Гибкая основа информационной системы для обучения♣

© С.В. Знаменский

Институт программных систем имени А.К. Айламазяна РАН, г. Переславль
svz@latex.pereslavl.ru

Аннотация

Пересмотр целей и содержания обучения в контексте *«один ученик – один компьютер»* нуждается в высоконадёжной информационной системе, гибко, эффективно и разносторонне организационно обеспечивающей разнообразную творческую работу в смешанных группах с опорой на электронные библиотеки. Доклад посвящён проблеме создания информационной системы с требуемыми качествами. Показывается принципиальная бесперспективность реляционных СУБД и описывается альтернативная основа.

1 Постановка задачи

1.1. Изменившиеся цели обучения

Традиционные технологии обучения создавались во времена, когда не было компьютеров, снимающих с человека тяжесть воспроизведения знаний и алгоритмических действий, и поэтому были нацелены на репродуктивную деятельность. Лекции, семинарские занятия, экзамены в основном ориентированы на приобретение и воспроизведение регламентированных стандартом знаний. Практические и лабораторные занятия, контрольные работы формируют умения и навыки решения типовых задач.

Экспоненциально растущие разнообразие и доступность специальных технологических знаний и неопределённость содержания будущей деятельности специалиста обесценивают усилия по выделению единой стандартной репродуктивной основы специального образования. Поэтому новое поколение стандартов в большей степени ориентируется на творческие компетенции, лучше отражающие (см., например, [1]) творческие способности общей природы.

Для IT-сферы к таким способностям относятся умения организовать и вести совместную разработку программной системы, документировать код и пользовательский интерфейс, взаимодействовать с пользователями и другими разработчиками. Сходные творческие компетенции требуются от специалистов и в других сферах.

Труды 12й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» – RCDL'2010, Казань, Россия, 2010

1.2 Новое содержание учебной деятельности

Поиск форм учебной деятельности, обеспечивающих формирование ключевых компетенций современного специалиста, находится в начальной стадии. Наиболее заметные находки, такие, как letopisi.ru [2], [3], сконцентрированы в программе «Intel® Обучение для будущего» и относятся к общему образованию. Все они представляют собой различные формы организации *творческой проектной деятельности на телекоммуникационной основе.*

Для подготовки специалистов это *означает участие студентов бок о бок с профессионалами в практически ценных творческих проектах*.

Это содержание в Университете города Переславля (УГП) имени А.К. Айламазяна реализуется в таких организационных формах, как учебная, производственная и преддипломная практики, курсовое и дипломное проектирование. В дополнение к классической базовой части учебный план расширен решениями учёного совета УГП практикой участия в научно-практических конференциях и практикой по разработке информационной системы УГП, ведущейся всеми студентами младших курсов специальности «Прикладная математика и информатика» под руководством немногих студентов старших курсов.

Дальнейшее развитие информационной поддержки учебного процесса в УГП направлено на поиск эффективных форм организации творческой проектной деятельности на аудиторных занятиях на основе полного перевода учебного процесса на схему *«один студент – один компьютер»*. Решением ученого совета университет с 1 сентября 2010 года полностью переходит на разрабатываемую студентами информационную систему.

1.3 Основания разработки

Творческая проектная деятельность имеет специфические сложности организации. Ниже выделены важнейшие из них для проектирования информационной системы.

1.3.1 Система оценок

Объективной абсолютной шкалы оценок результативности творческой деятельности не существует. Реально используемые относительные шкалы типа КТУ (коэффициент трудового участия – выражает долю индивидуального вклада в общем продукте) и рейтингового типа (учитывающего, чей вклад

больше, но игнорирующего, насколько больше) субъективны по своей природе и теряют адекватность при попытке перевода в пятибалльную систему.

По-видимому, проблема оценивания результатов творческой деятельности в полной общности неразрешима и должна разрешаться ситуативно. Это означает, что сохраняться и использоваться в системе должны не пятибалльные оценки, а содержательный отчёт, рецензии, КТУ, рейтинг и другие осмысленные характеристики.

1.3.2 Неограниченное разнообразие организационных форм

Индивидуальные творческие задания и задания малым однородным группам редко имеют столь яркий обучающий, воспитательно-ориентирующий и продуктивный эффект, как работа в продуктивном творческом коллективе с непростым уникально-специфическим распределением ролей и обязанностей. Информационная система должна обеспечить такие коллективы набором интерфейсов, удобных для сохранения промежуточной информации, разборов полёта, оценок и взаимооценок активности участников. Всего этого невозможно заготовить заранее, система должна постоянно дорабатываться.

1.3.3 Сопровождение и развитие системы – тоже проект

Сопровождение и развитие системы – это один из учебных проектов, и поэтому он должен быть полноценно поддержан в этой системе. Требуется обеспечить гибкую организацию хорошо защищённого доступа и авторизацию изменений к коду и данным,

2 Границы применимости реляционных информационных систем в задаче поддержки сложного непредсказуемо изменчивого взаимодействия

За десятилетия развития и интенсивного использования реляционные СУБД заслуженно заняли место безальтернативной основы для создания сложных информационных систем. Возможность абстрагирования от временной рассогласованности происходящих в реальности изменений (например, от того, что продавец либо сначала пробивает чек, а затем принимает деньги, либо наоборот, но никогда не одновременно) резко облегчила разработку, позволяя в полной мере опереться на логику предикатов. Разнообразные средства разработки ориентированы на систему с фиксированной (в принципе, даже подменяемой в следующей версии кода, но только не в процессе обработки данных) онтологией, стандарты представления и обмена данных, многочисленные кроссплатформенные высококачественные реализации СУБД и наличие большого числа свободно доступных модулей, учебники и другие ресурсы, созданные лучшими умами в последние десятилетия, — всё это придаёт технологиям реляционных СУБД ореол безальтернативности реализации качественной информационной системы.

Чтобы увидеть дальше этого ореола, давайте рассмотрим реляционную технологию с позиций математического моделирования.

2.1 Реляционная модель информационных процессов

Реляционная модель информационных процессов характеризуется следующими характеристиками:

1. В каждый момент система находится в согласованном (логически непротиворечивом) состоянии.
2. Изменения в данных вносятся исключительно через механизм транзакций [4], обладающих свойствами ACID = (atomicity, consistency, isolation, durability) и обеспечивающий логическую целостность данных системы в любой момент времени.
3. Единая и неизменная на момент осуществления транзакции онтология системы в целом формально описывает логические структуры данных, что позволяет контролировать целостность данных и планировать транзакции.
4. Внутреннее представление данных и язык запросов (SQL, XQuery,...) корректно реализуют логику исчисления предикатов.

Таким образом, реляционная модель – это описание и обработка информации, базирующиеся на исчислении предикатов.

Поддержание большой сложной системы в согласованном состоянии затратно. Ситуацию кардинально не меняет даже огромное число последующих работ (см., например, [5 – 7]), направленных на оптимизацию планирования параллельной обработки транзакций.

Обычно только учёт очевидных априорных предположений о структуре и типах возможных запросов позволяет резко повысить производительность системы. Любое отклонение от перечисленных характеристик усложняет систему. Простейшим вариантом такого отклонения является широко используемая денормализация – разделение однородной таблицы на изолированные части. Повышая вероятность ошибки при обработке непредусмотренного запроса с непредсказуемыми последствиями, такое отклонение угрожает лишить систему главного козыря – полноценной опоры на исчисление предикатов – и угрожает логической целостности данных. Оставаясь иллюзорной и практически неправдоподобной для систем, в которых данные неизменно разделены на независимо изменяемые части, эта угроза становится реальной для сложных целостных систем.

2.2 Разделение данных на изолированные части

Сложная неоднородная система может создаваться только во взаимодействии многих групп разработчиков, и поэтому разумная организация их рабочего взаимодействия также является сложной

адаптивной системой, нуждающейся в адекватной информационной поддержке.

Главная особенность постановки задачи состоит в том [8 – 10], что предметная область для требуемой информационной системы по сути относится к достаточно интенсивно исследуемому за рубежом классу социотехнических *сложных адаптивных систем (complex adaptive systems*). Имеются как исследования особенностей обработки информации для таких систем [11 – 15], так и сопоставительный анализ различных подходов к их разработке [16 – 20], влекущих новое представление о жизненном цикле такой информационной системы [21, 22]. Общим в этих работах является безусловное признание необходимости разделения системы на множество по сути независимо создаваемых и модифицируемых частей, заранее не регламентированных ограничениями дальнейшего развития и взаимодействия. Концептуальная модель такой обработки информации оказывается заведомо сложной и требующей специальных подходов к разработке. Гибкостью и универсальностью выделяется механизм контекстуализации [23, 24], позволяющий с учётом разнообразных специфических особенностей организовывать инкапсуляцию и наследование.

Эта идея нашла общепризнанный стандартный путь реализации [19, 20, 25 – 27] в рамках сервер-ориентированной архитектуры (SOA), нацеленной на объединение независимых сервисов.

Изоляция информационных процессов кардинально повышает быстродействие реляционной СУБД и помогает локализовать временные потери доступа при внесении изменений в онтологию, но сопряжена с серьёзными потерями качества пользовательского интерфейса.

2.2.2 Проблема совместного редактирования

Совместный доступ к изменяемой информации чреват коллизиями. Пользователь может записать свои изменения поверх тех, которые были сохранены в момент его редактирования и которых он не мог видеть. Новая технология Google wave разделяет процессы редактирования данных разными пользователями, позволяет объединять изменения в ходе работы и поддерживает индивидуальные многократные undo/redo. Недостатком этой, основанной на операционных преобразованиях, новейшей технологии является принципиальная невозможность добавления в редактор интерфейса выделения и перемещения блоков. Этот недостаток принципиально неустраним и крайне существенен: операция перемещения является важнейшей в редактировании структур, и удовлетворительного решения для такого редактирования на пути операционных преобразований не предвидится.

Задача выделения и оценки вклада каждого пользователя, по-видимому, должна решаться более адекватными средствами, как минимум учитывающими, что перестановка частей текста не тождественна написанию их с нуля. Желательно переработать алгоритм diff так, чтобы перестановка больших кусков текста показывалась именно как перестановка, а не как удаление и вставка.

Преодоление разделённости интерфейсов пользователей открыло бы возможности поиска простой эффективной комфортной организации совместного редактирования текстов и структур, поскольку для избегания коллизий при совместном редактировании достаточно показывать пользователю положения курсора и выделения других активных пользователей. Имеющие общность цели и не заинтересованные во взаимных помехах пользователи смогут при этом вносить неперекрывающиеся изменения независимо, может даже вплоть до того, что абзацы текста, правящиеся двумя пользователями, могут быть в это время переставлены третьим.

2.2.3 Проблема перестройки начавшихся процессов

Система рассматриваемого класса должна иметь функциональность WorkFlow, позволяющую создавать сложные схемы взаимодействия субъектов и запускать процессы многоступенчатого рассмотрения – согласования – утверждения и другие бизнес-процессы, работающие по заданной схеме. При ориентации разработки на разделение данных каждый такой процесс, точнее, информационная составляющая каждого такого бизнес-процесса, работает в своём пространстве состояний, изменения в которое можно в лучшем случае вносить индивидуально в каждый процесс. Если изменение условий требует немедленной перестройки начавшихся процессов, то система с разделёнными данными становится крайне неудобной. Перестройка процессов как необходимая функциональность не просто востребована, она прописана в стандартах контроля качества. Изоляция начатых процессов либо делает её невозможной, либо нерационально усложняет логику системы.

Эта проблема решена в системах управления задачами на основе единого для всех процессов и неизменного множества состояний. При таком подходе нет изоляции процессов, но жёсткая унификация процессов не всегда адекватна требованиям практики.

Таким образом, оба этих близких по назначению подхода принципиально ограничены заложенными изначально принципиальными ограничениями, лишающими пользователя возможности совместить их преимущества.

2.2.4 Проблема нескольких окон

Современные браузеры предоставляют пользователю удобную возможность открыть несколько окон и расположить их в удобном для работы порядке. Информационные системы традиционно лишают его этой возможности на том основании, что информация в открытых окнах устаревает и работа с несколькими открытыми окнами может привести к потере данных. Если, например, пользователь открыл длинную форму в двух окошках, начал запол-

нять в обоих, в одном заполнил, сохранил и закрыл, а в другом ошибочно сохранил, не глядя и помня, что он её уже заполнял, то основная работа по заполнению формы безвозвратно пропала. Современные AJAX-технологии в принципе позволяют перенести изменения в другое окно, даже если оно открыто в другом браузере или на другом компьютере, но здесь возникает тот же барьер изолированности процессов обработки сессий.

2.2.5 Проблема межпроцессной коммуникации

Механизм обмена сообщениями (e-mail, системные сообщения между изолированными процессами), на котором базируются современные коллаборативные системы, имеет несколько принципиальных недостатков:

1. *Крайне трудно освободить пользователя от удаления сообщений, по тем или иным причинам потерявших актуальность.* Это легко было бы сделать, если бы сообщения не отсылались, а генерировались в момент, когда пользователю доступен соответствующий интерфейс. Но как сделать, чтобы иконка уведомления о важном сообщении исчезала, когда сообщение потеряло актуальность? Если логика отправки сообщений хорошо формализована, то теоретически можно вычислить и посылать сообщения, отменяющие ранее посланные. Однако такой подход недоступно сложен в реализации и отладке.

2. *Если у пользователя изменились статус или настройки или другие параметры приняли значения, при которых сообщение должно бы было присутствовать, то сообщение должно вернуться, но оно удалено*. Чтобы оно появилось, система теоретически могла бы сгенерировать аналогичное сообщение взамен удалённого, но реализация такого поведения крайне сложна, и любое изменение в организации сообщений в таком случае должно требовать существенной перестройки механизмов генерации.

Чтобы пользовательские интерфейсы безошибочно показывали актуальную информацию на текущий момент во всех случаях, механизм сообщений должен быть заменён на процесс сбора и обработки текущей информации из различных подсистем. Такой подход несравненно проще в безупречной реализации, но несовместим с изоляцией контекстов данных.

2.2.6 Неизбежность изоляции данных информационных процессов

Неукоснительное требование полной согласованности всех данных большой сложной целостной системы резко усложняет внесение малейшего изменения просто потому, что предполагает немедленную диагностику необходимости внесения сопутствующих изменений в остальные данные системы и внесение этих изменений. Изоляция информационных процессов является основным путём резкого снижения этого бремени и повышения производительности системы в целом до приемлемого уровня. Выше показано, как эта изоляция отзывается принципиально неодолимой недружелюбностью пользовательских интерфейсов на уровне конкретных приложений.

Действительно ли необходимость изоляции информационных процессов является общей особенностью информационных систем? Либо она связана с опорой на переставшую быть адекватной новым постановкам задач реляционную модель обработки данных, и другая модель обработки информации сможет просто разрешить перечисленные проблемы для рассмотренного класса задач?

2.3 Обеспечение надёжности

Реляционная модель обоснованно считается наиболее эффективным средством разработки высоконадёжных информационных систем. Практически все современные реляционные СУБД включают в себя средства резервного копирования (в том числе горячего) и репликации данных, дающие администратору возможность быстро вернуться к сохранённой копии или переключить работу на резервный сервер.

Онтология сложной информационной системы должна быть доступна для внесения изменений, потребность в которых проясняется в ходе эксплуатации. Некоторые проблемы обеспечения надёжности при многопользовательском доступе к данным и пути их решения описаны, например, в [4, 28, 29].

Кодирование и отладка в работающей реляционной системе – это обычно крайне рискованная деятельность, требующая нескольких тестовых серверов, разделения кода и пользовательских данных, сложной организации хранения резервных копий кода и данных.

Для обеспечения её безопасности полезно полное сохранение всей истории изменений с их авторством и удобный доступ к прошлым состояниям сервисов, заложенные в СУБД на уровне, неподконтрольном пользователям-разработчикам.

2.3.1 Проблема репликации сервиса с целью повышения безотказности и отзывчивости

Обычным средством достижения высокой отказоустойчивости и реактивности сервиса является его дублирование. Для медленно изменяющейся информации этот способ широко используется в поисковых серверах и других web2.0-приложениях [30 – 32], наглядно показывая [33], что дублирование информации на многих серверах ускоряет обработку стандартных запросов на порядки ценой снижения согласованности информации на выходе и замедления реакции на изменения в данных. Например, результаты поиска в поисковых серверах зачастую отражают не сиюминутное, а прошлое состояние страничек, а сведения о количестве найденных ссылок могут многократно измениться по мере их просмотра.

Общеизвестно, что распределённость информационной системы с дублированием информации по-

тенциально способно кардинально повысить её жизнестойкость. Однако проблема внесения согласованных изменений в распределённо дублированную информацию не находит прозрачных решений, пригодных для большой целостной (не разделяемой на изолированные процессы) реляционной системы.

2.3.2 Проблема сохранности доступа к старым версиям при обновлениях

Проблема поддержки историчности в базах данных не без оснований считается решённой и детально описанной в монографии [34]. Последние исследования адресованы в основном проблемам представления исторической информации в XML [35 – 38] и проблеме повышения эффективности запросов в этих рамках [39, 40].

В рассматриваемой ситуации важно, чтобы при согласованных изменениях в данных и коде данные в запросах обрабатывались соответствующим кодом. Обеспечить согласованность версий кода и данных в реляционной базе достаточно сложно. Сложность в том, что модуль может включаться в работу в различное время для различных контекстов данных. Чтобы старые данные всегда корректно интерпретировались, предоставляющая авторизованный доступ к сохранению и востребованию версий кода и данных, подсистема контроля версий должна лежать на более низком логическом уровне архитектуры, чем разделение данных. Например, слияние или разделение учебных групп должно приводить к слиянию или разделению соответствующих таблиц, и эти действия нарушают (либо немыслимо усложняют) логику запросов к истории.

Переход от старой версии системы к новой требует внесения сложно согласованных изменений в структуры данных. Модификация может потребовать значительного времени. В промежуточном состоянии система не будет доступна пользователям. Перевод одной сложной внутренне согласованной системы в сложную систему с другой онтологией всегда является сложнейшей задачей, решение которой сопряжено с потерями второстепенного доступа.

При многократных обновлениях в результате неизбежна утрата доступа к временно забытым данным.

2.3.3 Проблема системных сбоев и исправления последствий ошибок в коде и администрировании

Полностью взять рутинную работу по полноценному воспроизведению ранее достигнутого доступа на себя, освободив разработчиков и администраторов для конструктивной деятельности, реляционная система вряд ли сможет в принципе. Но именно это было бы крайне ценно не только для разработчиков и администраторов, но и для обычных пользователей, страдающих от потери старого привычного доступа к давно не востребовавшейся информации после нескольких обновлений информационной системы.

2.3.4 Проблема непрерывного гладкого обновления системы

Частые обновления системы не должны доставлять помех пользователям. Сложность реализации этого требования также резко возрастает при усложнении системы. В реляционной модели переход от одной онтологии к другой – это всегда переход к новому качеству, и провести этот переход без приостановки сервиса для перезагрузки процессов нереально. Время на перезагрузку сложной системы непременно скажется на пользователе. Откат изменений потребует большего времени. Гладкость обновления означает, что система должна гарантировать доступность интерфейсов ввода и вводившейся пользователем информации независимо от установки и отката обновлений.

2.4 Цель статьи

Выше перечислен ряд проблем разработки качественной информационной системы. Некоторые из них предположительно могут быть решены и в рамках реляционной модели ценой значительного усложнения реализации. Совмещение этих требований в традиционных рамках реляционной модели приводит как минимум к немыслимому усложнению системы. В интернете не удаётся найти описаний систем, в которых качественно решена хотя бы часть этих проблем. Отсюда вывод, что решение этих проблем в комплексе лежит за гранью возможностей систем, основанных на реляционной модели.

Ниже приводится набросок модели информационных процессов, не нуждающейся в изоляции процессов, допускающей ничем не ограниченную дальнейшая разработку работающей системы при поддержке самой системы, ни на секунду не лишающей пользователей доступа к прошлой информации и возможностей сохранить изменения в ней даже при единичных случаях гибели системных процессов, ошибок разработчиков и администраторов. Все сформулированные проблемы могли бы решаться в ней вполне прозрачно и в комплексе. Это означало бы, что реляционная модель не всегда является адекватным средством и требует замены для практически важного класса информационных систем.

Проверка этой гипотезы невозможна без принятия её за основу, проработки альтернативной модели и доведения её до полнофункционального состояния.

3 Принципы асинхронной контекстно-автономной модели управления данными для сложных адаптивных систем

Изложенные выводы иллюстрируют гипотетичную бесперспективность реляционных СУБД в решении сложных задач организации совместной деятельности. Выглядит очевидной неадекватность

рассматриваемому классу задач реляционной модели,

- опирающейся на неизменную в процессе обработки запросов онтологию системы;
- требующей поддержания сверхзатратного для сложных систем согласованного состояния системы в целом;
- обещающей недостижимую в реальности возможность глобальной обработки запросов произвольной сложности посредством ACID транзакций;
- на практике эклектично разделяющей сложные системы на взаимодействующие части.

Отказ от реляционной модели в высшей степени болезнен и требует усилий, многократно превышающих затраты на создание системы сходной функциональности на традиционных технологиях, поскольку именно на реляционную модель ориентированы практически все существующие стандарты построения информационных систем, подсистемы и пользовательские интерфейсы, мощные средства разработки и обширная учебная литература. Поэтому такой отказ должен опираться на *альтернативную достаточно общую модель* с глубоко проработанными основаниями, гарантированно более адекватную в широкой прикладной области.

В качестве такой области в статье рассмотрена организация совместной целенаправленной деятельности, а в основу модели заложены такие качества, которые существенны для успешности организации совместной деятельности и важность которых общепризнанно подтверждена широкой практикой.

Такие общепринятые основы организации совместной деятельности отражены в стандартах документооборота и управления качеством и в описаниях корпоративных информационных систем. Хотя такие источники не содержат описания модели управления данными, но содержащиеся в них ключевые рекомендации ниже интерпретированы как базовые принципы асинхронной контекстно-автономной модели управления данными и соответствующей архитектуры ИС.

3.1 Процессный подход к управлению качеством

Базовым классом системы является информационный процесс в широком понимании. Это могут быть процессы совместной подготовки и принятия решения, поддержания в актуальном состоянии информационного ресурса или его части, уточнения пользовательского профиля и настроек пользовательских интерфейсов, выполнения сложных расчётов или иной обработки информации. Сопровождение, переработка процесса и создание новых сценариев осуществляются на языке высокого уровня в рамках самостоятельных процессов.

У процесса предполагаются данные, которые могут изменяться этим процессом и читаться им и другими процессами и приведение которых в адекватное текущей ситуации состояние является основной целью процесса.

У процесса есть хозяин – человек, который несёт персональную ответственность за осуществление этой цели.

Цель может разделяться на этапы, подзадачи, подцели и т. п., для реализации которых порождаются дочерние процессы. Поэтому каждый процесс кроме универсального процесса общего функционирования системы имеет *единственного родителя*, который иногда может измениться.

Дерево процессов может расширяться по установленным сценариям. Каждый сценарий содержится в данных процесса сопровождения этого сценария (*мастер-процесса*), которые в свою очередь могут строиться по сценарию, определённому чаще в другом *мастер-процессе*. Указатель на корень сценария берётся в данных корня ветки растущего по сценарию процесса либо в данных ближайшего предка этого корня, где он прописан. Расширяемый по сценарию процесс обязательно имеет в данных строку, идентифицирующую статус, определяющий наследуемые из мастер-процесса код и данные. Процесс может совпасть со своим мастер-процессом или мастер-процессом своего мастер-процесса. Процесс может не иметь собственных данных, опираясь на данные родительских мастер-процессов. Так изначально устроены многие подпроцессы, например, подпроцесс сбора пожеланий по улучшению, имеющийся в каждом процессе.

Если наследуемого данного не прописано в данных самого процесса и для заданной статусной строки в мастер-процессе, то это данное берётся из (пра-)родительского процесса.

Все процессы подразделяются на самостоятельные (не наследующие данных из родительских процессов) и подчинённые (не имеющие данных, которые могли бы наследоваться). Любой подчинённый процесс легко делается самостоятельным. Возможность лишения самостоятельности потребует усложнения системы, которое сможет при необходимости быть добавлено в ходе эксплуатации без приостановки сервиса.

Данные самостоятельного процесса и подчинённых ему дочерних и (пра-)внучатых процессов образуют общий ***автономный контекст*** данных этих процессов.

Процесс, выполнивший свою цель, засыпает, и система забывает о его существовании. Спящий процесс, данными которого интересуются, пробуждается при наличии изменений в ранее использованных им данных этого и других процессов и свободных ресурсов системы. Оценка наличия свободных ресурсов учитывает приоритет процесса, установленный разработчиком или администратором.

3.2 Пользовательский контроль произошедших изменений

Заходя в систему, пользователь видит не только текущее состояние, но персонализованную индикацию изменений, произошедших с окончания предыдущей сессии.

Элементы информации на экране пользователя снабжены иконками-индикаторами наличия и качества изменений с возможностями просмотра истории изменений и перехода на время любого из изменений или иное время.

Два последних времени перехода образуют промежуток, наличие и характер изменений за который показаны на экране.

Если одно из них текущее, то происходящие в системе изменения немедленно отражаются на экране.

3.3 Эффективный доступ к истории изменений

Эффективность доступа к истории изменений базируется на том, что любое отображение информации на экране пользователя – это отображение сохранённой и впредь неизменной страницы истории изменения данных в системе.

Для упрощения анализа изменений и снижения затрат на выдачу информации:

- информация хранится маленькими фрагментами;
- каждое изменение любого элемента или шаблона фрагмента фиксируется отдельной записью в базе;
- при чтении значения шаблонов фрагментов и данных на запрошенный пользователем момент времени рекурсивно подставляются;
- быстрый доступ к состоянию элементарного данного на момент времени и к истории изменений гарантируется хранением записей в глобальной базе B+tree [41, 42];
- индексация данных для поиска организуется на тех же принципах, так что результаты поиска всегда отражают состояние индексов на момент времени, указанный в запросе.

4 Архитектура системы

В серверной части системы работают два независимых базовых процесса, каждый из которых поддерживается многими процессами ОС.

Процесс взаимодействия с пользователями обеспечивает:

- выдачу любой предусмотренной информации на любой заданный пользователем момент времени;
- индикацию изменений, произошедших за истекший период;
- оперативное обновление отображаемой информации;
- сохранение полученных от клиентов данных без обработки.

Процесс обработки информации обеспечивает:

- контроль уровня загруженности системы;
- переработку полученных изменений информации в оптимизированные для быстрого чтения базы данных;
- пометку процессов с изменившимися входными данными и процессов, выходные данные которых используются;
- выделение процессов, первоочерёдно требующих обработки данных, и запуск обработки.

Появление, исчезновение и любое изменение информации фиксируются в базе типа B+tree отдельной записью, отражающей значение созданного или изменённого атрибута или функции. Ключ такой записи содержит по порядку:

- идентификатор (возможно, составной) процесса, характеристики (либо данного) процесса;
- идентификатор (возможно, составной) объекта либо список аргументов функции в порядке убывания критичности обновлений;
- идентификатор имени атрибута или функции;
- возможно дополнительно текстовая константа;
- разделитель;
- список моментов времени, последним из которых является время модификации записи; время представляется в лексикографически монотонной кодировке, а моменты в типичном случае являются концами промежутка времени, в котором производились изменения;
- разделитель;
- байт реальности данных, позволяющий выделить ссылки на записи и отладочные данные;
- идентификатор пользователя.

Начало ключа записи до первого разделителя – это логический ключ данного. Кроме логического ключа, ключи записей должны содержать «дату и подписи», т. е. штамп времени и идентификатор автора изменения.

Дополнительное включение в состав ключа записи байта реальности данных обеспечивает возможность незаметного для обычных пользователей тестирования обновляемых частей системы. Перед картиной реальных данных как бы встаёт невидимый для обычных пользователей слой тестовых изменений в реальных данных, позволяющий полноценно выверить и отладить новый код и включить его в работу с реальными данными без приостановки основного сервиса.

Значением записи может быть произвольный текст, в котором может быть сериализована произвольная структура. Для повышения эффективности хранения и обработки изменений большие структуры и тексты должны разбиваться на отдельно хранящиеся составляющие. Например, перестановка абзацев в тексте, совместно редактируемом пользователями, должна сохраняться и показываться пользователю как перестройки списка, а сами абзацы должны показываться неизменными.

Наравне с производственными процессами требуется информационно обеспечить **процесс балансировки обновлений системы**, данными которого являются исполняемый код, информация об уровне текущей загруженности системы, востребованности

обработки различных контекстов в данный момент, наличии необработанных изменений и т. д. Для этого в каждом *автономном контексте* поддерживаются:

- список пар (логический ключ данного, идентификатор использовавшего его контекста) обновляется при чтении данных из других контекстов;
- функция проверки доступности пользователю данных контекста, возвращающая процедуру определения доступности данного по логическому ключу данного и категории пользователя и при необходимости расширяющая этот список;
- функция определения категории пользователя (при изменении в ролях пользователя в автономном контексте или иных влияющих на доступ событиях его категория перевычисляется) и её текущие значения для пользователей с особыми правами;
- момент последнего изменения данных контекста (модифицируется при записи любых данных);
- момент последнего изменения использовавшихся в контексте данных других контекстов (модифицируется при записи любых данных).

Такая основа принципиально позволяет на фоне достаточно изолированных человеческих ошибок, программных сбоев и поломок оборудования гарантировать:

- неизменность и быструю доступность (с учётом авторизации) всей имеющейся информации;
- высокую сохранность информации и полную воспроизводимость прежнего доступа;
- возможность организации наследования данных, инкапсуляции изменений, каскадных стилей, политик и настроек;
- возможность безопасной организации отладки любых изменений кода в работающей системе (например, миграции) без предоставления разработчикам доступа к секретным данным;
- возможность организации автоматического оптимального выделения и пометки для последующего физического удаления наименее ценных давно устаревших записей.

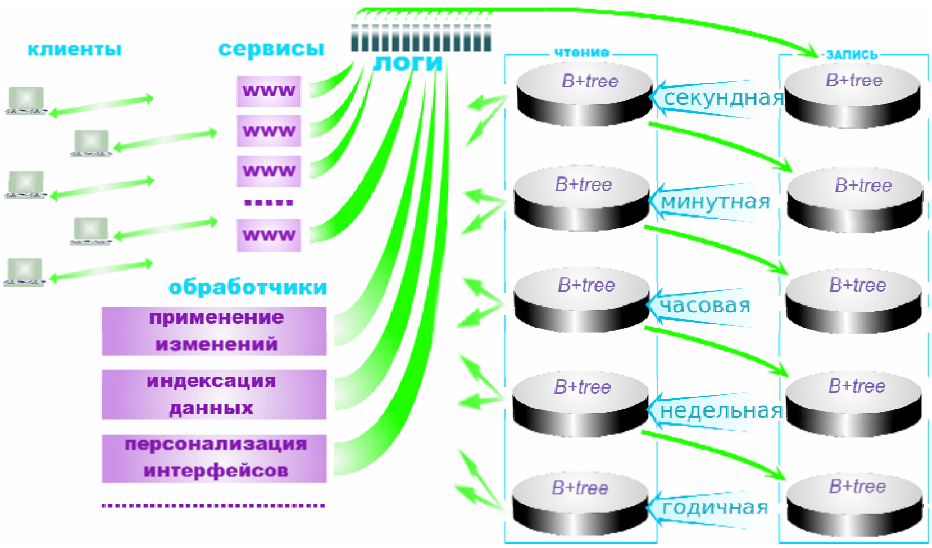

Рис. 1. Схема потоков информации

4 Производительность и проблема параллельной обработки

Узким местом масштабируемости СУБД является эффективная параллельная обработка. Наглядным частным случаем является проблема поддержания B+tree-базы данных. Проблем нет, если чтение и запись разделены по времени, но такое разделение сопряжено с блокировками и задержками. В документации к Berkeley DB 4 было написано, что она поддерживает режим модификации одним процессом одновременно с чтением другими. При этом блокировки доступа организуются на уровне страниц памяти, копии которых согласованно подготавливаются и периодически синхронно пишутся на диск.

В случае большой базы вероятность конфликта на уровне страниц памяти на порядки ниже вероятности конфликта на уровне файла. Однако такой подход сопряжён с высокой сложностью своевременной диагностики возможности возникновения конфликтов. Трудно предсказуемые конфликты возникают при балансировке дерева и асинхронной записи. В результате запрос на чтение может вернуть результат, отличный от того, который был перед началом записи и от того, который стал после её окончания. О масштабах сложности проблемы говорит уже тот факт, что фирма Oracle отказалась от поддержки такой функциональности в своей BerkeleyDB.

Сейчас новый вызов проблеме брошен в описании системы Kyoto Cabinet фирмой FAL Labs, производителем популярной СУБД Tokyo Cabinet.

Если он не окажется успешным, то масштабируемость описанной системы останется принципиально ограниченной предельной скоростью обновления B+Tree-базы, имеющей сегодня порядок миллиона записей в секунду.

Описанные принципы позволяют использовать одновременно несколько B+Tree-баз, оптимизированных на чтение и неизменных в течение периодов различной продолжительности. Базы большего объёма меняются редко, самые маленькие базы обновляются несколько раз в секунду, предоставляя доступ к последним изменениям данных.

Это позволяет обеспечить неразделяемую информационную систему высоко реактивным пользовательским интерфейсом и гибко модифицируемой логикой обработок растущей сложности [41, 42].

5 Состояние реализации

На текущий момент прорабатывается и кодируется прототип системы, который будет использоваться для поддержки учебного процесса в УГП имени А.К. Айламазяна.

Литература

[1] Борисов Н.В., Чугунов А.В. Стратегия развития информационного общества и задачи подготовки кадров в области информационно-коммуникационных технологий // Труды XVI Всерос. науч.-метод. конф. Телематика'2009 – 2009. – Т. 1, Секция А. – С. 171-172. – http://tm.ifmo.ru/tm2009/src/232a.pdf.

[2] Патаракин Е.Д., Ярмахов Б.Б. Веб 2.0 – управление, изучение и копирование// Educational Technology & Society. – 2007. – Т. 10, № 2. – http://ifets.ieee.org/russian/depository/v10_i2/ html/2.htm.

[3] Патаракин Е.Д. Сеть детских конструкторов// Большой московский семинар по методике раннего обучения информатике. Публикации семинара к 9 марта 2010 г. – http://ito.edu.ru/sp/SP/SP-0-2010_03_09.html.

[4] Weikum G., Vossen G. Transactional information systems: theory, algorithms, and the practice of concurrency control and recovery. – San Francisco: Morgan Kaufmann Publishers, 2001.

[5] Fekete A. Allocating isolation levels to transactions// Proc. of PODS. – 2005. – P. 206-215.

[6] Saito Y., Shapiro M. Optimistic replication// ACM Computing Surveys. – 2005. – V. 37, No 1. – P. 42-81.

[7] Alrifai M., Dolog P., Balke W.-T., Nejdl W. Distributed management of concurrent web service transactions// IEEE Transactions on Services Computing (TSC). – 2009. – V. 2, No 4. – P. 289-302.

[8] Zhao H., Zhang Y., Wang Z., Lee S.F., Kwong W.C. Research on group decision support system for concurrent product development process// J. of Material Processing Technology. – 2003. – V. 139. – P. 619-623.

[9] Govindu R., Chinnam R. B. MASCF: a generic process-centered methodological framework for analysis and design of multi-agent supply chain systems// Computers & Industrial Engineering. – 2007. – V. 53. – P. 584-609.

[10] Asproth V. Inter-organizational management and decision-making// Systemist. – 2006. – V. 28, No 2. – P. 4-12.

[11] «Intel® Обучение для будущего». Официальный сайт программы. – http://www.iteach.ru/abo/.

[12] Mukherjee I. The complexity paradigm: implications for information systems and their strategic planning// J. of Computer Science. – 2008. – V. 4, No 5. – P. 382-392.

[13] Mukherjee I. Understanding information system failures from the complexity perspective// J. of Social Sciences. – 2008. – V. 4, No 4. – P. 308-319.

[14] Tonchia S., Cozzi F. Industrial project management: planning, design, and construction. – Springer, 2008. – 230 p.

[15] Choi J.K., Nies L.F., Ramani K. A framework for the integration of environmental and business aspects toward sustainable product development// J. of Engineering Design. – 2008. – V. 19, No 5. – P. 431-446.

[16] Sünbül A., Weber H., Padberg J. Evolutionary development of business process centered architectures using component technologies// J. of Integrated Design and Process Science. – 2001. – V. 5, No 3. – P. 13-24.

[17] Wang X., Conboy K. Understanding agility in software development from a complex adaptive systems perspective// 17th European Conf. on Information Systems, 2009.

[18] Meso P., Jain R. Agile software development: adaptive systems principles and best practices// Information Systems Management. – 2006. – V. 23, No 3. – P. 19-30.

[19] Yeh-Chun Juan, Chao Ou-Yang, Jiun-Shiung Lin. A process-oriented multi-agent system development approach to support the cooperation-activities of concurrent new product development//Computers and Industrial Engineering. – 2009. – V. 57, No 4.

[20] Chih-Hsing Chu, Han-Chung Cheng. Business model innovation based on collaborative product development: a case study of Taiwan design services// Int. J. of Electronic Business Management. – 2005. – V. 3, No 4. – P. 257-269.

[21] Daneke G., Dooley K. From cycles to kaleidoscopes: mapping the nonlinear dynamics of technology evolution// INFORMS Conf., Denver, 2004.

[22] Daneke G., Dooley K. The life-cycle revisited: stage transitions and the failure of the Iridium project// PICMET Conf., Portland, 2007.

[23] Razek M.A., Frasson C., Kaltenbach M.A. Context-based information agent for supporting education on the Web. V. Kumar et al. (Eds.). ICCSA 2003, LNCS 2667, 2003. – P. 170-179.
[24] Analyti A., Theodorakis M., Spyratos N., Constantopoulos P. Contextualization as an independent abstraction mechanism for conceptual modeling// Information Systems. – 2007. – V. 32, No 1. – P. 24-60.
[25] Franklin M., Halevy A., Maier D. From databases to dataspaces: a new abstraction for information management// SIGMOD Record. – 2005. – V. 34, No 4.
[26] Choi J.-K., Nies L.F., Ramani K. A framework for the integration of environmental and business aspects toward sustainable product development// J. of Engineering Design. – 2008. – V. 19, No 5. – P. 431-460.
[27] Yeh-Chun Juan, Jiun-Shiung Lin, Chao Ou-Yang. A process-driven approach to develop multi-agent system for cooperation in concurrent new product development// CSIE '09: Proc. of the 2009 WRI World Congress on Computer Science and Information Engineering. – IEEE Computer Society, 2009. – V. 4.
[28] Bernstein P., Hadzilacos V., Goodman N. Concurrency control and recovery in database systems. – Addison-Wesley, 1987.
[29] Stonebeker M. Errors in database systems, eventual consistency, and the CAP theorem. BLOG@CACM, April 5, 2010. – перевод на русский С. Кузнецова с комментарием 19 мая 2010 г. – http://citforum.ru/gazeta/154/.
[30] Brewer E.A. Towards robust distributed systems// Principles of Distributed Computing. Portland, Oregon, July 2000, Invited Talk.
[31] Herlihy M.P., Wing J.M. Linearizability: a correctness condition for concurrent objects// ACM Transactions on Programming Languages and Systems. – 1990. – V. 12, No 3. – P. 463-492.
[32] Lamport L. On interprocess communication. I; II// Distributed Computing. – 1986. – V. 1, No 2. – P. 77-101.
[33] Risson J., Moors T. Survey of research towards robust peer-to-peer networks: search methods// Computer Networks. – 2006. V. 50, No 17. – P. 3485-3521.
[34] Snodgrass R.T. Developing time-oriented database applications in SQL// Morgan Kaufmann Series in Data Management Systems. – Morgan Kaufmann, 1997. – 504 p.
[35] Ali K.A., Pokorny J. A comparison of XML-based temporal models// Advanced Internet Based Systems and Applications. Lecture Notes in Computer Science, 2009. – P. 339-350.
[36] Grandi F. Multi-temporal RDF ontology versioning// Int. Workshop on Ontology Dynamics IWOD, 2009.
[37] Pugliese A., Udrea O., Subrahmanian V.S. Scaling RDF with time// Proc. of WWW Conf. – ACM Press, 2008. – P. 605-614.
[38] Tappolet J., Bernstein A. Applied temporal RDF: efficient temporal querying of RDF data with SPARQL// Proc. of ESWC Conf. – Springer-Verlag, 2009. – P. 302-322.
[39] Noh Seo-Young, Gadia Sh.K. Benchmarking temporal database models with interval-based and temporal element-based timestamping// J. of Systems and Software. – 2008. – V. 81, No 11. – P. 1931-1943.
[40] Moon H.J., Curino C.A., Deutsch A., Hou C.-Y., Zaniolo C. Managing and querying transaction-time databases under schema evolution// Very Large Data Base VLDB, 2008.
[41] Знаменский С.В. Контекстно-автономная информационная система. Четвёртая конференция «Свободное программное обеспечение в высшей школе». Переславль, 30 января – 1 февраля 2009 года. Тез. докл. – М.: ALT Linux, 2009. – С. 32-37.
[42] Абрамов С.М, Знаменский С.В., Живчикова Н.С., Котомин А.В., Титова Е.В. Информационная система для разработки технологий организации сложной совместной деятельности// Труды XI-й Всерос. науч. конф. «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL-2009. – Петрозаводск: КарНЦ РАН, 2009.

Information system for education

S.V. Znamenskij

Review the objectives and content of education in the context of "one student – one computer" should be supported by highly reliable information system. It should provide a flexible, efficient and versatile organization of a variety of creative collaborative work, based on digital libraries.

The report focuses on the problem of creating an information system with the required qualities. Explained why relational databases have no perspectives in future. Alternative approach description is provided.

♣ Работа выполнена при финансовой поддержке РФФИ (проект 09-07-00407)