ISSN 1992-6502 (Print) Вестник УГАТУ ISSN 2225-2789 (Online)
2015. Т. 19, № 2 (68). С. 188–199 http://journal.ugatu.ac.ru

УДК 004.65

СИТУАЦИОННО-ОРИЕНТИРОВАННЫЕ БАЗЫ ДАННЫХ: СОВРЕМЕННОЕ СОСТОЯНИЕ И ПЕРСПЕКТИВЫ ИССЛЕДОВАНИЯ

В. В. Миронов[1], Н. И. Юсупова[2], А. С. Гусаренко[3]

[1] mironov@list.ru, [2] yussupova@ugatu.ac.ru, [3] artyomgusarenko@gmail.com

ФГБОУ ВПО «Уфимский государственный авиационный технический университет» (УГАТУ)

Поступила в редакцию 2 февраля 2015 г.

Аннотация. На концептуальном уровне обсуждается современное состояние исследований в области ситуационно-ориентированных баз данных (СОБД). СОБД рассматривается в рамках общей задачи построения приложения обработки данных как динамической системы. Отмечается, что наличие в составе СОБД встроенной динамической модели позволяет решать указанную задачу на более высоком уровне абстракции. Рассматривается иерархическая ситуационная модель HSM, положенная в основу СОБД в качестве метамодели для динамических моделей конкретных приложений. Обсуждаются средства HSM, предназначенные для задания обработки документов на основе концепции объектов обработки данных. Предлагается концепция виртуальных документов, дающая возможность обработки реальных документов, заданных в различных форматах и использующих различную среду хранения.

Ключевые слова: веб-приложение; база данных; динамическая модель; NoSQL; HSM; XML; JSON; DOM; Smarty; PHP.

ВВЕДЕНИЕ

В настоящее время в рамках движения NoSQL [1], ставящего целью отход от традиционных реляционных СУБД, активно развиваются документо-ориентированные базы данных и основанные на них веб-приложения, предоставляющие клиентам сервисы доступа к удаленным ресурсам.

К этому направлению могут быть отнесены и ситуационно-ориентированные базы данных (СОБД), концепция которых (как и сам термин «СОБД») была предложена в 2010 г. (В. В. Миронов, Н. И. Юсупова, Г. Р. Шакирова [2, 3]). За прошедшее время при поддержке РФФИ проведены исследования различных аспектов построения СОБД:

- применение СОБД при построении веб-приложений [4–6];
- организация обработки XML-документов в СОБД на основе динамических DOM-объектов [7–9];
- организация иерархического пользовательского интерфейса СОБД [10–13];
- организация OLAP-ориентированного интерфейса СОБД [14, 15];
- организация обработки в СОБД документов в формате JSON, а также документов, предоставляемых веб-сервисами [16, 17].

В результате этого первоначальная концепция, терминология, а также общее понимание назначения СОБД подверглись корректировке. В этой связи в данной статье дается попытка единообразного изложения центральных положений, лежащих в основе СОБД, как нового подхода к построению приложений обработки данных.

1. ЗАДАЧА ПОСТРОЕНИЯ ПРИЛОЖЕНИЯ ОБРАБОТКИ ДАННЫХ КАК ДИНАМИЧЕСКОЙ СИСТЕМЫ

СОБД рассматривается нами как компонент приложения обработки данных (ПОД), обеспечивающий доступ к данным (документам), необходимым для реализации некоторого бизнес-процесса, в контексте текущей ситуации (текущего состояния бизнес-процесса). Рассмотрим, как эта задача решается традиционно и что предлагается при использовании СОБД.

Необходимо учитывать, что ПОД в общем случае – это дискретная динамическая система «вход–состояние–выход» (рис. 1).

Работа поддержана грантами РФФИ 13-07-00011 и 15-07-01565.

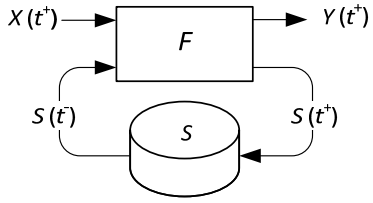

Рис. 1. Модель динамической системы «вход–состояние–выход»

Здесь F – отображение «вход–выход»; S – хранилище выходных (текущих) состояний; $X(t^+)$ – входное воздействие для текущего момента t^+; $S(t^-)$ – входное состояние; $Y(t^+)$ – выходное воздействие для текущего момента t^+; $S(t^+)$ – выходное состояние для текущего момента t^+. Для хранилища S задано начальное состояние S_0. Для начального момента времени t_0 в качестве входного состояния используется начальное состояние $S(t_0^-) = S_0$. Для последующих моментов входное состояние $S(t^-)$ равно выходному состоянию для предыдущего момента t^-.

ПОД как динамическая система. В плане ПОД нас интересует динамическая система с дискретным временем и конечным числом состояний (FSM – Finite State Machine – конечный автомат). ПОД хорошо укладываются в FSM-модель, если в качестве входных воздействий рассматривать поступающие в ПОД извне запросы на обработку данных, в качестве выходных – возвращаемые из ПОД вовне результаты обработки, а в качестве состояния – содержимое хранилища данных.

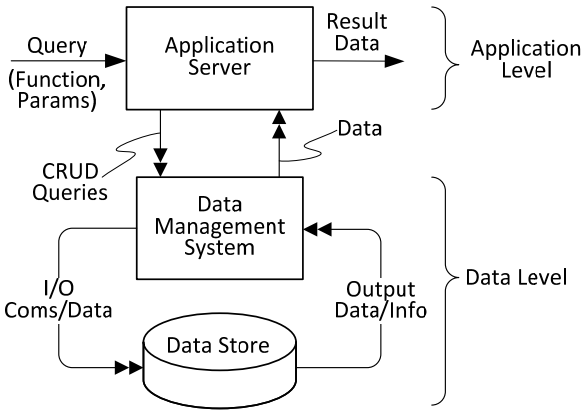

Рис. 2. Двухуровневая архитектура приложения обработки данных

На рис. 2 представлена укрупненная модель ПОД с двухуровневой архитектурой, выполняющего функции сервера приложений в соответствии с широко распространенной технологией «клиент–сервер». Верхний уровень – уровень приложения (Application Level) – включает сервер приложений (Application Server), а нижний – уровень данных (Data Level) – систему управления данными (Data Management System) и хранилище данных (Data Store). Технология «клиент–сервер» предполагает, что приложение-сервер исполняет запросы, которые поступают к нему от приложений-клиентов. В данном случае запрос Query на входе сервера задает функцию Function, которую необходимо выполнить, возможно, с определенными параметрами Params. Возвращаемым результатом может быть выборка данных Result Data.

Для выполнения функций, связанных с обработкой данных, сервер приложений обращается к уровню данных с помощью запросов CRUD Queries[1]. Система управления данными преобразует эти запросы в команды ввода-вывода I/O Coms/Data хранилищу данных. Результирующие выходные данные Output Data передаются серверу приложений для формирования Result Data.

Ситуационная модель. Функционирование сервера приложений должно подчиняться так называемой логике приложения, или бизнес-логике, отражающей правила и ограничения соответствующего бизнес-процесса. Во многих практически важных случаях бизнес-логику удобно задать в виде ситуационной модели, т. е. в виде набора ситуаций (состояний) бизнес-процесса и возможных переходов между ними. В результате получается модель в виде графа переходов, дуги которого нагружены предикатами, задающими правила переходов.

При реализации приложения разработчик должен воплотить бизнес-логику в виде компьютерных программ, обеспечивающих распознавание текущего состояния (ситуации) бизнес-процесса, действий, которые должны быть выполнены в текущем состоянии, возможных переходов текущего состояния.

Уровень данных представляет информационное обеспечение уровня приложения, т. е. обслуживание информационных потребностей бизнес-процесса в тех или иных ситуациях. Различные модели: реляционные базы данных, электронные документы и др.

На рис. 3 представлен пример двухуровневой модели бизнес-процесса. Модель соответствует типичному многопользовательскому веб-приложению, которое обслуживает как анонимных, так и зарегистрированных пользователей. Анонимным пользователям предоставляется

[1] CRUD – Create, Retrieve, Update, Delete – создать, извлечь, обновить, удалить – набор стандартных операций манипулирования данными.

доступ к общим (Public) данным, а зарегистрированным – дополнительно к специальным (Private) данным.

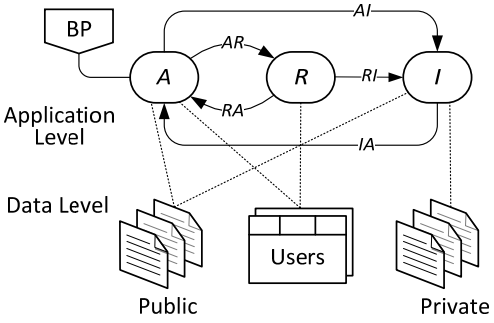

Рис. 3. Пример двухуровневой модели бизнес-процесса

Логика приложения (Application Level) представлена моделью в виде графа переходов, в котором вершины соответствуют возможным состояниям (ситуациям), а дуги – переходам состояний. Начальное состояние задано с помощью символа HSM. Дуги помечены предикатами активности, определяющими переход текущего состояния. Модель содержит три состояния (укрупненных):

- *A* (Anonymous) – работа с анонимным пользователем (начальное состояние – любой пользователь, начинающий работу с приложением, рассматривается как анонимный);
- *I* (Identified) – работа с идентифицированным зарегистрированным пользователем;
- *R* (Registration) – регистрация пользователя.

Предикаты активности модели имеют следующий смысл: *AI* – анонимный пользователь успешно прошел процедуру аутентификации; *AR* – анонимный пользователь пожелал зарегистрироваться; *RI* – регистрируемый пользователь успешно прошел процедуру регистрации; *RA* – регистрируемый пользователь не прошел процедуру регистрации; *IA* – идентифицированный пользователь завершил сеанс.

Уровень данных (Data Level) представлен тремя типами данных: электронными документами общего (Public) и специального (Private) доступа, а также таблицами сведений о зарегистрированных пользователях (Users). Документы Public доступны в состояниях *A* и *I*, документы Private – только в состоянии *I*, а таблица Users – состояниях *A* и *R*.

Реализация моделей. Разработчик приложения должен выполнить программную реализацию логики приложения, обеспечивающую:

- определение текущего состояния (либо оставшегося от предыдущего запроса, либо начальное);
- обнаружение активных переходов, исходящих из текущего состояния, и выполнение смены текущего состояния;
- выполнение действий, предусмотренных в текущем состоянии, в том числе тех, которые связаны с доступом к данным и их обработкой;
- сохранение измененного текущего состояние для обработки следующего запроса.

Реализация доступа к данным (Data Level) в типовых приложениях традиционно основана на использовании реляционных баз данных. На рис. 4 иллюстрируется архитектура уровня данных приложения в виде так называемого сервера баз данных, обслуживающего запросы обработки данных.

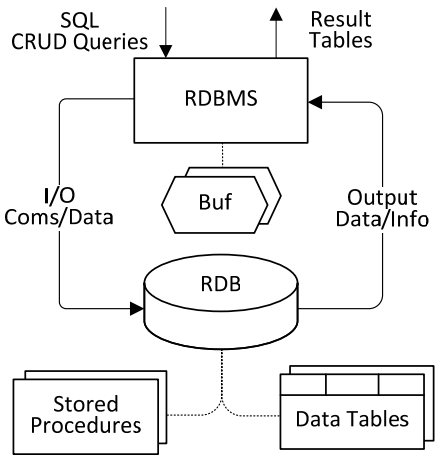

Рис. 4. Архитектура сервера реляционной базы данных

Здесь RDBMS – реляционная СУБД (Relational Database Management System); Buf – буферные объекты, создаваемые RDBMS на время обработки запросов; RDB – реляционная база данных; Data Tables – таблицы данных, являющиеся основными объектами реляционной базы данных; Stored Procedures – хранимые процедуры, также являющиеся объектами базы данных; другие типы объектов не показаны.

С уровня приложения в RDBMS поступают CRUD-запросы на языке SQL (SQL CRUD Queries). Обрабатывая эти запросы, RDBMS формирует команды ввода-вывода данных (I/O Coms / Data) к таблицам и другим объектам базы данных, используя буферные объекты для промежуточного хранения данных. В случае запросов выборки RDBMS формирует в буферах результирующие таблицы (Result Tables) в том виде, в каком они требуются на уровне приложения.

Хранимые процедуры (Stored Procedures) – это программы обработки данных, хранящиеся в базе данных (а не на уровне приложения). Они могут запускаться на выполнение в среде сервера баз данных по специальным запросам к RDBMS. Хранимые процедуры, таким образом, позволяют реализовать на уровне данных фрагменты логики вышестоящего уровня.

2. СОБД И ПРИЛОЖЕНИЯ ОБРАБОТКИ ДАННЫХ

СОБД изначально рассматривалась как новый, более эффективный подход к построению приложений обработки данных. Рассмотрим, как решается эта известная задача на более высоком уровне абстракции – на основе СОБД.

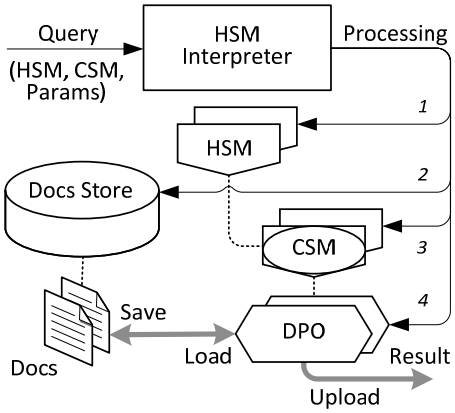

Рис. 5. Архитектура СОБД

Архитектура СОБД. На рис. 5 представлена архитектурная модель СОБД. Она включает следующие компоненты:

● Хранилище документов Docs Store, содержащее набор электронных документов Docs.

● Набор, включающий ситуационные иерархические модели HSM (Hierarchical Situation Model, Hierarchical State Model – иерархическая ситуационная модель, или иерархическая модель состояний). Каждая HSM является моделью некоторого бизнес-процесса, задавая для него набор состояний и переходов.

● Набор, включающий модели текущего состояния CSM (Current State Model). Каждая CSM соответствует одной (родительской) HSM, в то время как одной HSM может соответствовать несколько (или ноль) (дочерних) CSM. Таким образом, CSM соответствует некоторой независимой реализации (воплощению) HSM и задает текущее состояние этой реализации.

● Набор, включающий буферные объекты обработки данных DPO (Data Processing Object), предназначенные для программной обработки документов. Каждый DPO соответствует одной (родительской) CSM, т. е. одной реализации HSM. Для одной CSM возможно несколько буферных объектов DPO в зависимости от текущего состояния HSM.

● Интерпретатор ситуационной модели – HSM Interpreter – играет роль системы управления базой данных.

Функционирование СОБД. На вход интерпретатора ситуационной модели поступают запросы Query, задающие конкретные HSM, CSM и Params – иерархическую модель, ее текущее состояние и параметры обработки. Получив запрос, интерпретатор выполняет цикл обработки (цикл интерпретации): обрабатывает заданную HSM (*1*), начиная с текущих состояний, соответствующих заданной CSM (*3*) (или с начальных состояний, если CSM не задана). В ходе обработки интерпретатор контролирует переходы текущих состояний в соответствии со спецификациями HSM и при необходимости корректирует их в CSM (*3*). Кроме того, интерпретатор управляет хранилищем данных (*2*), а также создает на время цикла интерпретации буферные объекты DPO и управляет ими (*4*) (тоже в соответствии со спецификациями HSM), например:

● загружает в DPO определенные документы из хранилища (Load);

● обрабатывает документы в DPO;

● сохраняет измененные документы в хранилище (Save);

● выгружает документы как результат запроса (Upload).

Таким образом, результат цикла интерпретации может быть трояким:

● изменение в CSM текущего состояния обработанной HSM;

● изменение содержимого хранилища документов;

● выдача результирующего документа.

Таким образом, использование СОБД – это реализация «подхода на основе моделей» (Model Driven Approach), эффективность которого обусловлена использованием высокоабстрактных моделей, облегчающих процесс проектирования программного обеспечения.

Более высокий уровень абстракции при использовании СОБД достигается за счет того, что в HSM логика контроля текущих ситуаций и соответствующей обработки данных реализована в декларативном виде (а не в процедурном, как это имеет место при использовании скриптов или хранимых процедур). Разработчик составляет ситуационную модель, при этом рутина управления приложением ложится на интерпретатор.

3. ИЕРАРХИЧЕСКАЯ СИТУАЦИОННАЯ МОДЕЛЬ HSM

HSM является центральным компонентом СОБД. Модель представляет собой упорядоченное множество элементов, образующих иерархию (дерево) в том смысле, что модель имеет единственный корневой элемент, а каждый некорневой элемент имеет один родительский элемент и (возможно) несколько дочерних. Для каждого родительского элемента задан порядок следования дочерних элементов.

Каждый элемент включает три компоненты:

- тип (обязательный компонент);
- имя (обязательный компонент);
- набор атрибутов (необязательный компонент).

Формы записи HSM. Используются две эквивалентные формы (нотации) представления модели: графическая, предназначенная для разработчиков, и текстовая, предназначенная для обработки интерпретатором.

В графической форме тип элемента записывается в значке элемента, а имя и атрибуты – справа от значка элемента. Иерархия элементов образуется с помощью соединительных линий, подчиняющихся следующим правилам:

- линии, прикрепленные снизу или справа от исходного элемента, ведут к его дочерним элементам;
- линии, прикрепленные слева или сверху от исходного элемента, ведут к его родительскому элементу.

В текстовой форме для записи модели применяется XML-образный синтаксис. Тип элемента записывается в виде префикса пространства имен, а имя и атрибуты – по правилам записи имен и атрибутов XML-элементов. Иерархия образуется путем вложения элементов в соответствии с правилами XML.

Основные элементы HSM следующие:

sta – состояние (state) – дочерний элемент HSM (корневое состояние) или субмодели;

sub – субмодель (submodel) – дочерний элемент некоторого состояния. Используется для задания множества внутренних состояний, одно из которых в каждый момент является текущим;

act – акция (action) – дочерний элемент состояния. Используется для задания определенных действий, когда родительское состояние является текущим;

jmp – переход (jump) – дочерний элемент некоторого состояния. Используется для задания смены текущего состояния, для чего в нем предусмотрен предикат активности.

На рис. 6 приведен пример модели в графической форме. Эквивалентное представление в текстовой форме приведено в листинге 1. HSM соответствует модели бизнес-процесса, приведенной на рис. 3.

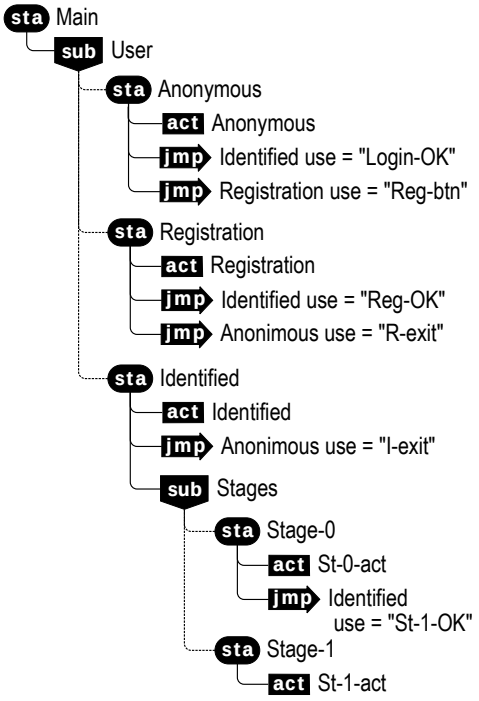

Рис. 6. Пример HSM в графической форме

Листинг 1

Пример HSM в текстовой форме

```
001<sta:Main>
002    <sub:User>
003        <sta:Anonimous>
004            <act:Anonimous/>
005            <jmp:Identified use = "Login-OK"/>
006            <jmp:Registration use = "Reg-btn"/>
007        </sta:Anonimous>
008        <sta:Registration>
009            <act:Registration/>
010            <jmp:Identified use = "Reg-OK"/>
011            <jmp:Anonimous use = "R-exit"/>
012        </sta:Registration>
013        <sta:Identified>
014            <act:Identified/>
015            <jmp: Anonimous use = "I-exit"/>
016            <sub:Stages>
017                <sta:Stage-0>
004                    <act:St-0-act/>
005                    <jmp:Stage-1 use = "St-1-OK"/>
007                </sta:Stage-0>
008                <sta:Stage-1>
009                    <act:St-0-act/>
012                </sta:Stage-1>
006            </sub:Stages>
007        </sta: Identified>
008    </sub:User>
016</sta:Main>
```

Корневое состояние sta:Main модели содержит субмодель sub:User, реализующую бизнес-логику управления пользователями. Эта субмодель включает три состояния:

● sta:Anonymous – начальное состояние субмодели, соответствующее анонимному пользователю (в начале сеанса каждый пользователь считается анонимным);

● sta:Registration – состояние регистрации пользователя в системе;

● sta:Identified – состояние, соответствующее идентифицированному (распознанному) пользователю.

В каждом из состояний субмодели sub:User предусмотрены акции, которые формируют результат пришедшего запроса:

● act:Anonymous – отсылает анонимному пользователю картинку, сформированную на основе документов Public (см. рис. 3), а также форму для ввода идентификационных данных или намерения зарегистрироваться;

● act:Registration – отсылает регистрируемому пользователю экранную форму для ввода анкетных данных или отказа от регистрации;

● act:Identified – отсылает идентифицированному пользователю картинку, сформированную на основе документов Private (см. рис. 3), а также элемент управления (кнопку) для возврата в режим анонимного пользователя.

Смена состояний обеспечивается элементами перехода, ссылающимися на функции-предикаты, которые проверяют введенные пользователем данные, пришедшие в параметрах запроса:

● jmp:Identified в состоянии sta:Anonymous обеспечивает переход в состояние sta:Identified, если пользователь ввел правильные логин и пароль, что проверяет функция Login-OK;

● jmp:Registration обеспечивает переход в состояние sta:Registration, если пользователь нажал кнопку регистрации, что проверяет функция Reg-btn;

● jmp:Identified в состоянии sta: Registration обеспечивает переход в состояние sta:Identified, если пользователь корректно заполнил регистрационную анкету, функция Reg-OK;

● jmp:Anonymous в состоянии sta:Registration и в состоянии sta:Identified обеспечивают возврат в состояние sta:Anonymous, если пользователь нажал кнопку выхода, функции R-exit и I-exit.

В состоянии sta:Identified предусмотрена также субмодель sub:Stages, которая задает для примера два внутренних подсостояния – sta:Stages-0 и sta:Stages-1.

Таким образом, HSM определяет в декларативной форме иерархию состояний и переходов между состояниями, а также действий, ассоциированных с состояниями.

4. ИНТЕРПРЕТАЦИЯ HSM

Рассмотрим, как интерпретатор выполняет обработку ситуационной модели в ходе цикла интерпретации по обработке запроса и формирования результата.

Многопроходовая интерпретация. Обработка HSM интерпретатором HSMI выполняется путем нисходящего обхода элементов в соответствии с их иерархическим порядком. Обход начинается с корневой ситуации, обработка ситуации включает обработку ее дочерних элементов и т. д. При обработке субмоделей обрабатываются лишь текущие элементы-состояния в соответствии с CSM. Применяется многопроходовая интерпретация – интерпретатор последовательно выполняет несколько проходов HSM в рамках одного цикла интерпретации. Обычно требуется два прохода (начальный и завершающий), в особых случаях выполняются дополнительные (промежуточные) проходы.

Начальный проход предназначен для создания (на первом цикле интерпретации) или обновления (на последующих циклах интерпретации) модели текущего состояния CSM. При создании CSM в качестве текущих состояний субмоделей берутся их начальные состояния. Обновление текущих состояний выполняется в соответствии с активностью элементов-переходов – при обнаружении активного перехода (с истинным предикатом активности) в CSM для обрабатываемой субмодели фиксируется новое текущее состояние.

Завершающий проход предназначен для формирования результата запроса в соответствии с достигнутыми состояниями. На этом проходе элементы-переходы игнорируются, а в субмоделях обрабатываются только те элементы-состояния, которые соответствуют текущим состояниям CSM.

Смена текущих состояний. Как следует из вышеизложенного, смена состояний выполняется только на первом проходе при обработке элемента-перехода, принадлежащего некоторому элементу-состоянию. Когда при обработке элемента-перехода интерпретатор выявляет его активность, то он выполняет следующее:

● обрабатывает родительский элемент-состояние в особом, эпилоговом режиме (в этом режиме элементы-переходы игнорируются);

● находит целевой элемент-состояние активного перехода и фиксирует его в CSM в качестве текущего состояния данной субмодели;

● обрабатывает новый текущий элемент-состояние в особом, прологовом режиме.

Предусмотрена возможность блокировать обработку HSM-элементов в зависимости от прохода и режима обработки с помощью атрибутов pass и mode соответственно.

Модель текущего состояния. CSM представляет собой поддерево HSM, в котором для субмоделей указаны их текущие состояния. На рис. 7 представлен пример CSM, соответствующей HSM, рассмотренной выше (см. рис. 6). CSM соответствует текущему состоянию sta:Identified для субмодели sub:User и текущему состоянию sta:Stage-1 для субмодели sub:Stages.

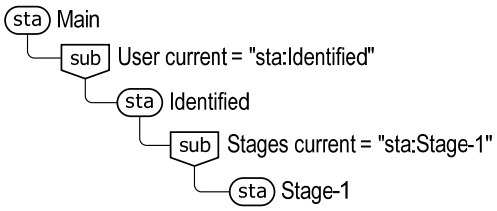

Рис. 7. Пример CSM в графической форме

5. ОРГАНИЗАЦИЯ ОБРАБОТКИ ДОКУМЕНТОВ В СОБД

Для обработки документов из хранилища документов СОБД предусмотрены специальные варианты элемента-акции, названные элементами обработки документов, или DPO-элементами. Элементы этого типа позволяют специфицировать обработку документов в HSM явно, не скрывая этот процесс внутри акций. В настоящее время проработаны два типа элементов DPO:

● dom-элементы, ориентированные на обработку XML-документов с применением технологии XSL-трансформации [7–9];

● smarty-элементы, ориентированные на обработку JSON-документов с применением технологии заполнения шаблонов [16].

Поясним общую концепцию DPO на примере dom-элементов. Элементы этого типа размещаются внутри элементов-состояний и специфицируют DOM-объекты, которые создаются и используются для загрузки, модификации и выгрузки XML-контента в течение цикла интерпретации HSM. Данные функции поддерживаются с помощью трех типов элементов HSM:

dom – dom-элемент – дочерний элемент элемента-состояния. Используется для создания DOM-объекта, который по умолчанию существует до окончания цикла интерпретации;

src – элемент-источник (source) – дочерний элемент dom-элемента или другого источника, ссылающийся на внешний XML-документ. Используется для загрузки XML-контента в DOM-объект, созданный родительским dom-элементом. Вложенные источники позволяют формировать содержимое DOM-объекта из нескольких XML-документов;

rcv – элемент-приемник (reciver) – дочерний элемент dom-элемента. Используется для выгрузки данных из DOM-объекта, созданного родительским dom-элементом, в хранилище документов или как результат запроса.

На рис. 8 иллюстрируется применение dom-элементов для обработки XML-документов.

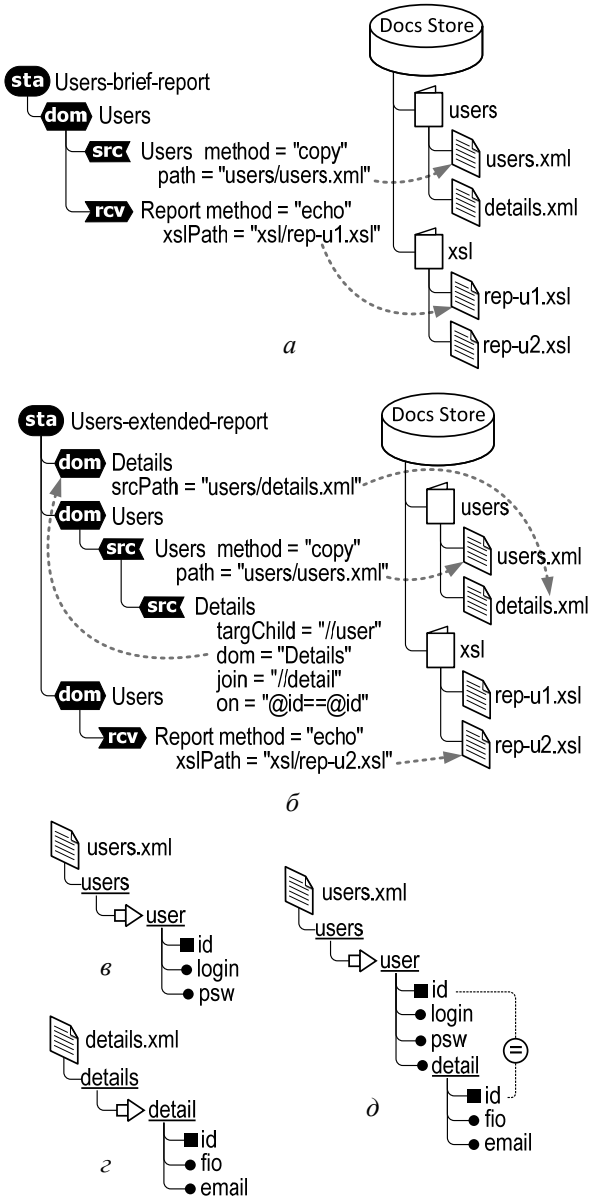

Рис. 8. Примеры обработки XML-документов

Рис. 8, *а* демонстрирует простой случай создания DOM-объекта на основе одного докумен-

та, а рис. 8, *б* – более сложный случай, когда содержимое DOM-объекта формируется из двух XML-документов. Модели исходных XML-документов приведены на рис. 8, *в* и *г*, а результирующего документа – на рис. 8, *д* (для представления структуры XML-документов использована нотация, предложенная в книге [18]).

Создание и загрузка DPO-объектов. В состоянии sta:Users-brief-report элемент dom:Users задает DOM-объект, содержащий краткие сведения о зарегистрированных пользователях приложения. В создаваемый DOM-объект с помощью элемента-источника src:Users загружается файл users.xml из директории users хранилища данных.

В состоянии sta:Users-extended-report первый элемент dom:Users задает DOM-объект, содержащий помимо кратких детальные сведения о зарегистрированных пользователях. Для этого сначала создается вспомогательный объект dom:Details, в который загружается файл details.xml, содержащий детальные сведения пользователей.

Далее интерпретатор создает и загружает объект dom:Users путем слияния (equijoin – эквисоединения) кратких и детальных сведений пользователей. А именно, каждому XML-элементу user из документа users.xml прикрепляется в качестве дочернего XML-элемент detail из dom:Details такой, что у них XML-атрибуты id имеют одинаковое значение (см. рис. 8, *д*). Для этого используется элемент-источник src:Details, вложенный в элемент-источник src:Users (см. рис. 8, *б*).

Источник src:Details содержит следующие атрибуты:

- targChild – содержит XPath-выражение, задающее множество целевых (родительских) узлов (в данном случае – множество всех элементов user из документа users.xml), к которым должны прикрепляться дочерние узлы этого источника;
- dom – содержит ссылку на ранее созданный DOM-объект, содержащий узлы этого источника;
- join – содержит XPath-выражение, задающее множество прикрепляемых (дочерних) узлов (в данном случае – множество всех элементов detail из документа details.xml), которые должны прикрепляться к целевым узлам;
- on – содержит условие соединения узлов (в данном случае требуется, чтобы атрибуты id в соединяемых узлах имели одинаковое значение).

Данный пример, в частности, демонстрирует использование нескольких DOM-объектов, задаваемых в одном состоянии. Отметим, что рассмотренное слияние XML-документов можно выполнить и без вспомогательного DOM-объекта, а применить источники, ссылающиеся непосредственно на файлы в хранилище документов.

Использование DOM-объектов для формирования результата. На рис. 9 поясняется формирование результата запроса с помощью элементов-приемников rcv:Report (см. рис. 8, *а* и *б*). Результат формируется в виде HTML-документа.

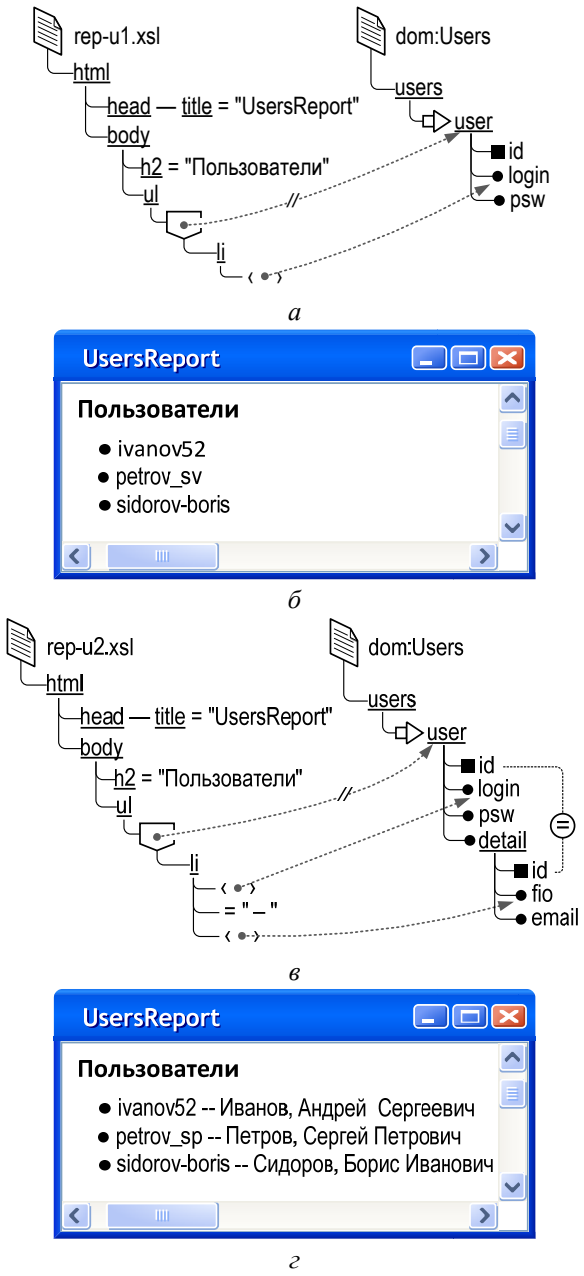

Рис. 9. Примеры формирования результирующего HTML-документа путем XSL-трансформации содержимого DOM-объекта

Указанные приемники задают XSL-трансформацию содержимого родительского DOM-объекта в соответствии с таблицами стилей, размещенными в хранилище документов в директории xsl. Модели трансформации, в соответствии с которыми интерпретатор формирует краткий и детальный отчеты, приведены на рис. 9, *а* и *в* (использована нотация, предложенная в книге [18]). Результирующие экранные формы представлены на рис. 9, *б* и *г*.

Отметим, что на рис. 8, *б* элемент-приемник, формирующий детальный отчет, размещен в дубликате элемента dom:Users. Вообще говоря, указанный приемник можно разместить в том же элементе dom:Users, в котором размещены источники src:Users и src:Details. Разделение сделано для иллюстрации возможности многократного использования dom-элементов, имеющих одинаковое имя, для ссылок на соответствующий DOM-объект (в рамках как одного, так и нескольких состояний).

6. ГЕТЕРОГЕННОЕ ХРАНИЛИЩЕ ДОКУМЕНТОВ

Первоначально предполагалось, что СОБД обрабатывает документы в формате XML, для чего в HSM были предусмотрены dom-элементы. Когда практические задачи потребовали обрабатывать JSON-документы, в HSM дополнительно были введены smarty-элементы. В настоящее время DPO-элементы ограничиваются этими двумя типами. Если в дальнейшем потребуется распространить СОБД на обработку документов других типов, то одним из путей решения этой задачи является введение в HSM новых разновидностей DPO-элементов.

Другой путь решения задачи обработки разнородных (гетерогенных) документов состоит в преобразовании «на лету» исходных документов в имеющиеся DPO-форматы при загрузке и обратно при выгрузке. В результате приходим к концепции виртуального хранилища документов, которое отображается на множество реальных документов и их хранилищ.

На рис. 10 представлена архитектура СОБД, в которой (в отличие от модели на рис. 5) хранилище документов является виртуальным.

Виртуальное хранилище документов (Virtual Docs Store) предоставляет набор виртуальных документов в формате XML, JSON и др., которые отображаются (Mapping) на реальные документы в различных форматах. Если необходимо загрузить в DPO виртуальный документ, выполняется его извлечение из реального хранилища и преобразование «на лету» в требуемый формат. При необходимости выгрузить из DPO виртуальный документ выполняется обратная процедура преобразования и сохранения.

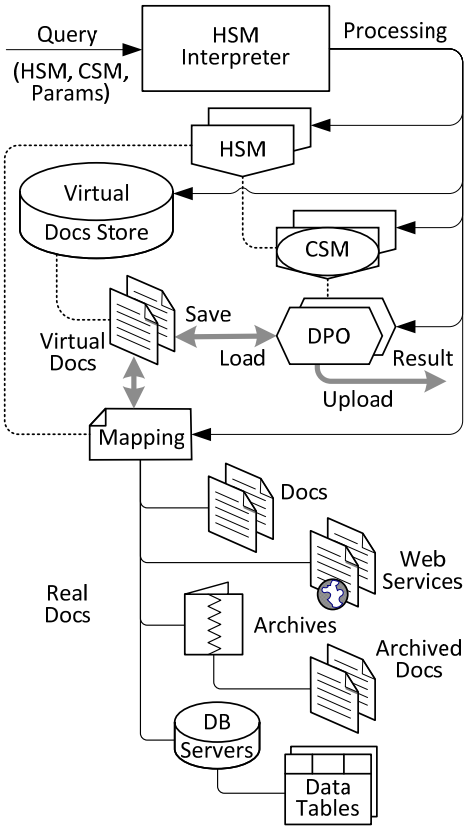

Рис. 10. Архитектура СОБД с виртуальным хранилищем документов

Перечень реальных типов документов, на которые могут отображаться документы виртуального хранилища, включает:

- документы в форматах XML и JSON под управлением файловой системы;
- документы в тех же форматах, предоставляемые веб-сервисами через сеть;
- документы, упакованные в ZIP-, RAR- и др. архивах;
- документы, «упакованные» в реляционных таблицах под управлением серверов баз данных и др.

Элементы doc. Чтобы учесть специфику указанных форматов в процессе загрузки-выгрузки документов, интерпретатору требуются соответствующие спецификации. Эти спецификации предоставляют doc-элементы, предусмотренные в HSM, задающие виртуальные документы и их отображение на реальные документы.

На рис. 11 иллюстрируется простейшее использование doc-элемента.

Элемент doc:Users представляет виртуальный XML-документ, который отображается на реальный документ users/users.xml. Источник

src:Users ссылается на виртуальный документ, а через него – на соответствующий реальный документ.

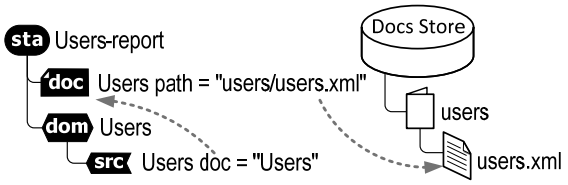

Рис. 11. Задание виртуального документа и отображение его на реальный документ

Совокупность doc-элементов образует промежуточный слой, который можно назвать *слоем интеграции*, поскольку он обеспечивает единообразный доступ к разнородным документам в форме виртуальных документов XML или JSON.

В настоящее время эта концепция реализована частично и охватывает отображение виртуальных XML-документов на реальные XML-документы (тривиальный случай), на XML-документы, предоставляемые веб-сервисами [17], на XML-документы в ZIP-архивах [19].

В качестве примера на рис. 12 представлен HSM-фрагмент, обеспечивающий модификацию через DOM-объект файла header1.xml из ZIP-архива w-1.docx (документа Word).

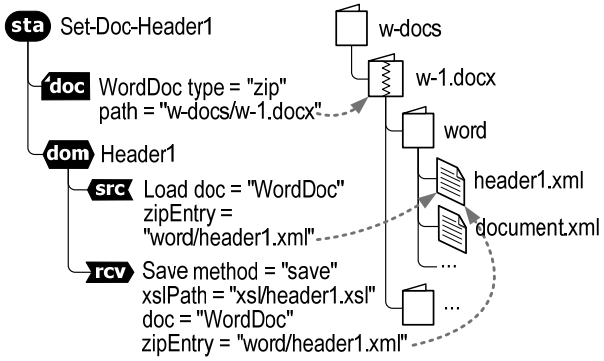

Рис. 12. Пример обработки документа из ZIP-архива

Как известно, Word-документ в формате Office Open XML представляет собой набор XML-файлов, упакованных в ZIP-архиве. Верхний колонтитул первой страницы размещен в файле word/header1.xml. На рис. 12 HSM-элемент doc:WordDoc ссылается на документ w-docs/w-1.docx как на ZIP-архив. Элемент dom:Header1 порождает DOM-объект, в который загружается файл word/header1.xml с помощью элемента-источника src:Load. Элемент-приемник rcv:Save обеспечивает XSL-трансформацию содержимого DOM-объекта и возврат результата на прежнее место в ZIP-архив.

ЗАКЛЮЧЕНИЕ

1. СОБД представляет новый подход к построению приложений обработки данных, базирующийся на принципах управления по модели (Model Driven Approach).

2. В основе СОБД лежит встроенная динамическая модель с конечным числом состояний в форме графов переходов состояний (ситуаций), отражающая логику бизнес-процесса, обслуживаемого приложением обработки данных.

3. Управление данными осуществляется путем интерпретации встроенной динамической модели с отслеживанием ее текущих состояний и доступа к данным, ассоциированным с текущими состояниями.

4. Для задания встроенных динамических моделей разработан декларативный язык HSM, позволяющий описывать в графическом или текстовом виде иерархию субмоделей, содержащих несколько состояний, для которых, в свою очередь, можно задать переходы состояний, акции и другие субмодели.

5. Интерпретатор HSM на каждом цикле интерпретации выполняет несколько нисходящих проходов динамической модели. CSM – модель текущего состояния – формируется интерпретатором в виде поддерева HSM, в котором для субмоделей указаны их текущие состояния.

5. Для задания спецификаций обработки документов в HSM предусмотрены элементы обработки документов, порождающие DOM- или Smarty-объекты. Для загрузки документов в объекты в HSM предусмотрены элементы-источники.

6. Для задания обработки загруженных документов и выгрузки результатов в HSM предусмотрены элементы-приемники. XML-документы обрабатываются путем XSL-трансформации, а JSON-документы – путем компилирующей обработки шаблонов.

7. Возможность доступа к разнородным документам в форме виртуальных документов XML или JSON обеспечивается слоем интеграции с помощью doc-элементов, задающих преобразование формата документов «на лету» с учетом специфики реальных хранилищ.

8. Дальнейшее развитие СОБД в рамках рассмотренных концепций предполагает расширение ассортимента обрабатываемых форматов документов за счет введения, во-первых, новых типов элементов обработки документов, во-вторых, – новых типов виртуальных документов.

СПИСОК ЛИТЕРАТУРЫ

1. **Strauch C., Kriha W.** (2012, Nov. 11). *NoSQL databases* [Online]. Available: http://www.christof-strauch.de/nosqldbs.pdf

2. **Миронов В. В., Юсупова Н. И., Шакирова Г. Р.** Ситуационно-ориентированные базы данных: концепция, архитектура, XML-реализация // Вестник УГАТУ. 2010. Т. 14, № 2 (37). С. 233–244. [[V. V. Mironov, N. I. Yusupova, and G. R. Shakirova, “Situation-oriented databases: concept, architecture, XML realization,” (in Russian), *Vestnik UGATU*, vol. 14, no. 4 (39), pp. 200-209, 2010.]]

3. **Миронов В. В., Юсупова Н. И., Шакирова Г. Р.** Ситуационно-ориентированные базы данных: внешние представления на основе XSL // Вестник УГАТУ. 2010. Т. 14, № 4 (39). С. 200–209. [[V. V. Mironov, N. I. Yusupova, and G. R. Shakirova, “Situation-oriented databases: external view in the basis of XSL,” (in Russian), *Vestnik UGATU*, vol. 14, no. 2 (37), pp. 233-244, 2010.]]

4. **Миронов В. В., Маликова К. Э.** Интернет-приложения на основе встроенных динамических моделей: идея, концепция, безопасность // Вестник УГАТУ. 2009. Т. 13, № 2 (35). С. 167–179. [[V. V. Mironov and K. E. Malikova, “Internet applications based on embedded dynamic models: idea, concept,” (in Russian), *Vestnik UGATU*, vol. 13, no. 2 (35), pp. 167-179, 2009.]]

5. **Миронов В. В., Маликова К. Э.** Интернет-приложения на основе встроенных динамических моделей: архитектура, структура данных, интерпретация // Вестник УГАТУ. 2010. Т. 14, № 1 (36). С. 154–163. [[V. V. Mironov and K. E. Malikova, “Internet applications based on embedded dynamic models: architecture, data structure, interpretation,” (in Russian), *Vestnik UGATU*, vol. 14, no. 1 (36), pp. 154-163, 2010.]]

6. **Миронов В. В., Маликова К. Э.** Интернет-приложения на основе встроенных динамических моделей: элементы управления пользовательского интерфейса // Вестник УГАТУ. 2010. Т. 14, № 5 (40). С. 170–175. [[V. V. Mironov and K. E. Malikova, “Internet applications based on embedded dynamic models: user interface controls,” (in Russian), *Vestnik UGATU*, vol. 14, no. 5 (40), pp. 170-175, 2010.]]

7. **Миронов В. В., Гусаренко А. С.** Ситуационно-ориентированные базы данных: концепция управления XML-данными на основе динамических DOM-объектов // Вестник УГАТУ. 2012. Т. 16, № 3 (48). С. 159–172. [[V. V. Mironov, A. S. Gusarenko, “Situation-oriented databases: the concept of managing XML-data based on dynamic DOM-objects,” (in Russian), in *Vestnik UGATU*, vol. 16, no. 3 (48), pp. 159-172, 2012.]]

8. **Миронов В. В., Гусаренко А. С.** Динамические DOM-объекты в ситуационно-ориентированных базах данных: лингвистическое и алгоритмическое обеспечение источников данных // Вестник УГАТУ. 2012. Т. 16, № 6 (51). С. 167–176. [[V. V. Mironov, A. S. Gusarenko, “Dynamic DOM-objects in situation-oriented databases: lingware and knoware of data sources,” (in Russian), in *Vestnik UGATU*, vol. 16, no. 6 (51), pp. 167-176, 2012.]]

9. **Гусаренко А. С.** Обработка XML-документов в ситуационно-ориентированных базах данных на основе динамических DOM-объектов: автореф. дис. … канд. техн. наук / Уфимск. гос. авиац. техн. ун-т. Уфа, 2013. [[A. S. Gusarenko, *Handling XML-documents in situationally-oriented databases based on dynamic DOM-objects*, (in Russian), Thesis abstract for the degree of candidate of technical sciences, UGATU, 2013.]]

10. **Канашин В. В., Миронов В. В.** Иерархические виджеты: организация интерфейса пользователя в веб-приложениях на основе ситуационно-ориентированных баз данных // Вестник УГАТУ. 2013. Т. 17, № 2 (55). С. 138–149. [[V. V. Mironov, V. V. Kanashin, “Hierarchical widgets: user interface organization in web applications on the basis of situation-oriented databases,” *Vestnik UGATU*, vol. 17, no. 2 (55), pp. 138-149, 2013.]]

11. **Канашин В. В., Миронов В. В.** Иерархические виджеты: ввод и контроль данных пользователя в веб-приложениях на основе ситуационно-ориентированных баз данных // Вестник УГАТУ. 2013. Т. 17, № 5 (58). С. 166–176. [[V. V. Mironov, V. V. Kanashin, “Hierarchical widgets: input and control of the user data in web applications on the basis of situation-oriented databases,” *Vestnik UGATU*, vol. 17, no. 5 (58), pp. 166-176, 2013.]]

12. **Канашин В. В., Миронов В. В.** Иерархические виджеты: алгоритмы контроля данных пользователя в веб-приложениях на основе ситуационно-ориентированных баз данных // Вестник УГАТУ. 2014. Т. 18, № 1 (62). С. 204–213. [[V. V. Mironov, V. V. Kanashin, “Hierarchical widgets: user data control algorithms in web applications on the basis of situation-oriented databases,” *Vestnik UGATU*, vol. 18, no. 1 (62), pp. 204-213, 2014.]]

13. **Канашин В. В., Миронов В. В.** Иерархические виджеты: опыт применения в веб-приложении на основе ситуационно-ориентированной базы данных // Вестник УГАТУ. 2014. Т. 18, № 2 (63). С. 185–196. [[V. V. Mironov, V. V. Kanashin, “Hierarchical widgets: experience of use in the web application on the basis of situation-oriented database,” *Vestnik UGATU*, vol. 18, no. 2 (63), pp. 185-196, 2014.]]

14. **Макарова Е. С., Миронов В. В.** Проектирование концептуальной модели данных для задач Web-OLAP на основе ситуационно-ориентированной базы данных // Вестник УГАТУ. 2012. Т. 16, № 6 (51). С. 177–188. [[E. S. Makarova and V. V. Mironov, “Web OLAP conceptual data model design on the basis of situation-oriented database,” (in Russian), *Vestnik UGATU*, vol. 16, no. 6 (51), pp. 177-188, 2012.]]

15. **Макарова Е. С., Миронов В. В.** Функции аналитики в веб-приложениях на основе ситуационно-ориентированных баз данных // Вестник УГАТУ. 2013. Т. 17, № 5 (58). С. 150–165. [[E. S. Makarova and V. V. Mironov, “Analytical functions in web applications based on situation-oriented databases,” (in Russian), *Vestnik UGATU*, vol. 17, no. 5 (58), pp. 150-165, 2013.]]

16. **Гусаренко А. С., Миронов В. В.** Smarty-объекты: вариант использования гетерогенных источников в ситуационно-ориентированных базах данных // Вестник УГАТУ. 2014. Т. 18, № 3 (63). С. 242–252. [[A. S. Gusarenko, V. V. Mironov, “Smarty-objects: use case of heterogeneous sources in situation-oriented databases,” (in Russian), *Vestnik UGATU*, vol. 18, no. 3 (63), pp. 242-252, 2014.]]

17. **Гусаренко А. С., Миронов В. В.** Использование RESTful-сервисов в ситуационно-ориентированных базах данных // Вестник УГАТУ. 2015. Т. 19, № 1 (67). С. 204–211. [[A. S. Gusarenko, V. V. Mironov, “Using of RESTful-services in situation-oriented databases,” (in Russian), *Vestnik UGATU*, vol. 19, no. 1 (67), pp. 204-211, 2015.]]

18. **Миронов В. В., Юсупова Н. И., Шакирова Г. Р.** Иерархические модели данных: концепции и реализация на основе XML / под ред. проф. Н. И. Юсуповой. М.: Машиностроение, 2011. 453 с. [[V. V. Mironov, N. I. Yusupova, and G. R. Shakirova, *Hierarchical data model: the concept and im-*

plementation of based on XML, (in Russian). Moscow: Mashinostroenie, 2011.]]

19. **Миронов В. В., Гусаренко А. С., Диметриев Р. Р., Сарваров М. Р.** Создание персонализированных документов на основе ситуационно-ориентированной базы данных // Вестник УГАТУ. 2014. Т. 18, № 4 (65). С. 191–197. [[V. V. Mironov, A. S. Gusarenko, R. R. Dimetriev, and M. R. Sarvarov, "The personalized documents generating using DOM-objects in situation-oriented databases," (in Russian), *Vestnik UGATU*, vol. 18, no. 4 (65), pp. 191-197, 2014.]]

ОБ АВТОРАХ

МИРОНОВ Валерий Викторович, проф. каф. автоматизированных систем упр. Дипл. р/физ. (Воронежск. гос. ун-т, 1975). Д-р техн. наук по упр. в техн. системах (УГАТУ, 1995). Иссл. в обл. иерарх. моделей и сит. упр.

ЮСУПОВА Нафиса Исламовна, проф., зав. каф. выч. мат. и кибернет., декан фак. информатики и робототехн. Дипл. радиофизик (Воронежск. гос. ун-т, 1975). Д-р техн. наук по упр. в техн. системах (УГАТУ, 1998). Иссл. в обл. критич. сит. упр, информатики.

ГУСАРЕНКО Артем Сергеевич, асс. каф. автоматизированных систем упр. Канд. техн. наук (УГАТУ, 2013). Дипл. инф.-экон. (УГАТУ, 2010). Иссл. в обл. иерарх. моделей, сит.-ориент. баз данных, NoSQL.

METADATA

Title: Situation-oriented databases: current state and prospects for research.

Authors: V. V. Mironov, N. I. Yusupova, and A. S. Gusarenko.

Affiliation:
Ufa State Aviation Technical University (UGATU), Russia.

Email: mironov@list.ru, yussupova@ugatu.ac.ru, artyomgusarenko@gmail.com.

Language: Russian.

Source: Vestnik UGATU, vol. 19, no. 2 (68), pp. 188-199, 2015. ISSN 2225-2789 (Online), ISSN 1992-6502 (Print).

Abstract: On a conceptual level, the current state of research in situation-oriented databases (SOBD) is discussed. SOBD considered as a dynamic system within the problem of data processing applications building. It is noted that the presence in the SOBD an embedded dynamic model allows to solve the problem on a higher level of abstraction. We consider hierarchical situational models HSM, underlying the SOBD as a meta-model for dynamic modeling of specific applications. HSM elements intended for document processing based on the concept of data processing objects are discussed. A concept of virtual documents, enabling the processing of real documents specified in a variety of formats and uses a variety of storage media, is discussed.

Keywords: situation-oriented database; web-application; dynamic model; NoSQL; HSM; XML; JSON; DOM; Smarty; PHP.

About authors:

MIRONOV, Valeriy Viktorovich, Prof., Dept. of Automated Systems. Dipl. Radiophysicist (Voronezh State Univ., 1975). Dr. of Tech. Sci. (USATU, 1995).

YUSUPOVA, Nafisa Islamovna, Prof. Dr.-Eng. Dean of the Faculty of Computer Science and Robotics, Head of the Dept. of Computational Mathematics and Cybernetics. Dipl. Radiophysicist (Voronezh State Univ., 1975). Dr. (Habil.) Tech. Sci. (UGATU, 1998).

GUSARENKO, Artem Sergeevich, Cand. of Tech. Sci. (USATU, 2013), Dept. of Automated Systems., Grad. informatic-economist (USATU, 2010).