ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

SOFTWARE DESIGN

УДК 004.657; 004.041

МЕХАНИЗМЫ УПРАВЛЕНИЯ СОБЫТИЯМИ В АКТИВНЫХ БАЗАХ ДАННЫХ

Шибанов С.В., Вишняков П.В., Лысенко Э.В., Смирнов Д.С., Орешкин К.А.

Аннотация. Рассматривается проблема построения систем управления активными базами данных (СУАБД) средствами современных систем управления базами данных (СУАБД). Предлагается система метаданных для представления активных правил в *ECA*- и *SECA*-моделях. Исследуются механизмы управления событиями в современных реляционных и объектных СУБД, а также в системах управления событиями (*CEP*-системах) для оценки их применения в активных базах данных.

Ключевые слова. Системы управления активными базами данных, события, активные правила, метаданные, триггеры, асинхронные очереди, *CEP*-системы.

EVENT MANAGEMENT MECHANISMS IN ACTIVE DATABASES

Shibanov S.V., Vishnyakov P.V., Lysenko E.V., Smirnov D.S., Oreshkin K.V.

Abstract. The active database management system (ADBMS) design based on modern database management systems (DBMS) is discussed. ECA and SECA rules metadata are proposed. Event management in modern relational and object DBMSs as well as CEP systems is review and evaluated for use in active databases.

Keywords. Active database management system, events, active rules, metadata, triggers, asynchronous queues, CEP.

Проблема управления событиями в системах управления базами данных

С развитием технологий баз данных (БД) и систем управления базами данных (СУБД) увеличивается круг областей применения и решаемых ими задач. Изначально базы данных рассматривались лишь как репозитории, которые хранят информацию, необходимую для приложений, и доступны прикладным программам или пользователям. Традиционные системы управления базами данных пассивны в том смысле, что действия, выполняемые СУБД, инициируются пользователями или программными приложениями.

Примерами систем, при разработке которых приходится преодолевать пассивность традиционных баз данных, являются системы контроля и управления техническими объектами, системы управления движущимися объектами, геоинформационные системы, распределённые информационные системы и базы данных [1]. Как правило, в таких системах имеется большая быстро изменяющаяся во времени база элементарных фактов, описывающая состояния объектов. Система должна обнаруживать некоторое, возможно, большое число заданных ситуаций и автоматически реагировать на их возникновение соответствующими действиями. Каждая из обнаруживаемых ситуаций задается своей системой правил.

Чтобы обеспечивать реагирующее поведение, система управления активными базами данных (СУАБД) должна поддерживать соответствующие модели описания и выполнения правил. Классическая модель описания активных правил основывается на правилах, которые содержат три компонента: событие (event), условие (*condition*) и действие (*action*) и называются *ECA*-правилами (*Event-Condition-Action*) [2]. Компонент Событие (*event*) описывает какое-либо событие, которое может произойти в СУБД или вне нее. Компонент Условие (*condition*) проверяет контекст, при котором событие произошло. Действие (*action*) описывает процедуру, которая должна быть выполнена правилом, если соответствующее событие произошло и условие оказалось истинным. Действия могут менять структуру базы данных или набора правил, выполнять некоторый вызов поведения внутри базы данных или внешний вызов, информировать пользователя или системного администратора о какой-либо ситуации, прерывать транзакцию или принимать альтернативную линию поведения.

Классическая модель *ECA*-правил не учитывает, что в реальных приложениях поведение объекта часто зависит от его состояния (*state*). Состояние – это абстракция значений и связей объекта, которая определяет отклик объекта на получаемые события. В конкретном состоянии обрабатываются только те события, для которых явным образом описано поведение, любые другие события игнорируются. Это свойство состояний особенно важно – оно

позволяет наделить объекты системы свойством изменчивости – способности изменять свое поведение в зависимости от возникающих событий.

Модель, учитывающую состояние объекта называется расширенной моделью *ECA*-правил, моделью *SECA*-правил или *SECA*-моделью (*State-Event-Condition-Action*). Добавление в классическую модель *ECA*-правил понятия состояния сокращает количество необходимых правил и упрощает процесс разработки активной системы [2]. Выбор модели активных правил зависит от особенностей предметной области и логики выполнения активных правил.

Системы управления активными базами данных являют собой пример событийно-управляемой системы. Существую вполне удачные проекты реализации СУАБД. Однако большинство таких систем, например, описанных в [2] и [3], рассматривают управление событиями только исключительно в рамках собственных уникальных АБД и не предлагают общей концепции хранения и обработки событий, переносимой на другие платформы. Таким образом, проблема реализации событийного управления в СУАБД, построенных на платформе современных СУБД общего назначения, таких как *Microsoft SQL Server, Oracle* или *Cache*, остается открытой.

Система метаданных для управления активными правилами

В практических реализациях СУАБД, как правило, используется одна из указанных ранее моделей представления активных правил, а именно, *ECA*-модель или *SECA*-модель активных правил [2, 3]. Представляется актуальным разработка системы метаданных, поддерживающей обе модели активных правил и позволяющей создавать и исполнять активные правила в обеих моделях в рамках одной системы [4].

Логическая схема системы метаданных для управления активными правилами представлена на рис. 1. В представленной схеме каждое правило представляется в виде кортежа, включающего событие, условие и действие. Каждый из этих компонентов может обладать дополнительными параметрами – контекстом. Контекст позволяет передать обработчику правил параметры, необходимые при вычислении условия или выполнении действия. Помимо этого, *SECA*-правила связаны с состояниями объектов предметной области, управляя в процессе исполнения переводами объектов из одного состояния в другое.

Не вдаваясь в детали, можно отметить, что метаданные разделяются на метаданные времени конструирования и метаданные времени исполнения.

Метаданные времени конструирования предназначены для описания активных правил, их компонентов и свойств, влияющих на их исполнение [5]. Метаданные времени конструирования формируются на этапе создания и предварительного анализа активных правил. К метаданным времени конструирования относятся:

– *RuleItems* – класс для представления компонентов активных правил (событий, условий или действий);

– *DataTypes* – класс типов данных параметров, используемых при описании событий, условий и действий;

– *Parameters* – класс параметров, используемых при описании контекста событий, условий и действий;

– *Procs* – класс исполняемых компонентов (процедур, методов) активных правил (условий, действий);

– *ProcVersions* – версии исполняемой компоненты (процедуры, метода);

– *EventTypes* – класс типов событий;

– *SECAObjects* – класс информационных объектов для *SECA*-правил;

– *SECAStates* – класс состояний, определенных для информационных объектов *SECA*-правил;

– *Rules* – активные правила в *ECA*- и *SECA*-моделях;

– *ConnectedRules* – порожденные активные правила, определенные на этапе конструирования.

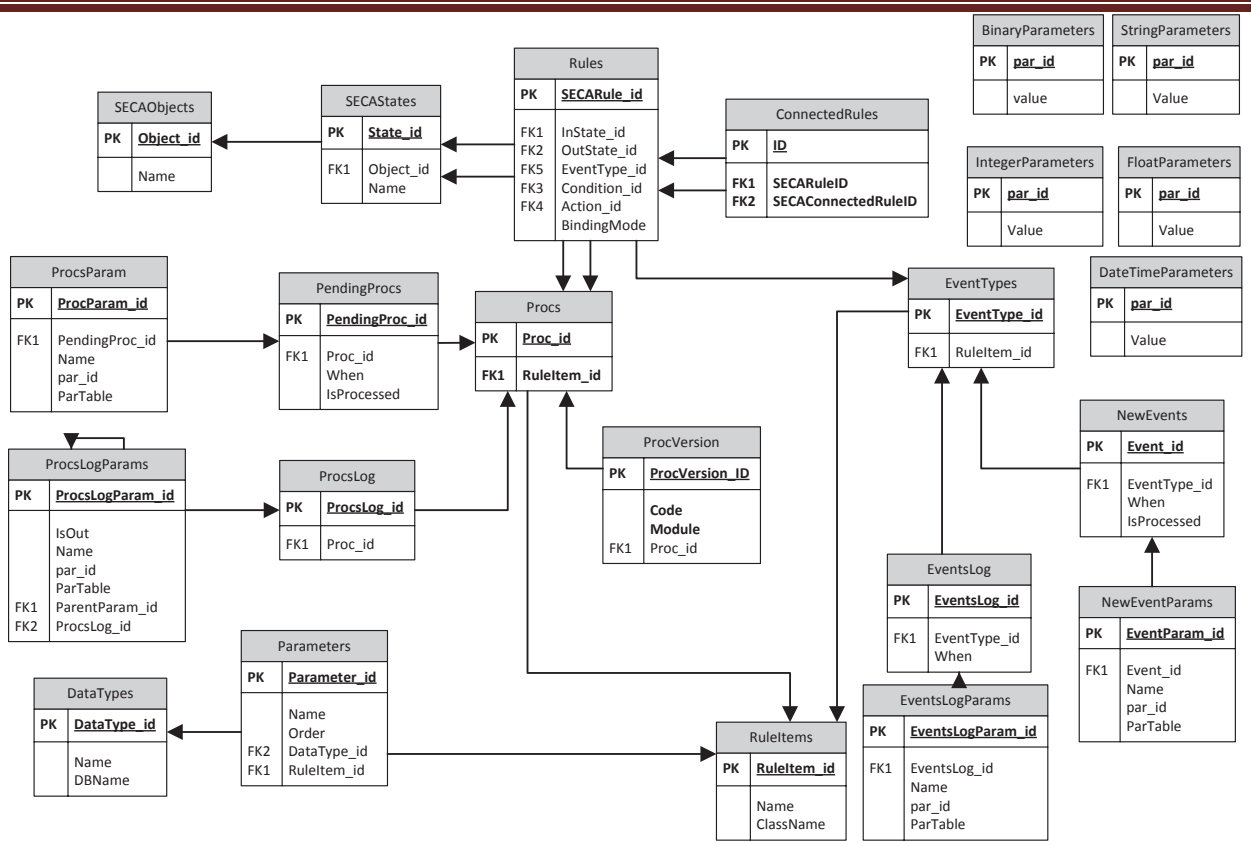

Рис. 1. Логическая схема системы метаданных для управления активными правилами

Метаданные времени исполнения предназначены для описания активных правил, их компонентов и свойств в процессе их исполнение [5]. Метаданные времени исполнения формируются на этапе динамического анализа и исполнения активных правил. К метаданным времени исполнения относятся:

– *NewEvents* – кэш событий, ожидающих обработки;
– *NewEventParams* - контекст событий, находящихся в КЭШе событий;
– *PendingProc* – кэш операций (процедур, методов) проверки условий и выполнения действий для активных правил, ожидающих выполнения;
– *ProcParam* – контекст операций (процедур, методов), находящихся в кэше операций;
– *EventsLog* – журнал обработанных событий;
– *EventsLogParams* – контекст обработанных событий;
– *ProcsLog* – журнал выполненных операций (процедур, методов) проверки условий и выполнения действий для активных правил;
– *ProcsLogParams* – журнал контекст выполненных операций (процедур, методов).

Служебные метаданные представляются классами *BinaryParameter*, *StringParameter*, *FloatParameter, IntererParameter и DateTimeParameter* и позволяют хранить значения параметров контекста согласно базовым типам данных: двоичному, строковому, с плавающей запятой, целочисленному, временному.

Схема управления событиями и активными правилами на основе системы метаданных

Конструирование активных правил в рамках системы метаданных осуществляется в полуавтоматическом режиме [5]. При этом пользователю необходимо определить (или создать, если подходящие компоненты отсутствуют) событие, по которому будет срабатывать правило, а также определить условие его выполнения и действие, выполняемое правилом. Для SECA-правил также необходимо определить первоначальное и конечное состояние объекта. После создания процедур, отвечающих за обработку условия и действия, создаются шаблоны процедур, с подставленным именем, а также подставленными параметрами.

Каждое активное правило представляется следующими экземплярами объектов:
– один экземпляр объекта класса *Rules*;

– один экземпляр объекта класса *EventTypes*;
– один экземпляр объекта класса *Procs*;
– один или несколько экземпляров объектов класса *ProcVersion*;
– ноль, один или несколько экземпляров объектов класса *ConnectedRules*;
– два или три экземпляра объектов класса *RuleItems* для представления события, условия и действия;
– ноль, один или несколько экземпляров объектов классов *Parameters* и *DataTypes*;
– ноль или один экземпляр объекта класса *SECAObjects* и ноль, один или два экземпляра объектов класса *SECAState* для *SECA*-правил.

Исполнение активных правил (рис. 2) осуществляется следующим образом [6]:

1. При возникновении нового события оно вместе с его контекстом помещается в кэш событий.

2. АБД обнаруживает в кэше новое событие и начинает его обработку: находит связанные с этим событием правила, выявляет взаимосвязи этих правил, определяет порядок их срабатывания.

3. АБД помещает в кэш операций условия и действия правил согласно порядку выполнения, определенному на предыдущем этапе

4. АБД выполняет проверку условий и выполняет действия, находящиеся в кэше операций.

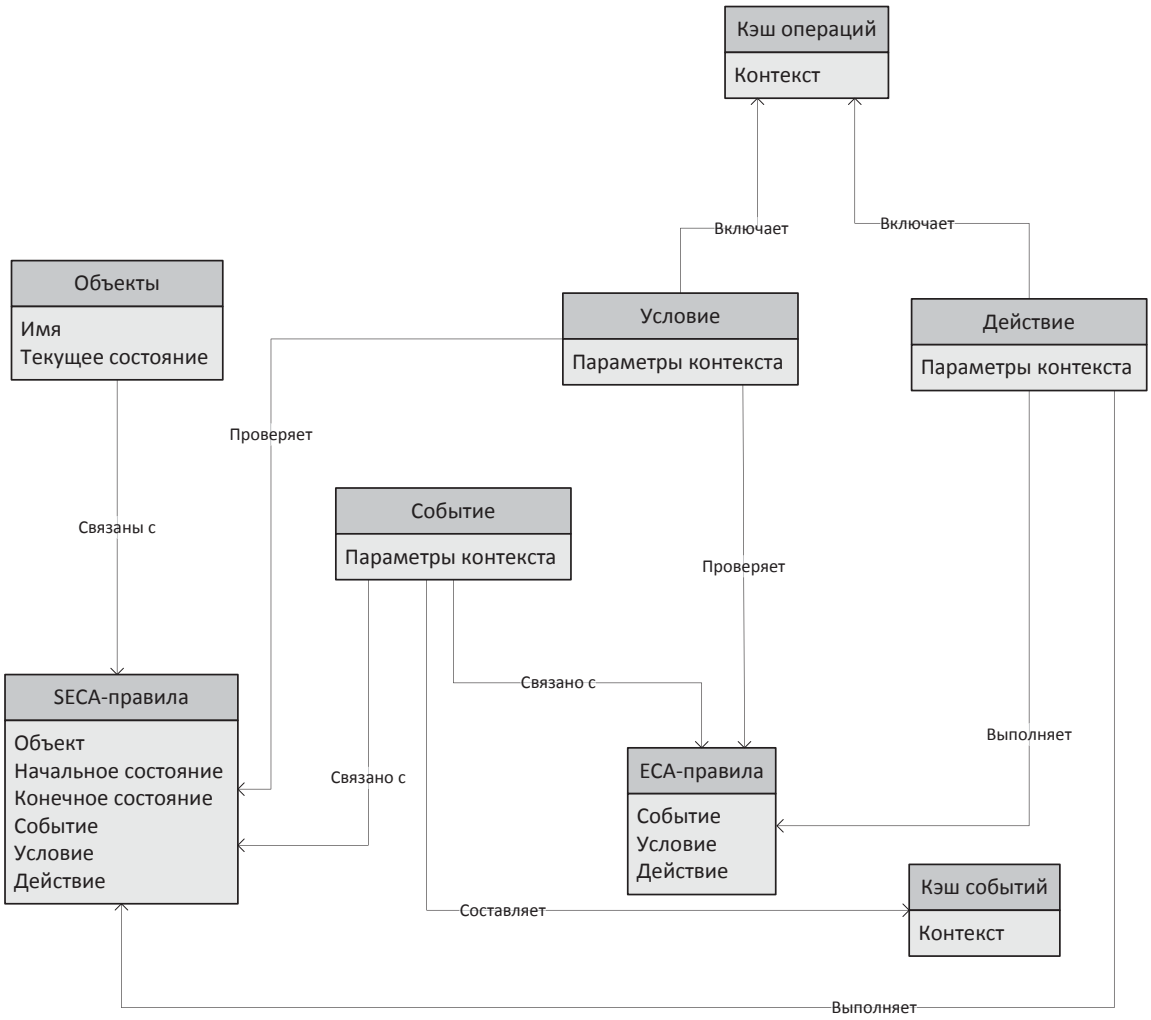

Рис. 2. Схема исполнения активных правил

Этапы 1-4 выполняются до тех пор, пока в кэше событий остаются необработанные события.

Таким образом, в ходе процесса обработки каждое событие предметной области в данной модели распадается на четыре: собственно событие предметной области и три служебных события, являющихся неотъемлемой частью механизма исполнения активных правил. К служебным событиям относятся помещение информации о событии и его контексте в кэш событий, помещение информации о вычислении условия и его контексте в кэш операций и помещение информации о выполнении действия, соотнесенного с правилом, в кэш операций. Служебные события создают дополнительную нагрузку на СУАБД, однако не связаны непосредственно с предметной областью. Эффективность их реализации напрямую зависит от режимов и механизмов управления событиями.

Режимы и механизмы исполнения активных правил

Синхронное выполнение правил с помощью встроенных средств СУБД. К встроенным программным средствам, реализующим те или иные аспекты концепции активных баз данных в СУБД, можно отнести триггеры в реляционных и функции обратного вызова в объектных системах управления базами данных.

Наиболее близким к концепции АБД является механизм триггеров в реляционных СУБД. По своей сути, триггеры являются заданием реакции на события и могут быть с событиями обработки данных – вставкой, удалением или изменением кортежей или событием установки сеанса того или иного пользователя. Триггер может срабатывать до, после или вместо вызванного события.

В реляционных СУБД, таких как *Microsoft SQL Server* [7], *Oracle* [8] или *PostgreSQL* [9], конкретные особенности триггеров, такие как возможность определения порядка срабатывания нескольких триггеров, связанных с одним событием, максимальная длина цепочки триггеров или их транзакционность.

В объектной СУБД *Cache* [10] события могут быть обнаружены с помощью функций обратного вызова, которые аналогичны триггерам в реляционных СУБД и могут быть ассоциированы с событиями создания, чтения, удаления, а также вызываться до и после сохранения данных в базу данных. Также для обнаружения событий в СУБД *Cache* может быть использован механизм обёрток (*wrappers*), который позволяет выявлять вызов методов объекта, однако это требует внесения дополнительного кода в соответствующий метод.

В этом случае для схемы управления событиями, приведенной на рис. 2, каждое из событий (как предметной области, так и служебные), описывается одним триггером (одной функцией обратного вызова), при этом триггеры или функции обратного вызова, фиксирующие изменение кэшей, должны быть максимально короткими, во избежание нецелесообразного увеличения нагрузки на БД. При этом сами кэши (кэш событий и кэш операций) будут реализованы сущностями СУБД – таблицей для реляционных СУБД (*SQL Server, Oracle*) и объектом для объектных СУБД (*Cache*).

Однако триггеры и функции обратного вызова позволяют реализовать лишь наиболее простую, последовательную обработку событий – в этом случае события обрабатываются строго последовательно без учета их взаимосвязи и без какого-либо анализа. Поэтому они не могут быть применены для обработки сложных событий, состоящих из последовательности простых событий. Кроме того, выполнение триггеров осуществляется синхронно, в рамках транзакции, приведшей к их возникновению. Таким образом, как исходное событие, так и все события, порожденные в ходе его обработки, будут выполнены в рамках одной транзакции, тогда как активные базы данных предполагают возможность обработки не только всего правила в одной транзакции, но и разделения его обработки его компонентов (события, условия и действия) на отдельные транзакции.

Асинхронное выполнение правил с помощью очередей. Более совершенным механизмом, позволяющим реализовать событийную модель управления, являются асинхронные очереди сообщений, имеющиеся в современных СУБД. Средства работы с ними, такие как *SQL Server Service Broker* [11], *Oracle Advanced Queuing* [12] и *Cache Ensemble* [13], позволяют посредством очередей сообщений обеспечить слабую асинхронную связь различных приложений, находящихся как внутри, так и вне активной базы данных.

Основными особенностями этих средств являются:
– асинхронный прием сообщений – приложение, отправившее сообщение, не ожидает, пока начнется его обработка, а продолжает работать;
– транзакционная обработка сообщений – доставка сообщения получателю и обработка сообщения получателем осуществляется в рамках одной транзакции;
– отслеживание неизменности порядка сообщений – сообщения поступают получателю в том же порядке, в котором они были отправлены;
– поддержка как схемы работы «точка-точка» (отправка сообщения от одного отправителя одному получателю) так и «писатель-подписчик» (отправка сообщения одним отправителем нескольким получателям);
– возможность группировка сообщений – отправитель сообщения может сгруппировать несколько сообщений, чтобы гарантировать их совместную обработку.

Работа с асинхронными очередями осуществляется согласно следующему алгоритму. Отправитель публикует сообщение в очереди. Средства мониторинга очереди выявляют новые сообщения и передают их подписчикам. Каждый подписчик независимо друг от друга обрабатывает сообщение и, при необходимости формирует ответ. Монитор очереди передает этот ответ отправителю. При этом как в качестве получателя, так и в качестве отправителя могут выступать не только приложения базы данных, но и клиентские приложения, работающие с базой, а также внешние службы.

При использовании асинхронных очередей кэш событий и кэш операций превращаются, соответственно, в очередь сообщений и очередь операций. При этом в очередь операций помещаются только сообщения о выполнении действий правил, тогда как вычисление условий осуществляется при обработке системой исполнения правил сообщения о возникновении события в очереди событий.

Преимуществом асинхронных очередей является их слабая связность. Источник события, очередь и обработчик события могут быть не только разными приложениями, но и размещаться на разных компьютерах, а также быть реализованными с использованием средств, наиболее подходящих для данной задачи. Так языки управления данными, такие как SQL, наиболее применимы в задачах, связанных с манипулированием данными, тогда как проверка условий или вычисление действий, определенных в правилах, может потребовать сложных вычислений, которые эффективнее реализовать с помощью традиционных объектно-ориентированных языков.

Выполнение правил с помощью CEP-систем. На наиболее высоком уровне находятся специализированные системы непрерывной обработки событий (*CEP – Continuous Event Processing*), такие как *IBM WebSphere* [14], *Microsoft StreamInsight* или *Esper* [15].

CEP-системы представляют собой сервера приложений, использующие собственные базы метаданных для описания правил и связывающие в единую событийно-управляемую систему независимые компоненты, такие как базы данных, клиентские приложения или внешние сервисы. CEP-системы полностью замыкают работу с событиями на себя, самостоятельно осуществляя выявление событий, построение иерархий событий, поиск взаимосвязей между ними, а также выполнение действий, связанных с событиями. База данных в этом случае используется как простое хранилище данных, единственной задачей которого остается контроль целостности данных и проверка прав доступа, скрытое от клиентских приложений сервером обработки событий, тогда как все метаданные, связанные с описанием правил, хранятся внутри *CEP*. Как и асинхронные очереди, *CEP*-система позволяет ослабить связность отдельных компонентов (клиентских приложений, внешних сервисов и баз данных). Однако если при использовании очередей сообщений система в целом становится децентрализованной, использование *CEP* делает ее центром всей системы.

В отличие от вышеописанных механизмов, *CEP*-системы изначально поддерживают концепцию сложных событий, т.е. событий, являющихся комбинацией нескольких простых

событий, а также позволяют самостоятельно разрешать проблемы, связанные с управлением событиями – например, выполнение событий с учетом их взаимного влияния.

Главным преимуществом *CEP*-систем является возможность выявления событий не только на уровне операций с данными (вставки, удаления, модификации), но и на уровне предметной области. Основной недостаток этих систем проистекает из их функциональности – для полноценного использования *CEP* информационная система (как база данных, так и клиентские приложения) должна быть изначально спроектирована с учетом наличия *CEP*. Поскольку возможности каждой *CEP*-системы и способы описания правил внутри нее значительно отличаются, то в общем случае не представляется возможным описать изменения, которые претерпит предложенная система метаданных при переносе внутрь *CEP*.

Заключение

Таким образом, среди рассмотренных способов, асинхронные очереди сообщений обеспечивают функционал, необходимый для работы активных баз данных. Они обладают тесной интеграцией с нижележащей СУБД, обеспечивают асинхронность работы и слабую связность компонентов. Кроме этого, в отличие от *CEP*, асинхронные очереди могут быть внедрены в уже существующие системы без необходимости их полного изменения.

Использование триггеров (функций обратного вызова) для реализации событийного управления в активной базе данных является наименее эффективным из-за недостатков, присущих этим средствам, а именно – тесной привязки к физической модели, синхронности выполнения и сильной связности, которую привносят эти средства в модель.

Список используемых источников

1. Шибанов, С.В. Концептуальные особенности систем управления активными базами данных / С.В. Шибанов, Э.В. Лысенко // Альманах современной науки и образования. – Тамбов: Грамота, 2010. – №11 (42). – Ч. 2. – С. 106-114.
2. Norman W. Paton, Oscar Diaz. Active Database Systems. ACM Computing Surveys, Vol. 31, No. 1, March 1999.
3. The Active Database Management System Manifesto: A Rulebase of ADBMS Features. A Joint Report by the ACT-NET Consortium. [Text] SIGMOD Record, Vol. 25, No. 3, September 1996.
4. Шибанов, С.В. Интегрированная модель активных правил / С.В. Шибанов, А.А. Скоробогатько, Э.В. Лысенко // Математическое и программное обеспечение систем в промышленной и социальной сферах, 2011. – № 1-1. – С. 41-47.
5. Шибанов, С.В. Реализация абстрактной модели активных баз данных средствами современных СУБД / С.В. Шибанов, Э.В. Лысенко, А.А. Скоробогатько, А.Б. Зудов, П.В. Вишняков / В кн.: Надежность и качество: Труды международ. симпозиума, т. I. - Пенза, Информац.-изд. центр Пенз. гос. ун-та, 2010. – С. 306-314.
6. Шибанов, С.В. Оперативная обработка событий в предметной области информационной системы на основе активных правил / С.В. Шибанов / Региональный молодежный форум «Открытые инновации – вклад молодежи в развитие региона». – Пенза: Изд-во ПГУ, 2013. – С. 217-221.
7. Шибанов, С.В. Реализация технологий активных баз данных в среде Microsoft SQL Server 2008 / С.В. Шибанов, Э.В. Лысенко, П.В. Вишняков // Технологии Microsoft в теории и практике программирования. Материалы конференции. – Н. Новгород: Изд-во Нижегородского гос. ун-та, 2010. – С. 55-58.
8. Шибанов, С.В. Применение методологии активных баз данных в Oracle Database 11g / С.В. Шибанов, Э.В. Лысенко, А.Б. Зудов // Технологии Microsoft в теории и практике программирования. Материалы конференции. – Н. Новгород: Изд-во Нижегородского гос. ун-та 2010. – С. 270-273.
9. Шибанов, С.В. Анализ возможности построения активных баз данных на базе Cache / С.В. Шибанов, Э.В. Лысенко, А.А. Скоробогатько // Технологии Microsoft в теории и практике программирования. Материалы конференции. – Н. Новгород: Изд-во Нижегородского гос. ун-та, 2010. – С. 267-270.
10. Официальный сайт PostrgreSQL. URL: http://www.postgresql.org/.
11. Aschenbrenner K. Pro SQL Server 2008 Service Broker [Text], Apress 1 edition, 2008.
12. Oracle Database 11g: Advanced Queuing/ Официальный сайт. URL: http://www.oracle.com/technetwork/database/features/data-integration/oracle-aq-tech-wp11-2-191324.pdf/.
13. Интеграционная платформа InterSystems Ensemble для автоматизации розничных сетей. Официальный сайт Российского филиала InterSystems. URL: http://www.intersystems.ru/ensemble/solutions-retail.pdf
14. IBM. WebSphere Application Server V8.5 Concepts, Planning, and Design Guide [Text], IBM Press, 2013.
15. Thomas Bernhardt and Alexandre Vasseur Esper: Event Stream Processing and Correlation [Text] // http://www.onjava.com/pub/a/onjava/2007/03/07/esper-event-stream-processing-and-correlation.html.

Шибанов Сергей Владимирович – кандидат технических наук, доцент кафедры «Математическое обеспечение и применение ЭВМ» ФГБОУ ВПО «Пензенский государственный университет». E-mail: serega@pnzgu.ru.

Вишняков Павел Валерьевич – соискатель кафедры «Математическое обеспечение и применение ЭВМ» ФГБОУ ВПО «Пензенский государственный университет». E-mail: vishnyakov.pavel@me.com.
Лысенко Эдуард – соискатель кафедры «Математическое обеспечение и применение ЭВМ» ФГБОУ ВПО «Пензенский государственный университет». E-mail: vishnyakov.pavel@me.com.
Смирнов Дмитрий Сергеевич – магистрант кафедры «Математическое обеспечение и применение ЭВМ» ФГБОУ ВПО «Пензенский государственный университет». E-mail: bazaar@yandex.ru.
Орешкин Константин Александрович – магистрант кафедры «Математическое обеспечение и применение ЭВМ» ФГБОУ ВПО «Пензенский государственный университет». E-mail: studyoreshkin@gmail.com.

Шибанов С.В., Вишняков П.В., Лысенко Э.В., Смирнов Д.С., Орешкин К.А. Механизмы управления событиями в активных базах данных // Математическое и программное обеспечение систем в промышленной и социальной сферах. – 2014. – №2. – С. 68-75.

Shibanov, S.V., Vishnyakov, P.V., Lysenko, E.V., Smirnov, D.S. and Oreshkin, K.V., 2014. Mathematical and algorithmic support of software for simulation dynamic systems. Software of systems in the industrial and social fields, 2: 68-75.

УДК: 621:004.942

СТРУКТУРА ИНТЕРАКТИВНОЙ СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ КОНСТРУКЦИИ СЕКЦИЙ ВТОРИЧНОГО ОХЛАЖДЕНИЯ МАШИНЫ НЕПРЕРЫВНОГО ЛИТЬЯ ЗАГОТОВОК

Сафонов Д.С., Логунова О.С.

Аннотация. В работе сформулирована проблема отсутствия специализированных средств автоматизации проектирования схемы расположения форсунок в секциях вторичного охлаждения МНЛЗ при одновременном наличии развитого математического аппарата, позволяющего использовать современные средства вычислительной техники для поиска оптимальной схемы расположения форсунок. Приведены результаты анализа требований к интерактивной системе автоматизации проектирования схемы расположения форсунок в секциях вторичного охлаждения МНЛЗ, выделены возможные сценарии взаимодействия проектировщика с системой. Разработаны структуры входных и выходных данных, а также построена структурная схема интерактивной системы. Полученные результаты являются формализацией итогов исследования предметной области и разработки высокоуровневой архитектуры интерактивной системы и содержат в себе аналитическую основу для реализации данной системы.
Ключевые слова. Проектирование машины непрерывного литья заготовок, схема расположения форсунок, системы автоматизации проектирования

STRUCTURE OF COMPUTER-AIDED DESIGN SYSTEM FOR DESIGNING CONSTRUCTION OF THE SECONDARY COOLING SECTIONS FOR CONTINUOUS CASTING MACHINE

Safonov D.S., Logunova O.S.

Abstract. The problem of absence of specific computer-aided design systems for designing nozzle layout in secondary cooling sections of the continuous casting machine in the view of existence of the mathematical models that could help in searching for optimal nozzle layout design is stated. Results of the requirements analysis for computer-aided design system for designing nozzle layout in secondary cooling sections of the continuous casting machine are presented and possible scenarios of interaction between user and system are outlined. The structure of input and output data is developed, and finally the high-level schematic diagram of the system is constructed. Obtained results constitute the analytical basis for developing such system.
Keywords. Design of continuous casting machine, nozzle layout, computer-aided design.

Введение

Современное промышленное производство использует технологические агрегаты, которые в своем составе имеют сложно-структурированные технологические узлы. Каждый из узлов вносит вклад в формирование качества производимой продукции или полуфабриката. Машина непрерывного литья заготовок (МНЛЗ) является одним агрегатов, используемых в металлургическом производстве, и на этапе непрерывной разливки закладываются предпосылки качества стального листа, балок и т.п., которые впоследствии передаются в другие отрасли народного хозяйства.

Конструкция МНЛЗ предполагает несколько основных технологических узлов, среди которых наиболее сложным является зона вторичного охлаждения (ЗВО) (*рис. 1*) [1, 3, 5].