmp:ConnErr` означает переход к ситуации обработки ошибок `sta:ConnErr`, когда в состоянии `СписокАвторов1` возникает ошибка.

Когда ошибка появляется, сведения о ней записываются интерпретатором СОБД [26—30] в глобальный массив ошибок, где размещаются сведения-значения:

— *код ошибки* — уникальный идентификатор, по которому можно обратиться к ошибке в массиве в случае, когда к ней вернется пользователь;

— *имя ситуации* — название ситуации, в которой возникает ошибка;

— *сообщение об ошибке* — отладочная информация, предназначенная администратору системы для предотвращения ошибки в будущем; эта служебная информация не требуется пользователю, но необходима администратору для устранения проблем в приложении;

— *сообщение об ошибке для пользователя* — сообщение, демонстрируемое на экране, уведомляющее пользователя о действиях, которые требуется выполнить.

В дальнейшем, когда произошла ошибка, ее обработка, после записи сведений о ней в глобальный массив, осуществляется в специальном обработчике ошибок [31—35] `ConnErr`, в этой ситуации обычно описываются действия для интерпретатора по устранению возникшей ошибки. Ошибка обязательно сопровождается сообщением о том, что соединение с СУБД не удалось, и, если требуется, выдаются рекомендации пользователю о его дальнейших действиях. Другой вариант: когда ошибок соединения не обнаружено, срабатывает `jmp`-переход `ConnOk`, соответствующий одноименной ситуации, где выполняются действия, свойственные нормальной работе приложения [36—40]; переход в этом случае помечается атрибутом `ctrlNoErrs`.

Создание, удаление баз данных и таблиц, внесение изменений в метаданные, DDL. Для манипулирования базой данных в СУБД MySQL используется язык SQL, по стандарту которого строятся запросы. Поэтому для того чтобы вносить изменения в базу, оперируя выражениями DDL и DML, требуется модель с заданными операциями по изменению структуры и сведений базы данных [40—42].

В модели СОБД язык манипулирования метаданными обеспечен средствами задания операций в элементах `doc` [29—30]. Создание, удаление, изменение структуры данных активируется декларациями атрибута `action` элемента `doc`. Как показано на рис. 2, в субмодели `Подключение к MySQL` атрибутом `action = ddl` в состояниях `Создать базу журнала`, `Создать таблицу`, `Удалить таблицу`, `Изменить таблицу` задается действие по изменению метаданных базы. В атрибуте `sql` задается конкретный вид действия над метаданными базы в форме SQL-запроса: это может быть запрос `create` для создания новой базы данных или `drop`, если базу данных и все ее сведения следует удалить. Когда возникают задачи по изменению структур данных внутри таблицы или ее отдельного элемента, такого как ключ, используется индекс `action = alter`, что означает внесение изменений в существующую структуру.

На синтаксической диаграмме (см. рис. 2) представлена ситуация, в которой заданы четыре элемента `doc`, содержащие операции на языке DDL для MySQL. Элемент `Создать базу журнала` реализует SQL-запрос [35] создания базы данных, в квадратных скобках указываются необязательные атрибуты. Состояние `Создать таблицу` реализует пример создания

таблицы tbl_name, где в секции create_definition указываются названия колонок, ключи или индексы и их типы. Для удаления таблицы со всем содержимым предусматривается ситуация Удалить таблицу, где используется преимущественно операция DROP с указанием условия IF EXISTS, означающим, что требуется удалить таблицу, в случае если она существует, т.е. это — предотвращение ошибки в запросе средствами СУБД. Параметр удаления Restrict | Cascade указывает, как удалить таблицу, например — выполнить каскадное удаление.

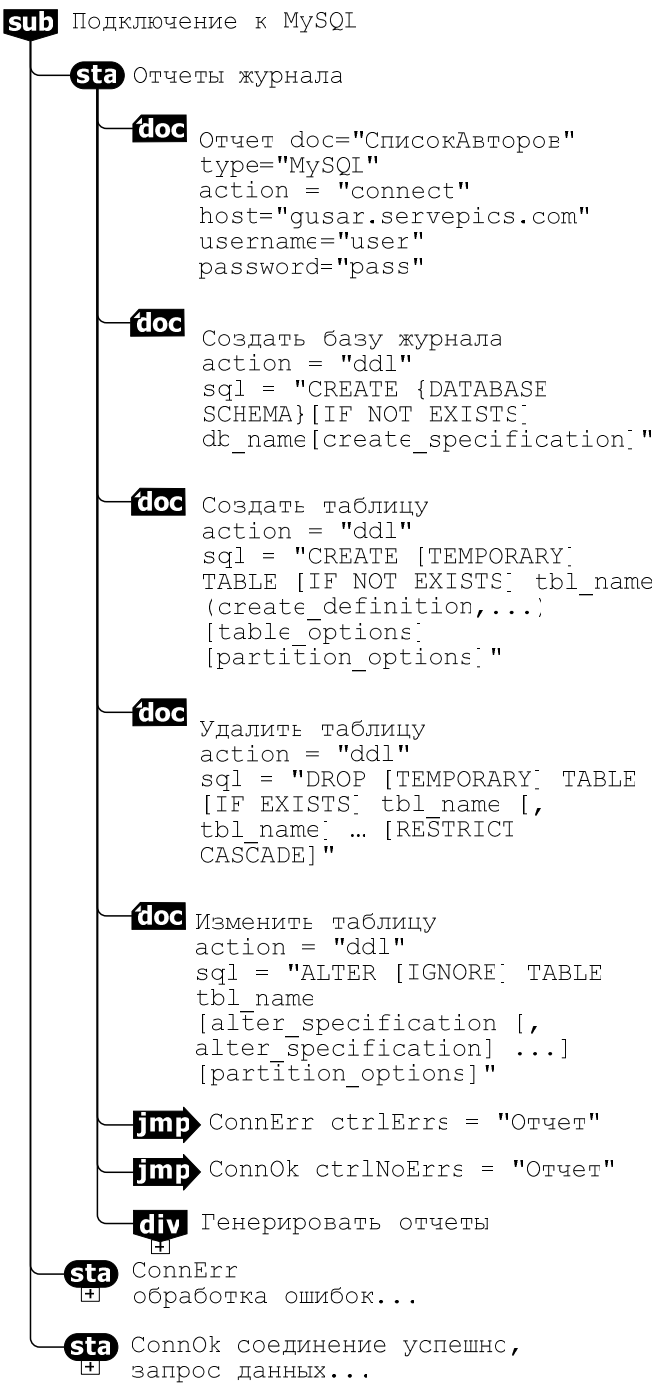

Рис. 2

В состоянии Изменить таблицу операция ALTER задает параметры изменения существующих объектов базы данных. Операции изменения указываются в секции alter_specification, где можно разместить декларации по добавлению колонки с опре-

делением ее места среди остальных колонок. Также можно указывать общие для всех элементов таблицы операции удаления или переименования для ключей, индексов и колонок. Возможно указание типа индекса из двух вариантов для MySQL — это бинарное дерево или кэш (BTREE | HASH).

Создание баз данных и манипулирование данными СУБД MySQL. Обеспечить приложение новыми функциями можно введением в модель дополнительного декларативного описания операций в источнике данных с помощью элементов `doc`. Таким образом, предусматривается как минимум два вида `doc`-элементов, служащих для взаимодействия с MySQL:

— *doc-элемент соединения* требуется в модели для установления связи с СУБД, в нем указываются все подробности подключения; атрибуты собираются интерпретатором в целостную строку подключения `connString`, в дальнейшем используемую в расширении интерпретатора для подключения к конкретной СУБД;

— *doc-элемент модификации данных внутри СУБД* — элемент со спецификациями для оперирования данными внутри СУБД; в элементах данного типа указывается, как собрать документ из полей таблицы или, например, извлечь XML-документ, хранящийся в таблице; из элементов этого типа можно ссылаться на элемент соединения, так называемый `conndoc`.

Примером `doc`-элемента соединения является `Отчет` в ситуации `Отчеты журнала`, а в ситуации `XMLReport` из субмодели `Генерировать отчеты` используется `doc`-элемент модификации данных `Report`, где указана ссылка на элемент соединения с СУБД с помощью атрибута `conndoc = Отчет` (рис. 3). Таким образом, несколько элементов модификации данных внутри СУБД могут использовать одну и ту же спецификацию `doc`-элемента соединения. Поскольку элементы модификации предназначены для запросов на изменение данных [18], то в сложных нетривиальных ситуациях рекомендуется использовать специализированные запросы в атрибуте `sql = CREATE DATABASE [IF NOT EXISTS] db_name [CHARACTER SET charset] [COLLATE collation]`, результатом выполнения которых будет база данных `db_name` (в квадратных скобках запроса указываются необязательные атрибуты).

Если запросы SQL, входящие в группу DDL требуются изредка, то запросы DML очень часто требуются в приложении на этапе его активного использования. Группа запросов на извлечение данных `Select` также может быть использована как серия атрибутов, из которых формируется SQL-запрос источника и приемника сведений на подготовительных этапах обработки данных в DOM-объектах [13]. Данные из СУБД запрашиваются с помощью выражения `select` (см. рис. 3) в элементе-источнике `src` с указанием колонки `x` из таблицы `y` по условию `idRow=z`. В примере синтаксической диаграммы используется расширение MySQL для выполнения SQL-запроса к базе данных. В DOM-объекты загружаются сведения из базы данных, представляющие собой иерархические структуры в форме XML. Строки, в которых содержатся XML-данные [31], обрабатываются наряду с остальными вложенными источниками DOM-объекта.

Часто не все иерархические данные таблицы представлены XML-данными. Чтобы решить эту задачу, используется встроенный в интерпретатор модуль для снабжения тегами всех сведений, содержащихся в строках таблицы. Модуль задействуется в состоянии модели для источника данных, который не имеет XML-представления в базе данных. После обработки результат представляет собой иерархическую структуру [16], снабженную XML-тегами и пригодную для загрузки в DOM-объект. Эта задача может быть решена двумя способами: первый — на стороне базы данных с помощью UDF; второй — модуль интерпретатора на языке программирования PHP [6]. Оба способа используются в модели СОБД для загрузки источников данных, принимаемых в виде XML к обработке внутри DOM-объекта.

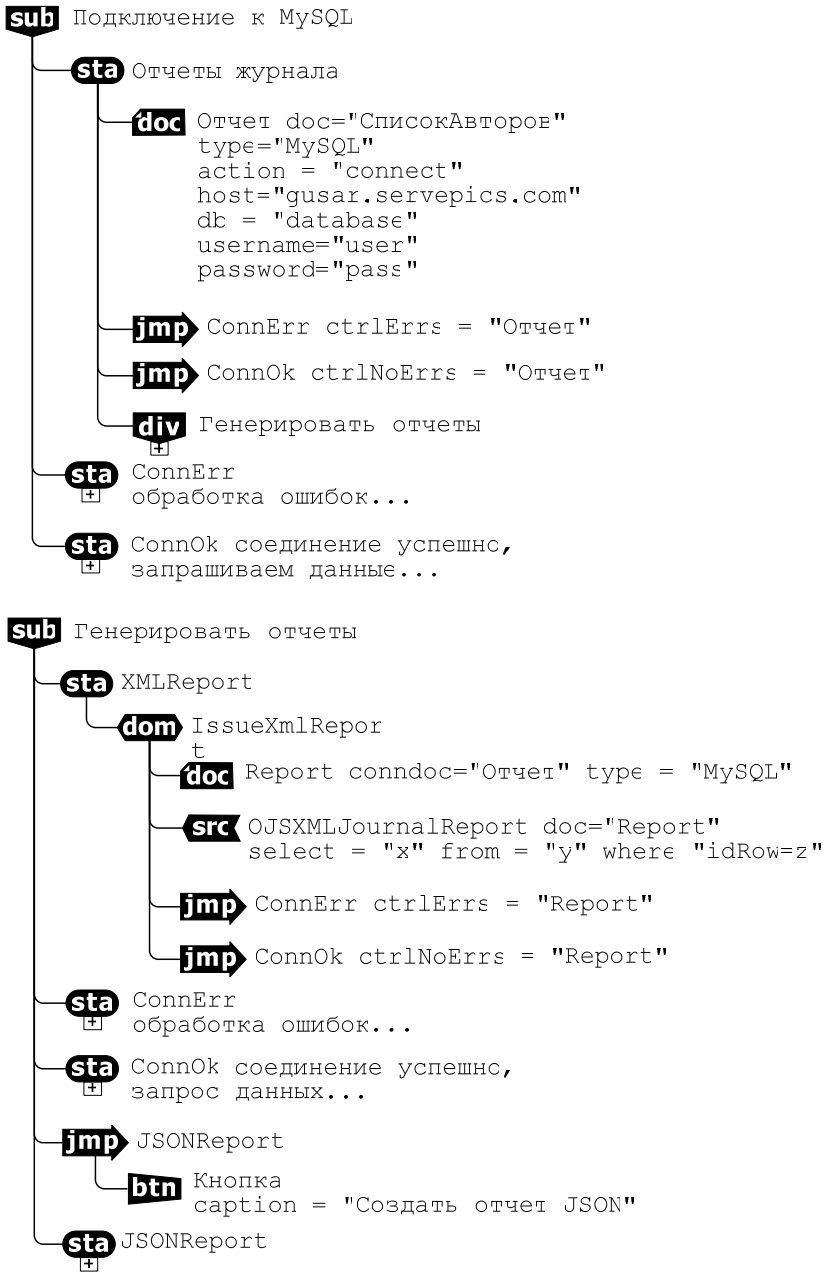

Рис. 3

Вставка и обновление данных. Внесение данных в таблицу и их обновление осуществляется в приемнике данных в виде SQL-запроса `UPDATE [IGNORE] table_references SET col_name1={expr1|DEFAULT}[,col_name2={expr2|DEFAULT}] ... [WHERE where_condition]` к выбранной базе данных. Выражения также задаются в элементе `doc`, их синтаксические конструкции не отличаются от DDL-спецификаций по изменению уже существующих в базе данных объектов, но используются гораздо чаще, чем DDL-операции.

Заключение. Представлены результаты разработки средств СОБД для взаимодействия источников данных в составе динамических DOM-объектов с серверами баз данных MySQL. Разработано обеспечение модели двумя типами doc-элементов для взаимодействия с СУБД MySQL:

1) doc-элемент для соединения с СУБД;

2) doc-элемент для манипулирования данными в СУБД.

Модель оснащена необходимыми спецификациями для выполнения запросов и загрузки их результатов в DPO. Рассмотрены наиболее актуальные варианты модификации баз данных

с помощью действий `action` doc-элементов. Освещены такие разделы языка как DDL — определение метаданных СУБД — и DML — манипулирование данными внутри СУБД.

Статья подготовлена по результатам работы, выполненной при поддержке Российского фонда фундаментальных исследований, грант № 16-07-00239.

СПИСОК ЛИТЕРАТУРЫ

1. *Strauch C., Kriha W.* NoSQL Databases. [Электронный ресурс]: <http://www.christof-strauch.de/nosqldbs. pdf> (дата обращения 07.11.2012).

2. *Фаулер М., Садаладж П. Дж.* NoSQL: новая методология разработки нереляционных баз данных. М.: Вильямс, 2013. 192 с.

3. *Богатырев В. А.* Надежность и эффективность резервированных компьютерных сетей // Информационные технологии. 2006. № 9. С. 25—30.

4. *Богатырев В. А., Богатырев С. В., Богатырев А. В.* Оптимизация древовидной сети с резервированием коммутационных узлов и связей // Телекоммуникации. 2013. № 2. С. 42—48.

5. *Tatarnikova T., Kolbanev M.* Statement of a task corporate information networks interface centers structural synthesis // IEEE EUROCON-2009. St. Petersburg, 2009. P. 1883—1887.

6. *Пуха Г. П.* Современное высокоуровневое и объектно-ориентированное программирование. СПб: СПбГУСЭ, 2013.

7. *Миронов В. В., Шакирова Г. Р.* Концепция динамических XML-документов // Вестн. УГАТУ. 2006. Т. 8, № 5. С. 58—63.

8. *Миронов В. В., Шакирова Г. Р.* Интерпретация XML-документов со встроенной динамической моделью // Вестн. УГАТУ. 2007. Т. 9. № 2. С. 88—97.

9. *Миронов В. В., Шакирова Г. Р.* Программно-инструментальное средство для создания и ведения динамических XML-документов // Вестн. УГАТУ. 2007. Т. 9, № 5. С. 54—63.

10. *Миронов В. В., Юсупова Н. И., Шакирова Г. Р.* Ситуационно-ориентированные базы данных: концепция, архитектура, XML-реализация // Вестн. УГАТУ. 2010. Т. 14, № 2 (37). С. 233—244.

11. *Миронов В. В., Гусаренко А. С.* Ситуационно-ориентированные базы данных: концепция управления XML-данными на основе динамических DOM-объектов // Вестн. УГАТУ. 2012. Т. 16, № 3 (48). С. 159—172.

12. *Миронов В. В., Гусаренко А. С.* Динамические DOM-объекты в ситуационно-ориентированных базах данных: лингвистическое и алгоритмическое обеспечение источников данных // Вестн. УГАТУ. 2012. Т. 16, № 6 (51). С. 167—176.

13. *Гусаренко А. С.* Обработка XML-документов в ситуационно-ориентированных базах данных на основе динамических DOM-объектов: Автореф. дис. … канд. техн. наук. Уфа, 2013. 16 с.

14. *Зоболотский В. П., Оводенко А. А., Степанов А. Г.* Математические модели в управлении: Учеб. пособие. СПб: СПбГУАП, 2001. 196 с.

15. *Валеев С. С., Никитин А. П.* Многоуровневая система фильтрации спама на основе технологий искусственного интеллекта // Вестн. УГАТУ. 2008. Т. 11, № 1 (28). С. 215—219.

16. *Валеев С. С., Уразбахтин Р. Н., Христофоров С. В.* Ситуационное управление процессом грузоперевозок в транспортной компании // Вестн. УГАТУ. 2012. Т. 16, № 6 (51). С. 234—251.

17. *Valeev S. S., Karimov R. R., Karpenko O. Yu., Kondratyeva N. V.* Information support of complex technical systems on the base of soft computing technologies // Vestnik UGATU. 2013. Vol. 17, N 6 (59). P. 57—60.

18. *Валеев С. С.* Информационные технологии BIG DATA в авиации // Proc. of the 2nd Intern. Conf. “Information Technologies for Intelligent Decision Making Support” and the Intended Intern. Workshop “Robots and Robotic Systems”. Ufa: USATU, 2014. P. 150—152.

19. *Канашин В. В., Миронов В. В.* Иерархические виджеты: организация интерфейса пользователя в веб-приложениях на основе ситуационно-ориентированных баз данных // Вестн. УГАТУ. 2013. Т. 17, № 2 (55). С. 138—149.

20. *Канашин В. В., Миронов В. В.* Иерархические виджеты: ввод и контроль данных пользователя в веб-приложениях на основе ситуационно-ориентированных баз данных // Вестн. УГАТУ. 2013. Т. 17, № 5 (58). С. 166—176.

21. *Канашин В. В., Миронов В. В.* Иерархические виджеты: алгоритмы контроля данных пользователя в веб-приложениях на основе ситуационно-ориентированных баз данных // Вестн. УГАТУ. 2014. Т. 18, № 1 (62). С. 204—213.

22. *Канашин В. В., Миронов В. В.* Иерархические виджеты: опыт применения в веб-приложении на основе ситуационно-ориентированной базы данных // Вестн. УГАТУ. 2014. Т. 18, № 2 (63). С. 185—196.

23. *Макарова Е. С., Миронов В. В.* Проектирование концептуальной модели данных для задач Web-OLAP на основе ситуационно-ориентированной базы данных // Вестн. УГАТУ. 2012. Т. 16, № 6 (51). С. 177—188.

24. *Макарова Е. С., Миронов В. В.* Функции аналитики в веб-приложениях на основе ситуационно-ориентированных баз данных // Вестн. УГАТУ. 2013. Т. 17, № 5 (58). С. 150—165.

25. *Гусаренко А. С., Миронов В. В.* Smarty-объекты: вариант использования гетерогенных источников в ситуационно-ориентированных базах данных // Вестн. УГАТУ. 2014. Т. 18, № 3 (63). С. 242—252.

26. *Гусаренко А. С., Миронов В. В.* Использование RESTful-сервисов в ситуационно-ориентированных базах данных // Вестн. УГАТУ. 2015. Т. 19, № 1 (67). С. 204—211.

27. *Миронов В. В., Юсупова Н. И., Шакирова Г. Р.* Иерархические модели данных: концепции и реализация на основе XML / Под ред. *Н. И. Юсуповой*. М.: Машиностроение, 2011. 453 с.

28. *Миронов В. В., Гусаренко А. С., Диметриев Р. Р., Сарваров М. Р.* Создание персонализированных документов на основе ситуационно-ориентированной базы данных // Вестн. УГАТУ. 2014. Т. 18, № 4 (65). С. 191—197.

29. *Миронов В. В., Юсупова Н. И., Гусаренко А. С.* Ситуационно-ориентированные базы данных: современное состояние и перспективы исследования // Вестн. УГАТУ. 2015. Т. 19, № 2 (68). С. 188—199.

30. *Гусаренко А. С.* Модели создания документов в формате Office Open XML на основе ситуационно-ориентированной базы данных // Прикладная информатика. 2015. Т. 10, № 3. С. 62—75.

31. *He W., Zhai J.* Application of the indent conversion based on XML and DOM // Proc. of Intern. Conf. on Computational and Information Sciences (ICCIS' 2013). 2013. Vol. 1. P. 411—413.

32. *Benzaken V.* et al. Static and dynamic semantics of NoSQL languages // Proc. of the 40th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL' 2013), Rome, Italy. 2013. P. 101—113.

33. *Sladic G., Milosavljevic B., Konjovic Z., Vidakovic M.* Access control framework for xml document collections // Computer Science and Information Systems. 2011. Vol. 8, N 3. P. 591—609.

34. *Sudarsan R., Gray J.* Metamodel search: Using XPath to search domain-specific models // J. of Research and Practice in Information Technology. 2006. Vol. 38, N 4. P. 337—351.

35. *Tang N., Yu J. X., Wong K. F., Li J. X.* Fast XML structural join algorithms by partitioning // J. of Research and Practice in Information Technology. 2008. Vol. 40, N 1. P. 33—53.

36. *Dekeyser S., Hidders J., Paredaens J.* A transaction model for XML databases // World Wide Web-Internet and Web Information Systems. 2004. Vol. 7, N 1. P. 29—57.

37. *Jea K. F., Chang T. P., Chen S. Y.* A semantic-based protocol for concurrency control in DOM database systems // J. of Information Science and Engineering. 2009. Vol. 25, N 5. P. 1617—1639.

38. *Kudrass T., Conrad M.* Management of XML documents in object-relational databases // XML-Based Data Management and Multimedia Engineering. 2002. Vol. 2490. P. 210—227.

39. *Nassis V., Dillon T. S., Rajagopalapillai R., Rahayu W.* An XML document warehouse model // Proc. of Database Systems for Advanced Applications. 2006. Vol. 3882. P. 513—529.

40. *Batory D.* Multilevel models in model-driven engineering, product lines, and metaprogramming // IBM Systems Journal. 2006. Vol. 45, N 3. P. 527—539.

41. *Dejanovic I., Milosavljevic G., Perisic B., Tumbas M. A.* Domain-specific language for defining static structure of database applications // Computer Science and Information Systems. 2010. Vol. 7, N 3. P. 409—440.

42. *Su-Cheng H., Lee C. S.* Efficient preprocesses for fast storage and query retrieval in native XML database // IETE Techn. Rev. 2009. Vol. 26, N 1. P. 28—40.

Сведения об авторе

Артем Сергеевич Гусаренко — канд. техн. наук; УГАТУ, кафедра автоматизированных систем управления; E-mail: artyomgusarenko@gmail.com

Рекомендована кафедрой автоматизированных систем управления

Поступила в редакцию 09.02.16 г.

Ссылка для цитирования: *Гусаренко А. С.* Усовершенствование модели ситуационно-ориентированной базы данных для взаимодействия с MySQL // Изв. вузов. Приборостроение. 2016. Т. 59, № 5. С. 355—363.

IMPROVEMENT OF SITUATION-ORIENTED DATABASE MODEL FOR INTERACTION WITH MySQL

A. S. Gusarenko

Ufa State Aviation Technical University, 450000, Ufa, Russia
E-mail: artyomgusarenko@gmail.com

Completing of situation-oriented database model with specialized means for interaction with database management system MySQL is considered. An improved model of situation-oriented database (SODB) is proposed; the model extends the capabilities of the use of XML-data stored in a relational database MySQL. As a result, the proposed model is provided with tools for the database connection and management of its facilities. The interaction consists in creation of databases, tables, entering sample data and updating data; the functions are realized on the platform of SODB dynamic model interpreter developed with the use of PHP technologies. The model has an XML-based representation and equivalent representation in the form of diagrams.

Keywords: web-application, situation-oriented database, dynamic model, NoSQL, XML, DOM, PHP

Data on author

Artem S. Gusarenko — PhD, Ufa State Aviation Technical University, Department of Automated Control Systems; E-mail: artyomgusarenko@gmail.com

For citation: *Gusarenko A. S.* Improvement of situation-oriented database model for interaction with MySQL // Izv. vuzov. Priborostroenie. 2016. Vol. 59, N 5. P. 355—363 (in Russian).

DOI: 10.17586/0021-3454-2016-59-5-355-363