С.Н. Ардатский
О.С. Бартунов
Е.Б. Родичев
Ф.Г. Сигаев

ТЕХНОЛОГИЧЕСКИЕ ПРИНЦИПЫ ИНФОРМАЦИОННОГО ОБРАЗОВАТЕЛЬНОГО РЕСУРСА

Трехуровневая схема сервера информационного образовательного ресурса

В основе технологической платформы информационного образовательного ресурса лежит применение трехуровневой схемы, которая обеспечивает большую гибкость и масштабируемость, чем более простая и широко применяемая схема «клиент – сервер». Верхний уровень такой схемы представляют собой внешние интерфейсы. Их количество неограниченно, они могут добавляться к системе по мере необходимости. По этим интерфейсам осуществляется все общение системы с внешним миром — это могут быть web-серверы, почта для приема/выдачи информации, современные объектные протоколы типа IIOP, или даже совсем специфические, например, сделанные по заказу конкретного клиента.

Средний уровень — это общая шина данных и операций. Она имеет единый стандартизованный интерфейс. Все внешние интерфейсы, общаясь с внешним миром по своим различным протоколам, при общении с общей шиной трансформируют запросы и данные в единый стандарт шины. Основная задача общей шины — диспетчеризация и маршрутизация информационных потоков, представленных в стандартном едином формате.

Нижний уровень состоит из произвольного количества хранилищ и обработчиков данных. Это могут быть различные серверы баз данных, файловые хранилища, специфические поисковые серверы и т.д. Имея совершенно различную внутреннюю структуру, все эти серверы опять же общаются с шиной по единому протоколу, обмениваясь с ней информацией, принимая и выдавая команды на обработки, и т.п.

В частности, этот нижний уровень логически составляет единую базу данных такой системы. Наиболее общей структурной единицей в системе является объект, и общая шина обеспечивает его целостность. Это означает, к примеру, что заголовок статьи может физически храниться в одной базе данных нижнего уровня (например, для быстрого поиска по заголовкам), а текст статьи — совершенно в другой, скажем, оптимальной для полнотекстового поиска. Но на запрос внешнего интерфейса «показать такую-то статью» она будет выдана шиной целиком, в своем исходном виде.

Такая схема имеет много важных преимуществ при построении крупных проектов. Одним из наиболее существенных в нашем случае является масштабируемость. Количество серверов на каждом из трех уровней определяется не количеством клиентов (их может быть сколь угодно много), а лишь количеством принципиально разнотипных операций и задачей равномерного распределения нагрузки по серверам для обеспечения высокой нагрузочной способности системы в целом. Кроме того, добавление новых серверов производится на ходу и ни в коей мере не нарушает постоянной работоспособности системы.

Многостадийная система разработки

Программное обеспечение информационных образовательных ресурсов, особенно его часть, связанная с адекватной визуализацией хранимой информации, требует постоянного расширения и доработки. Однако внесение исправлений непосредственно в работающую версию недопустимо, поскольку может нарушить работоспособность системы. Для решения этой проблемы предназначена многостадийная система разработки и отладки прикладного программного обеспечения серверов информационных образовательных ресурсов.

Данная система включает в себя следующие уровни (этапы разработки).

- Уровень предварительной разработки (preview).

Все приложения данного уровня работают в чисто динамическом режиме и доступны для внесения изменений администраторам и разработчикам.

На этом же этапе производится быстрая первичная отладка и выявление основного множества ошибок в новом или модифицированном программном коде.

- Уровень окончательного тестирования (approve).

Страницы, не требующие обновления, представлены в статическом виде. Приложения данного уровня доступны администраторам, разработчикам и тестерам в режиме «только для чтения/исполнения».

- Уровень рабочего использования (production).

Приложения представлен в таком же виде, как на уровне approve. Приложения данного уровня доступны всем пользователям системы в режиме «только для чтения/исполнения».

Приложения всех уровней пользуются одной и той же базой документов, файловым хранилищем и рубрикатором.

Для перевода программного обеспечения на следующий уровень разработки используется специальная процедура выкатки. При этом при переводе программных модулей с этапа preview на этап approve производится генерация статических страниц из динамических для приложений, не требующих обновления. Перевод приложений с этапа approve на этап production заключается в простом копировании соответствующих файлов приложений в общедоступную директорию.

Допускается частичная выкатка программного обеспечения. Списки выкатываемых модулей определяются разработчиками системы.

В системе информационного обеспечения учебного процесса одновременно сосуществуют программные модули, находящиеся на каждом из трех уровней разработки. Для непривилегированных пользователей системы доступны только приложения уровня production.

Политика ролевого разграничения доступа

В самом общем виде любая система разграничения доступа к ресурсам реализует так называемую матрицу доступа, которая за-

дает права доступа каждого субъекта (пользователя) к каждому из ресурсов системы (файлам, программам, устройствам и т.д.). Пример подобной матрицы приведен ниже.

	Ресурс 1 (Файл)	**Ресурс 2 (Директория)**	**Ресурс 3 (Устройство)**	**Ресурс 4 (Сервис)**
User 1	R W	R W X	R X	—
User 2	R	R X	—	—
User 3	—	—	X	X

Однако поддержание подобной матрицы в общем виде очень трудоемко и ненадежно. В типичной системе фигурируют сотни субъектов (пользователей) и десятки тысяч объектов (файлов, документов). Кроме того, многочисленные пользователи системы обычно объединены в существенно меньшее число групп с одинаковыми внутри каждой группы правами. Поэтому очень удобной является система ролевого разграничения доступа к ресурсам, в которой элементарные права пользователей объединены в функциональные группы (роли) и присвоение пользователю той или иной роли означает включение его в соответствующую группу владельцев данных прав (или данных сочетаний прав).

Модель (политика) ролевого разграничения доступа основывается на функциях, которые данный пользователь должен выполнять в соответствующей системе. Реализация данной политики требует разработки набора необходимых ролей (библиотеки ролей), соответствующей данной системе. Роль определяется как набор прав доступа к объектам системы и прав на выполнение над ними определенного набора элементарных действий, соответствующих функций, которые должен выполнять в системе определенный пользователь.

Модель ролевого разграничения доступа обладает рядом полезных свойств.

1. Позволяет разрабатывать набор ролей на основе представления о правильном функционировании той или иной системы, не

используя сведений о реальном разделении функций, исторически сложившихся в ходе ее эксплуатации.

2. Позволяет разделить сферу ответственности между администратором авторизационной схемы и диспетчером прав доступа пользователей. При этом администратором авторизационной схемы обычно является привилегированный пользователь, который занимается созданием и поддержкой авторизационной схемы системы (определяет привилегии, роли). Администратор авторизационной схемы обязан хорошо знать объекты системы, приложения и обычно входит в группу разработчиков (проектировщиков) системы. Диспетчер доступа определяет отношения пользователя и ролей. Обычно он входит в контентную группу, работающую с пользователями, и распределяет роли среди них.

3. Разделение полномочий на управление отношений пользователь — роль, роль — привилегии позволяет естественным образом минимизировать как затраты на управление, так и вероятность внесения ошибок администратором.

Кроме того, важнейшим свойством RBAC-модели является ее нейтральность, т.е. RBAC предоставляет аппарат для выражения политики безопасности, а не накладывает определенные ограничения, соответствующие определенной политике. Используя RBAC-модель, можно эмулировать стандартные политики дискретного и мандатного ограничения доступа к ресурсам. Это является крайне важным, так как можно разрабатывать системы, поддерживающие RBAC-модель и конфигурировать их в случае необходимости для реализации дискретного или мандатного управления доступом.

Базы данных

При построении современных информационных систем приходится решать разнообразные технологические задачи, связанные с хранением, доступом и поиском информации. Учитывая современные требования к производительности, надежности и шкалированию таких систем, такие задачи требуют использования достаточно сложных алгоритмов и специализированных структур данных.

Хранилище документов выполняет функции сохранения во внутреннем формате и выдачи по запросу системы документов.

На хранилище документов также возлагается реализация функции поиска документов по заданным атрибутам и функция индексации текстовых частей сообщения для выполнения полнотекстового поиска. В качестве хранилища документов выбрана распространяемая с исходными текстами SQL база данных PostgreSQL.

СУБД PostgreSQL отличается высокой степенью надежности, скоростью работы и устойчивостью при больших нагрузках. Независимое тестирование показало, что производительность PostgreSQL сравнима с производительностью коммерческих БД Oracle и Informix.

В настоящее время СУБД PostgreSQL обладает рядом уникальных свойств, отсутствующих или не полностью реализованных в коммерческих базах данных.

1. Поддержка интерфейсов ODBC, JDBC, а также интерфейсов с языками C, C++, php, perl, tcl, ECPG, python, Ruby.

2. Реализация триггеров, подзапросов, функциональных и частичных индексов.

3. Реализация механизмов наследования и создания последовательностей.

4. Динамически загружаемые модули.

5. Встроенная поддержка протокола шифрования SSL.

6. Стандартизированный способ создания новых типов данных и методов работы с ними.

7. Полная локализация (в том числе русская), поддержка Unicode.

Работа PostgreSQL протестирована на 34 аппаратных платформах по целым рядам операционных систем (как 32-, так и 64-битных). В 2001 г. СУБД PostgreSQL была признана «лучшей базой данных для операционной системы Linux».

В отличие от коммерческих БД распространение и установка PostgreSQL не требует лицензионных отчислений. PostgreSQL распространяется под так называемой лицензией BSD, которая позволяет строить на основе СУБД PostgreSQL свободные или коммерческие приложения и программные продукты без лицензионных отчислений и практически без каких-либо ограничений. Условия

распространения PostgreSQL не могут измениться по решению производителя, что возможно для коммерческих СУБД.

Основные функции PostgreSQL (ввод, вывод, поиск и т.д.) полностью локализованы. Система допускает поддержку пяти основных кодировок кириллицы (CP-866 (MS-DOS), CP-1251 (Windows), KOI8-r, Macintosh и ISO-8859-5), а также стандарта Unicode. Для системы PostgreSQL разработан ряд приложений расширенной работы с русским языком: поддержка и поиск по словоформам, работа со словарями и т.д.

Кроме того, в архитектуре PostgreSQL были заложены прогрессивные идеи, такие как расширяемость, нашедшие затем развитие в коммерческих СУБД (например, Informix, Illustra). Именно архитектура PostgreSQL позволила решить множество проблем, связанных с необходимостью введения новых типов данных, быстрых методов доступа и специализированных запросов. Некоторые из авторов данной статьи являются разработчиками обобщенного поискового дерева (GiST), входящего в ядро PostgreSQL, которое дает возможность специалистам в конкретной области знаний создавать специализированные типы данных и обеспечивает индексный доступ к ним не будучи экспертами в области баз данных. Информация по GiST, исходные тексты, ссылки на статьи и история развития доступны на страничке авторов по адресу http://www.sai.msu.su/~megera/postgres/gist/.

Внутренний и внешний полнотекстовый поиск

Основной задачей любой информационной системы является обеспечение быстрого доступа к документам. Одним из способов ее решения является полнотекстовый поиск по всей коллекции документов. Поиск должен обеспечивать высокую релевантность результатов поиска запросу, актуальность поисковых индексов и хорошую масштабируемость по производительности.

Хорошо известные поисковые машины общего назначения (Яндекс, Rambler, Google) в той или иной мере обеспечивают поиск по всем ресурсам любой открытой системы. Однако эти поисковые си-

стемы в силу своей «общности» привносят большой «шум», а поисковая база отстает на 2—3 месяца. Кроме того, «закрытость» программного обеспечения этих систем не позволяет интегрировать ее с собственными внутренними сервисами образовательной информационной системы.

Использование полнотекстового поиска, встроенного в базу данных, позволяет использовать метаданные документа для задания различных критериев поиска и более гранулированного контроля доступа определенных групп пользователей к внутреннему хранилищу документов.

Использование полнотекстового поиска по внешним ресурсам с поддержкой каталога ресурсов позволяет создавать поисковые машины, специализированные по определенной тематике.

Технологические принципы и приемы, описанные в работе, прошли апробацию при создании портала «Rambler». Они также были использованы при создании некоторых из федеральных образовательных порталов, созданных в рамках Федеральной целевой программы «Развитие единой образовательной информационной среды (2001—2005)».