ОРГАНИЗАЦИЯ РАСПРЕДЕЛЕННОГО МНОГОПОЛЬЗОВАТЕЛЬСКОГО РЕЖИМА ПОДГОТОВКИ ИСХОДНЫХ ДАННЫХ ДЛЯ РЕШЕНИЯ ЗАДАЧ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ ТРАНСПОРТНЫХ СИСТЕМ МЕТОДАМИ БАЗЫ ДАННЫХ PostgreSQL

А. Г. Надуев, А. Д. Черевань, Д. А. Кожаев

ООО «Центр компетенций и обучения», г. Саров

Требования к многопользовательскому режиму

Одним из важных требований при разработке современных комплексов математического моделирования является организация многопользовательского режима разработки сценария (полная постановка начальных условий расчетной задачи), при котором неограниченное количество пользователей, используя один из компонентов программного комплекса препостпроцессор (ППП), имеет возможности одновременно:

- редактировать существующие объекты сценария;
- создавать новые объекты сценария;
- удалять устаревшие объекты сценария;
- улаживать конфликты различных версий сценария;
- восстанавливать одно из предыдущих состояний любого объекта, используя историю всех изменений сценария.

Всем перечисленным выше требованиям отвечают системы управления версиями, одна из которых – «Mercurial» [1] была выбрана в качестве «донора» интерфейса, разрабатываемого многопользовательского режима разработки сценария.

Постановка задачи

Для реализации многопользовательского режима необходимо:

- предоставить абстрактный интерфейс хранения контекста любого объекта сценария для сохранения всех объектов с использованием единого подхода;
- определить возможность уникальной идентификации всех объектов сценария для использования сквозной нумерации/идентификации объектов как в ППП, так и в контейнере хранения;
- определить общее, доступное удаленно, хранилище всех версий разрабатываемого сценария расчетной задачи;
- предоставить единый интерфейс сохранения новых версий одного сценария и общий интерфейс улаживания конфликтов версий одного и того же сценария, созданных различными пользователями.

XML как контейнер хранения контекста объекта

Для реализации абстрактного интерфейса хранения контекста любого объекта данных был выбран расширяемый язык разметки XML (e**X**tensible **M**arkup **L**anguage) [2], который позволяет сериализовать/десериализовать объекты сценария с любой глубиной вложенности в единое текстовое (XML) поле. Этот подход имеет следующие положительную и отрицательную особенности:

- положительная – контекст всех объектов хранится единообразно, для его сериализации/десериализации используются общие функции;

• отрицательная – атомарной информационной единицей является объект. Для анализа отдельных значений атрибутов объекта требуется предварительная десериализация объекта.

Уникальный идентификатор объектов сценария

Уникальным идентификатором всех объектов выбрано целочисленное значение *sysId,* выдаваемое ППП каждому новому объекту при его создании. Для обеспечения уникальной идентификации объектов как в ППП, так и в общем хранилище используется пороговое значение (*threshold*), разделяющее идентификаторы объектов на:

• «глобальные» – уже находящиеся в общем хранилище. Эти идентификаторы не могут быть изменены в дальнейшем;

• «локальные» – идентификаторы объектов, существующих пока только в ППП, которые могут быть перенумерованы при попадании в общее хранилище.

Полный цикл работы с *sysId* и *threshold* выглядит следующим образом:

1. При получении копии сценария из общего хранилища фиксируется *threshold* текущего максимального идентификатора объектов актуальной, глобальной ревизии сценария.

2. При создании новых объектов в ППП им выдается уникальный *sysId,* превышающий *threshold*.

3. В момент слияния двух версий сценария – локальной и глобальной вновь созданные локальные объекты автоматически перенумеровываются в соответствии с текущим состоянием базы – разница между *threshold* различных ревизий сценария локальной и глобальной добавляется к идентификаторам новых объектов.

4. Перенумерованные объекты во всех источниках имеют единый уникальный номер.

СУБД как контейнер хранения сценария задачи

В качестве общего удаленного хранилища всех версий разрабатываемого сценария расчетной задачи была выбрана современная, свободная, объектно-реляционная система управления базами данных (СУБД) PostgreSQL [3], которая по своим характеристикам полностью подходит для реализации подобного распределенного многопользовательского режима, хотя используемые особенности этой СУБД столь минимальны, что вместо нее возможно использовать любую другую современную SQL СУБД.

Основные единицы реализации:

• отдельная схема для хранения сценария (например, под названием *«scenario»*);

• таблица ревизий (например, *«revisions»*) для обеспечения контроля версий;

• таблица объектов (например, *«objects»*), содержащая все объекты сценария;

• триггеры таблицы данных, вызывающие триггерные функции перед добавлением, изменением или удалением строки таблицы;

• триггерные функции, осуществляющие основную работу по управлению версиями объектов структуры данных ППП.

Таблица ревизий

Таблица ревизий *«revisions»* имеет следующие столбцы:

• serial *id* – автоматически наращиваемый номер версии;

• timestamp *timestamp* – метка времени создания версии;

• text *owner* – имя пользователя, создавшего версию;

• text *message* – сообщение, которым подписана версия.

Последовательность работы с таблицей *«revisions»*:

1. Новое значение в таблице появляется в момент начала транзакции внесения изменении в базу.

2. Посылка сообщения в информационный канал о параметрах новой ревизии для уведомления всех пользователей, соединенных с базой данных.

3. Новый (далее текущий) номер ревизии используется для подписи состояния всех объектов данных ППП, присутствующих в базе.

Пример таблицы ревизий приведен на рис. 1.

	id integer	timestamp timestamp without time zone	owner text	message text
1	1	2014-05-27 13:56:25.09	postgres	Init

Рис. 1. Пример таблицы ревизий

Таблица объектов

Таблица объектов *«objects»* имеет следующие столбцы:

- int *sys_id* – уникальный идентификационный номер выданный объекту ППП;
- xml *context* – контекст объекта в виде XML типа;
- int *begin_revision* – начальная ревизия, указывает номер ревизии, когда объект (строка) был создан в БД;
- int *end_revision* – конечная ревизия, указывает номер ревизии, когда объект (строка) был удален из БД. Действующий объект, в столбце *end_revision* имеет значение *MAX_INT* = 2147483647.

Таким образом, комбинация трех полей *sys_id*, *begin_revision* и *end_revision* представляет первичный ключ, который однозначно индексирует объект в таблице.

Операции над объектами

- Добавление объекта:

1. добавляется новая строка в таблицу с *sysId* и контекстом объекта;

2. триггерная функция устанавливает *begin_revision* = номер_текущей_ревизии (максимальный *id* из таблицы ревизий), *end_revision* изменяется на значение *MAX_INT*.

- Изменение объекта:

1. триггерная функция у текущей строки с *sysId* объекта изменяет *end_revision* со значения *MAX_INT* на номер_текущей_ревизии (максимальный *id* из таблицы ревизий).

2. добавляется новая строка с sysId объекта и новым контекстом, у которой: *begin_revision* = = номер_текущей_ревизии (максимальный *id* из таблицы ревизий), а *end_revision* = *MAX_INT*.

- Удаление объекта – триггерная функция у текущей строки с *sysId* изменяет *end_revision* со значения *MAX_INT* на номер_текущей_ревизии (максимальный *id* из таблицы ревизий).

Пример таблицы объектов приведен на рис. 2.

	sys_id integer	context xml	begin_revision integer	end_revision integer
1	777	<data/>	1	5
2	777	<data/>	5	10
3	777	<data/>	10	11
4	778	<data/>	1	2147483647

Рис. 2. Пример таблицы объектов

Рассмотрим подробнее пример, приведенный на рис. 2:

• первая строка описывает состояние объекта с *sysId* = 777 в промежутке между 1-ой и 5-ой ревизиями (подробности о ревизии можно узнать из таблицы ревизий);

• вторая строка описывает состояние объекта с *sysId* = 777 в промежутке между 5-ой и 10-ой ревизиями;

• третья строка описывает состояние объекта с *sysId* = 777 в промежутке между 10-ой и 11-ой ревизиями. Так как больше записей о состоянии с *sysId* = 777 больше нет, он был удален из базы в 11-ой ревизии;

• четвертая строка описывает состояние объекта с *sysId* = 778, начиная с первой ревизии по текущую – оно не изменялось, объект действующий.

Операции над сценарием

• Получение состояния проекта (*snapshot*) на момент какой-либо ревизии осуществляется с помощью вызова функции *get_scenario_objects ()*, которая содержит запрос к таблице объектов с условием выборки *«WHERE version >= begin_revision and version < end_version»*.

• Получение разницы между двумя ревизиями проекта $v1 < v2$ (*diff*) осуществляется с помощью последовательного вызова двух функций:

1. *get_added_and_changed_scenario_objects ()* возвращает объекты, добавленные и измененные в промежутке между двумя ревизиями. Эта функция осуществляет запрос к таблице объектов со следующим условием выборки *«WHERE v1 < begin_revision and v2 < end_revision and v2 >= = begin_revision»*;

2. *get_removed_scenario_objects ()* возвращает список объектов, удаленных в промежутке между двумя ревизиями.

Реализация со стороны ППП

Единый интерфейс распределенного многопользовательского режима со стороны ППП обладает следующими возможностями/особенностями:

• разработка сценария, в терминах систем управления версиями, ведется в одной «ветке» (поддержка «ветвлений» находится вне зоны интересов данного подхода);

• слияние разных версий сценария ведется на крупноблочном уровне – информационная единица объект (текущее состояние реализации диалога «ручного» улаживания конфликтов легко изменяется путем расширения его возможностей);

• возможна сериализация/десериализация локальной версии сценария в файл XML формата;

• получение разницы между ревизиями в «экономичном» режиме – только версии объектов, отсутствующие в ППП;

• автоматическое улаживание конфликтов при отсутствии спорных ситуаций;

• «ручное» улаживание конфликтов с помощью специального диалога в случае спорных ситуаций;

• возможна отмена и необходимость в повторном слиянии двух версий сценария, если процедура выполняется несколькими пользователями одновременно.

Схема редактирования сценария в многопользовательском режиме приведена на рис. 3.

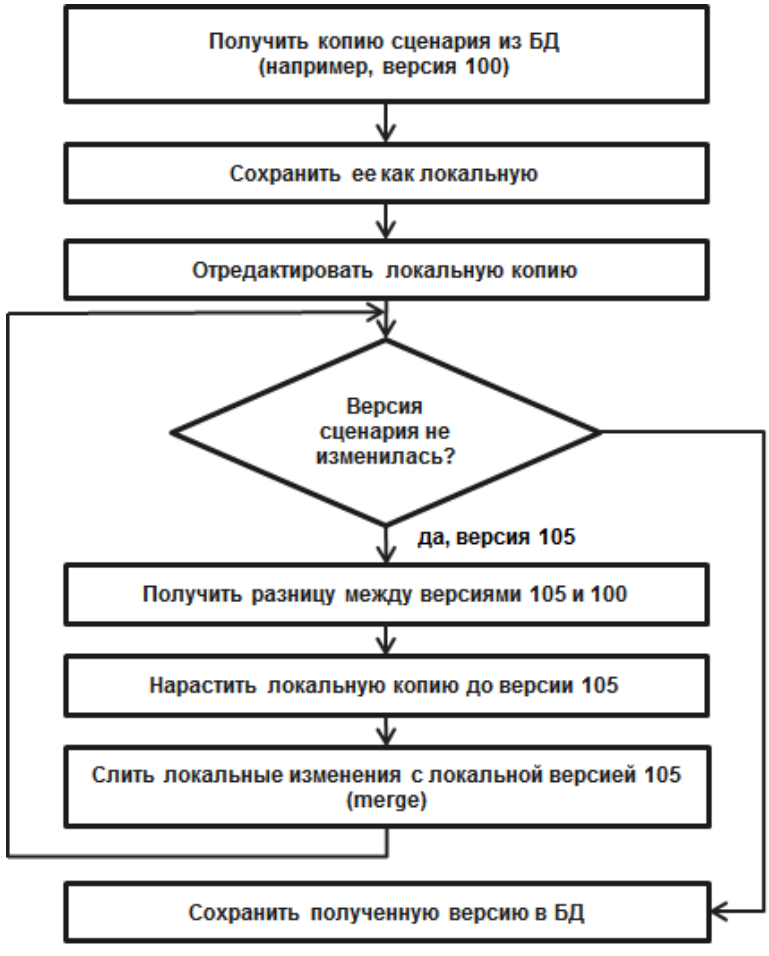

Рис. 3. Схема редактирования сценария в многопользовательском режиме

Режим автоматического слияния

Автоматическое слияние сценария между локальной и удаленной версиями выполняется в том случае, если не возникают спорные ситуаций, следующим образом:

1. Формирование последней доступной сохраненной в БД ревизии сценария (конечный результат такой процедуры аналогичен полной загрузке последней ревизии сценария, но выполняется оптимизированным образом без лишней передачи неизменившихся данных – «наложение» разницы сценариев (*diff*)).
2. Добавление вновь созданных в локальной версии объектов.
3. Удаление локально удаленных объектов, если в глобальном хранилище не было изменений этих объектов.
4. Изменение локально измененных объектов, если в глобальном хранилище не было изменений этих объектов.

Диалог «ручного» слияния «конфликтных» версий сценариев

Если при слиянии локальной и глобальной версий сценариев происходят спорные ситуации, то будет вызван диалог «ручного» улаживания конфликтов. Простейший диалог «ручного» улаживания конфликтов представлен на рис. 4.

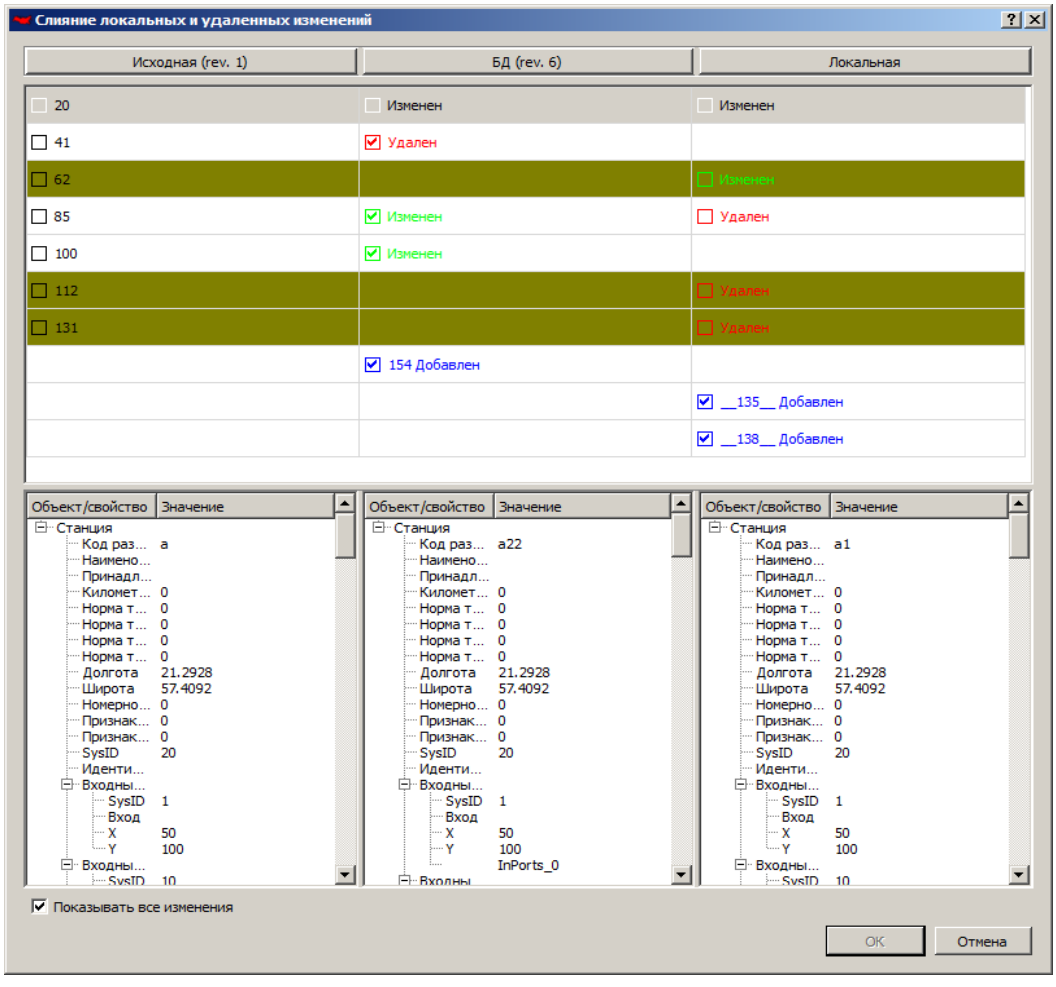

Рис. 4. Диалог «ручного» улаживания конфликтов

Представленное диалоговое окно вызывается в случае наличия спорных ситуаций, а именно:

- объект был удален в локальной версии, но не удален в удаленной версии;
- объект был удален в удаленной версии, но не удален в локальной версии.

Диалоговое окно слияния выполнено в виде таблицы и содержит три колонки:

- исходная глобальная ревизия – ревизия, на которой базируется локальная версия сценария в ППП;
- текущая глобальная ревизия – актуальная ревизия сценария в БД;
- локальная ревизия – ревизия исходного глобального сценария, в которую внесены локальные изменения.

В каждой колонке представлен список измененных объектов и их контекста в виде «дерева».

Для наглядной идентификации изменений используется следующая цветовая гамма:

- красный цвет шрифта – удаленный объект;
- зеленый цвет шрифта – измененный объект;
- синий цвет шрифта – добавленный объект;
- светло-зеленый цвет фона всей строки – для данного объекта имеется неразрешенное противоречие.

Под разрешением противоречий подразумевается выбор одного из вариантов «конфликтного» объекта, который должен попасть в итоговую версию.

Добавленные объекты не вызывают противоречий слияния, однако при их показе можно ознакомиться с их содержимым и отказаться от их добавления в итоговую версию сценария.

Литература

1. Сайт «Mercurial SCM». [Электронный ресурс]. Режим доступа: http://mercurial.selenic.com.
2. Сайт «Extensible Markup Language (XML)». [Электронный ресурс]. Режим доступа: http://www.w3.org/XML.
3. Сайт «PostgreSQL». [Электронный ресурс]. Режим доступа: http://www.postgresql.org.

ИССЛЕДОВАНИЕ ПРИМЕНИМОСТИ КОММЕРЧЕСКИХ CFD-КОДОВ ДЛЯ МОДЕЛИРОВАНИЯ ПРОЦЕССОВ ТЕПЛОМАССОПЕРЕНОСА В ЖИДКОМЕТАЛЛИЧЕСКОМ ТЕПЛОНОСИТЕЛЕ

В. Р. Низамутдинов, С. Л. Осипов, И. С. Прокопцов, С. А. Рогожкин

ОАО «Опытное конструкторское бюро машиностроения им. И. И. Африкантова», г. Нижний Новгород

Введение

В моделях турбулентности, реализованных в большинстве Computational Fluid Dynamics (CFD) программных комплексах (ANSYS CFX, STAR-CCM+, FlowVision, Fluent и др.), для учета теплопереноса используется аналогия Рейнольдса – аналогия между теплообменом и переносом количества движения в турбулентном потоке. В жидких металлах вследствие большой теплопроводности и малой вязкости распределение полей скорости и температуры существенно различаются друг от друга. Таким образом, теплоперенос в средах с $Pr \ll 1$ (например, жидкие металлы) существенно отличается от механизма теплопереноса в средах с $Pr \sim 1$ (воздух, вода и др.) и использование моделей турбулентности с аналогией Рейнольдса для моделирования теплогидравлических процессов с натриевым теплоносителем может привести к некорректным результатам.

В данной работе представлены результаты численного моделирования течения натрия в круглой трубе с помощью коммерческих CFD-кодов: ANSYS CFX, STAR-CCM+, FlowVision и их сравнение с аналитическим решением.

Расчетная схема

В задаче моделируется развитое турбулентное течение жидкометаллического теплоносителя (натрия) в круглой трубе. Расчетная схема (рис. 1) состоит из двух участков. Первый участок – адиабатический (длина $l_1 = 40d$), принимается для формирования профиля скорости в трубе при численном моделировании течения, на втором участке (длиной $l_2 = 10d$) поддерживается постоянная температура стенки $t_c = 150$ °C. Температура теплоносителя на входе в трубу принимается равной $t_{вх} = 550$ °C. В результате решения определяется средняя температура теплоносителя на выходе из трубы $t_{вых}$ при варьировании критерия Пекле (Pe) от 300 до 10000 (путем изменения расхода на входе в трубу G), в соответствии с областью применения формулы для расчета критерия Нуссельта $Nu = 5 + 0{,}025Pe^{0{,}8}$ [1], используемой в аналитическом решении.