Эти вопросы должны быть решены и система технически должна позволять настроить необходимые параметры.

– *Понять смысл сразу (to catch the meaning at once)*

Крупная система тяготеет к повышенной сложности понимания. Программные интерфейсы и описания данных должны обладать богатыми выразительными возможностями. Небольшой набор концепций должен обеспечивать легкое понимание механизмов системы. В качестве таких понятий может быть использована, например, аналогия XML-файлов (по аналогии с абстракцией файла в POSIX системе).

– *Дружественность к разработчику (developer friendliness)*

К построенной системе разработчики прикладных подсистем должны иметь доступ с помощью привычных средств и языков. Механизмы веб-сервисов в некоторой степени позволяют обеспечить доступность системных интерфейсов для разработчиков из различных языков.

**УДК 658.1:62-52**

**Григорьев А. А., инженер-программист, РУП «ГИВЦ Минсельхозпрода», г. Минск**

## О МЕТОДОЛОГИИ ПОСТРОЕНИЯ КОРПОРАТИВНЫХ СИСТЕМ ПРИНЯТИЯ РЕШЕНИЙ И ДОКУМЕНТООБОРОТА

Успешное внедрение инновационных информационных систем, разработанных в ГИВЦ, во многом зависело от методологий, использованных при проектировании и разработке, которые рассмотрены в данной статье.

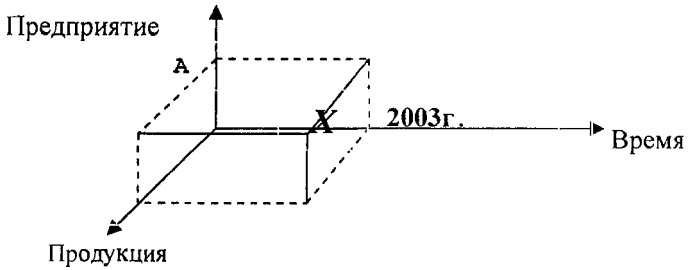
Создание информационных систем – дело далеко не простое. По мере того как возрастает их сложность, процессы конструирования соответствующего программного обеспечения становятся все более трудоемкими, причем затраты труда растут экспоненциально. Наиболее сложные вопросы, с которыми приходится сталкиваться многим разработчикам корпоративных систем, – это конфликты между гибкостью и безопасностью, полнотой информации и простотой интерфейса пользователя, частым масштабированием и физической

удаленностью терминалов, а также повышением скорости разработки и сохранением качества.

В статье будут рассмотрены примеры: ядро многомерного анализа данных и ядро системы документооборота предприятия. Это базовые компоненты систем подготовки принятия решений «Нива-2» и бухгалтерского учета.

*Многомерный анализ данных*

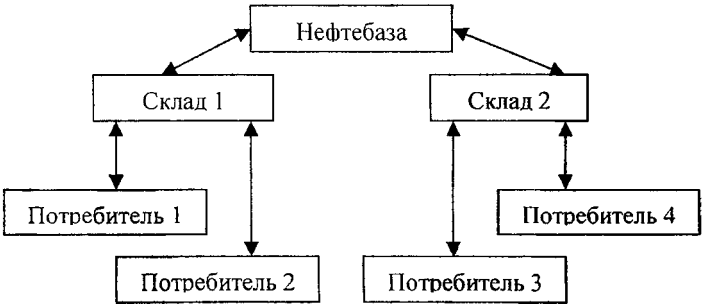
Что понимается под многомерной структурой информации и почему важно использовать это свойство информации? Предположим, хозяйство А в течение года произвело Х тонн пшеницы. Что на самом деле представляет собой это утверждение с точки зрения информационных технологий?




Это утверждение говорит о том, что число Х (которое наверняка должно храниться в базе данных и использоваться для анализа и планирования) зависит от трех параметров: географического, временного и вида продукции. Число Х мы называем фактом. Параметры мы называем осями. Итак, для реализации хранения такого факта нам потребуются 4 (!) реляционные таблицы: три таблицы осей и одна таблица фактов с тремя индексами. Естественно, что рассматриваемая ситуация является упрощенной до предела: количество осей в реальных задачах планирования значительно большее, факты могут иметь разную размерность и, что более важно, факты могут иметь случайную природу. Важно учитывать это свойство информации на самых первых шагах проектирования хранилища данных.

Теперь рассмотрим другой аспект проблемы – задачу оперативного планирования. Например, необходимо распределять горючее из центрального резерва в ситуации нехватки во время уборочной

кампании. В конце каждого дня поступает текущая информация об остатках горючего и необходимом количестве на следующий день. Эта информация будет использована на следующее утро на складе горюче-смазочных материалов. Но еще до этого данные будут агрегированы и отправлены на нефтебазу с централизованным резервом. И уже там, исходя из объема резерва, будет рассчитано, в какое хозяйство сколько горючего будет отправлено. Моделью такой ситуации может служить дерево.




Для хранения дерева в реляционной базе данных должна использоваться лишь одна таблица, а агрегированные данные получаться расчетом. При большом количестве обращений это требует огромных вычислительных ресурсов. При использовании специализированных хранилищ данных можно обойти некоторые ограничения нормализованности реляционных таблиц без усложнения проекта.

Решением двух описанных методологических проблем, является использование ядра СУБД, учитывающего важнейшие свойства информации. Ядро системы подготовки принятия решений «TreeStorage» разрабатывалось с целью обеспечить хранение данных в многомерном пространстве параметров, в котором каждая ось представлена деревом, и автоматическую агрегацию данных. На пересечении отметок на осях хранятся факты. «TreeStorage» поддерживает как факты первого уровня – те, что вводятся вручную, так и агрегированные значения. Правила агрегирования задаются формулами. В реальности формулы могут быть очень разнообразны. От простых математических выражений, таких, как сумма, к более сложным, таким, как дисперсия, и до совсем сложных, таких, как экстраполяци-

онные отчеты. Еще одним важным аспектом работы СППР являются графики. Сами по себе графики уже не являются первичными фактами, однако при этом они допускают повторное использование: можно объединять графики одного показателя за разные сроки (например, присоединять график за каждый следующий месяц к предыдущему) или строить на одних координатных осях графики двух разных величин для сравнения. Такой тип операций с графиками мы также называем формулами, хотя они совсем не похожи на обычные математические выражения.

Итак, рассмотрено несколько методик, благодаря которым СППР, основанные на «TreeStorage», предоставляют гибкость и масштабируемость в сочетании с удобством настройки под конкретную задачу. И список преимуществ отнюдь не ограничивается этим, однако более интересно будет рассмотреть дальнейшие вопросы на примере ядра системы документооборота.

*Документооборот*

Система документооборота является очень характерным примером того, как информационные технологии могут привести к радикальной экономии времени, сил и более правильной организации производственного процесса. Как мы этого добиваемся? Рассмотрим характерный пример. Начальник отдела АСУ пишет служебную записку с просьбой о покупке нового компьютера. Потом идет с ней к заместителю директора, – тот оставляет вопрос открытым, – потом к самому директору. Директор не возражает, но лишь в том случае, если модель будет дешевле на 200000. Документ возвращается в АСУ. Выбирается новая модель. Потом к директору. Потом в бухгалтерию. И так далее. Каждый, кому приходилось собрать полтора десятка подписей на лист бумаги, наверняка думал о следующем подходе. Ввести документ в компьютер. Отправить его по определенному маршруту. И узнать о результате. Ну а те, кто привык находиться «по другую сторону стола» т.е. подписывать, тоже заинтересованы в том, чтобы все документы приобрели единообразный вид, чтобы можно было легко и быстро просмотреть, принять решение и «одним кликом» отправить дальше. Проектирование систем на основе бизнес-процессов является следующей ступенью развития после объектного проектирования.

Обратимся к модели документа, реализованной в ядре системы документооборота. Она держится на трех китах: эволюционности, жизненном цикле и разделении доступа. Эволюционность означает, что перед внесением каких-либо изменений в документ его копия сохраняется в архиве. Благодаря этому никакие изменения не остаются бесследными. Всегда можно узнать: кто, когда, почему и что изменил. Эволюционность позволяет увидеть содержание документа на любой момент его жизни от создания до удаления по причине истечения срока хранения. Жизненный цикл представлен набором состояний документа и набором операций, переводящих документ из одного состояния в другое. Операции могут быть выполнены как вручную, так и автоматически. Разделение доступа основывается на трехуровневой системе контроля, построенной по принципу «все, что не разрешено, – запрещено». Первый уровень контроля – разрешение доступа к документам конкретного типа, второй – доступ к этим документам в лишь ограниченном подмножестве состояний и третий – разрешение на выполнение лишь строго определенных операций в каждом состоянии.

Жизненный цикл документов и система доступа реализуют концепцию витрин данных. Поэтому каждый из пользователей системы документооборота имеет доступ только к тем документам, которые он сам должен обработать. Это обеспечивает с одной стороны удобство работы, а с другой – надежную защиту от несанкционированного доступа. Система документооборота благодаря реализации методологии витрин данных приобретает дополнительную функциональность, программирование которой обычными способами потребовало бы значительных трудовых ресурсов: возможности групповой работы над документами, протоколирование обсуждений, журнал операций и изменений, внутренняя система электронной почты. Использование витрин данных на стороне сервера существенно ускоряет создание интерфейса пользователя.

Основные технологии:

Политика ГИВЦ в разработке программного обеспечения основывается на использовании новейших технологий как инструментального, так и научно-методологического характера. При построении описанных ядер были использованы следующие принципы:

– Надежное и быстрое сохранение и чтение данных. Достигается использованием в основе ядра «TreeStorage» постреляционной СУБД Intersystems Cache.

– Возможность хранения различных типов данных. Необходимый минимум включает числа, текст, формулы (арифметика, ссылки, запросы к другим базам), графики, ссылки на файлы, ссылки на хранимые объекты, XML-контент.

– Транзактность. Журналирование всех изменений в хранилище информации.

– Разграничение прав пользователей. Возможность создания и изменения витрин данных.

– Многопользовательский удаленный доступ к данным. Доступ через Internet. Автоматическая генерация интерфейса пользователя для витрин данных сервером.

– Поддержка автоматизации расчетов внутри хранилища данных.

– Удобный внутренний интерфейс: объектный доступ, инкапсуляция рутинных процессов выборки, фильтрации, обновления. Возможности для развития и повторного использования кода.

*Заключение*

Описанные методологии позволяют создавать более гибкие и масштабируемые информационные системы. Их основными преимуществами являются естественная ориентация, фундаментальные свойства информации и бизнес-процессов. Однако важно помнить, что для соответствия программного обеспечения современным стандартам необходимо его постоянное совершенствование. К дополнительным сложностям ведения инновационных разработок следует также отнести необходимость высокой квалификации разработчиков. К счастью, однако, результат стоит затраченных ресурсов.