Информационные ресурсы и технологии

ПЕТРОВ Олег Михайлович – *доктор технических наук, профессор, зав. кафедрой Московской государственной академии приборостроения и информатики (МГАПИ)*

ЖУКОВ Дмитрий Олегович *- кандидат физико-математических наук, доцент, директор ЦНИТ МГАПИ*

ЗОТОВ Дмитрий Дмитриевич – *инженер-программист ЦНИТ МГАПИ*

ОСОБЕННОСТИ ХРАНЕНИЯ И ШИФРОВАНИЯ ДАННЫХ В WEB-СИСТЕМАХ ОБРАБОТКИ ИНФОРМАЦИИ

Наряду с вопросами о типе системы и выборе платформы при разработке сложных Web-систем (например, Интернет-магазина, системы дистанционного обучения и т.д.) одним из основных вопросов, который необходимо решать, также является вопрос о структуре и типе данных [1]. Проблема заключается в том, что ответ на него может меняться по ходу разработки. Однако в самом начале все-таки можно выделить несколько основных положений, это:

- какого типа будут данные,
- примерный объем данных,
- где и в каком виде они будут храниться.

В случае разработки системы дистанционного обучения (наиболее актуальная в настоящее время задача, стоящая перед большинством вузов) возникает следующая ситуация: поскольку такая системы является серверно-ориентированным приложением, то все данные, за исключением информации о конкретном входе в систему пользователя, хранящейся в cookies браузера на стороне клиента, должны храниться и на сервере. Подобный подход обеспечивает дополнительную безопасность данных, а главное, позволяет клиенту обращаться к системе и получать ответ с любой машины, имеющей доступ в интернет.

Предположительный объем данных рассчитать крайне сложно, поскольку общая концепция системы предусматривает открытость для пользователей (как для студентов, так и для преподавателей), а, следовательно, не имеет количественных ограничений за исключением ограничений доступа в связи с перегруженностью системы и пропускной способностью сетей. Зарегистрированные преподаватели имеют право на размещение собственных курсов лекций, тестов, контрольных работ и другой информации, отвечающей задачам системы. Учебные курсы – это мультимедийные гипертекстовые документы, и их размер может очень сильно варьироваться в зависимости от объема графики и текста. О том, как рассчитать примерный объем системной информации (учетных записей пользователей, подразделений ВУЗа, предметов и т. д.), мы поговорим чуть позже. Примем предполагаемый объем данных равным 500 Мб ~ 2 Гб [2, 3].

Теперь рассмотрим, какого типа будут данные. Из вышесказанного видно, что данные в системе будут двух типов - неоднородные мультимедийные данные в виде гипертекстовых файлов и жестко структурированные данные в виде таблиц базы данных. К неструктурированным данным также относятся всевозможные материалы факультетов, кафедр, и групп.

Структура «табличных» данных такой системы может иметь следующий вид:

- учетные записи пользователей;
- учетные записи подразделений ВУЗа;
- учетные записи предметов;
- информация о посещениях системы;
- расписание;
- результаты тестирования;
- данные из системы сообщений;
- данные с форума.

В качестве СУБД может быть выбран сервер баз данных MySQL [1].

Во-первых, MySQL – полностью кроссплатформенная, переносимая система, что позволяет устанавливать системы дистанционного образования, разработанные с ее использованием как на сервера на основе систем семейства Unix, так и на Windows-машины. Во-вторых, MySQL предоставляет огромные возможности для работы с данными, при этом имея значительно более низкие системные требования и высокое быстродействие в сравнении с MSSQL, Oracle или DB2. Правда, есть и ряд ограничений, в частности, по объему базы данных, а, следовательно, и по количеству записей. Начиная с версии 3.23, MySQL поддерживает размер таблицы до 8 миллионов Терабайт, однако надо отметить, что операционные системы накладывают свои ограничения на размеры базы данных.

Мы рассмотрим возможный размер таблиц для платформ Intel x86, поскольку они являются наиболее распространенными в небольших системах и системах средней величины (платформы Alpha и UltraSPARC, а также операционную систему Solaris мы рассматривать не будем, так как подобные сервера являются весьма дорогостоящими и их использование в качестве платформы не оправдано) [4].

В операционных системах Linux размер таблицы колеблется от 2 до 4 Гб, в зависимости от версии ядра и дистрибутива. Существует патч LFS, позволяющий увеличить размер таблицы до одного терабайта.

Разработчики дистрибутивов Linux Mandrake и RadHat утверждают, что размер таблицы MySQL ограничивается только объемом свободного места в разделе /usr, хотя эта информация не подтверждается разработчиками MySQL.

О размере таблиц в системах FreeBSD, OpenBSD и NetBSD разработчики MySQL также умалчивают, хотя некоторые источники в сети Интернет, ссылаясь на разработчиков MySQL и университет Berkeley, утверждают, что во FreeBSD и OpenBSD размер таблиц колеблется от 2 до 16 Гб, а система NetBSD поддерживает размер таблицы до 1 терабайта на встроенных жестких дисках и до 8 миллионов терабайт во внешних хранилищах.

Информации из официальных источников о размере таблиц в среде Windows отсутствует, в то время как неофициальные источники утверждают, что размер таблиц ограничивается только свободным местом на диске [5].

В любом случае, если планируется создать таблицу размером больше 4 Гб, можно сделать это «принудительно», независимо от операционной системы, используя при создании таблицы команды AVG_ROW_LENGTH и MAX_ROWS. Команда AVG_ROW_LENGTH вычисляет среднюю длину строки таблицы, а MAX_ROWS – максимальное количество строк. При создании таблицы ее будущий максимальный размер вычисляется по формуле AVG_ROW_LENGTH * MAX_ROWS. Таким образом, если при создании таблицы задать значение MAX_ROWS=1000000, вы получите таблицу, в которую можно занести один миллион записей, не зависимо от того, какие ограничения накладывает операционная система. (Надо заметить, что значение MAX_ROWS можно изменять и после создания таблицы с помощью команды ALTER.) Конечно, если размер таблицы превысит объем свободного места на диске, возникнут ошибки, результат которых трудно спрогнозировать.

Важно помнить, что речь идет не о размерах всей базы данных, а о размерах одной таблицы. На сегодняшний день в системах дистанционного образования используется в среднем 7-9 таблиц, но в дальнейшем, при совершенствовании системы количество таблиц может измениться.

Но даже если мы рассматриваем размер одной таблицы, не превышающей 4 Гб, возможно будет хранить примерно следующее количество данных (с учетом заполненных «дополнительных» полей и максимальной длины всех вводимых данных):

- 8,5 миллионов учетных записей пользователей;
- 9,7 миллионов записей подразделений ВУЗа (факультетов, кафедр, групп);
- 102 миллиона записей о посещении системы;
- 10 тысяч записей о предметах, если в каждом предмете содержится по 1000 тестов (учитывая, что в каждом предмете содержится в среднем 15 тестов, эта цифра возрастает до 667 тысяч).

Таким образом, если системой ежегодно пользуются 50000 человек (5 курсов примерно по 10000 человек, плюс преподаватели, плюс администраторы), то можно хранить информацию в базе данных, не удаляя ненужные и устаревшие записи примерно 170 лет.

К плюсам MySQL можно отнести еще и то, что данная СУБД распространяется бесплатно, а язык PHP содержит ряд очень удобных средств для работы с ней, что позволяет легко создавать запросы любой сложности.

Как упоминалось выше, таблицы базы данных содержат «дополнительные поля». Они необходимы, если система строится по модульному принципу, то есть возможна замена либо добавление какого-нибудь модуля. В случае замены старого модуля более новой версией в большинстве случаев не требуется изменение структуры данных. Но если к системе подключается новый модуль, выполняющий функции, не предусмотренные базовой конфигурацией разрабатываемой системы (например, модуль отчетов, написанный сторонним разработчиком), очень часто возникает потребность в изменении структуры базы данных, в частности в добавлении новых полей. Конечно, можно добавлять, удалять и изменять любые поля таблицы и после ее создания, но в некоторых случаях это может привести к повреждению и потере данных, а если в таблице уже есть несколько тысяч записей, то процесс изменения структуры может занять длительное время. Если же максимальный размер таблицы был определен при ее создании (директива MAX_ROWS), то изменение структуры может привести к непредсказуемым последствиям [6].

Из данной ситуации можно выйти следующим образом: в каждой таблице создается «дополнительное» пустое поле строкового типа диной 255 символов. В это поле можно заносить любые данные, например, ссылку на выборку полей из другой базы данных, находящейся на другом сервере. Формат данных – произвольный. Разработчику остается только написать функцию разложения этой строки на операторы и данные (parser). Кроме того, можно создать стандартный parser и описать рекомендуемую логику заполнения дополнительных полей. При этом выбор, использовать ли стандартные функции или писать собственные, остается за разработчиками.

Еще один важный аспект: дополнительное поле, как, впрочем, и большинство строковых полей в базе данных, имеет тип VARCHAR(N) – строка произвольной длины. Данный тип, в отличие от CHAR(N), не выделяет заранее N байт дискового пространства для записи строки, то есть если поле типа VARCHAR не содержит символов, оно не занимает места на диске. Подобный подход к построению базы данных позволяет сократить размер одной записи, а, следовательно, увеличить максимальное количество записей в таблице. А дополнительные поля, в случае если они не используются, не увеличивают объема данных.

Теперь поговорим еще об одном немаловажном моменте, а именно, о хранении материалов предметов на сервере. Материалы предмета могут содержать абсолютно разнородную информацию: тексты лекций, статические и динамические изображения, видеофрагменты, аудиоклипы, тесты, контрольные работы, демонстрации опытов и т. п. [7]. Из этого следует, что поместить учебные материалы предметов в базу данных не представляется возможным. На то есть и еще одна причина: если через некоторое время после публикации предмета возникает необходимость внести изменения, то, учитывая неоднородность информации, придется формировать очень сложный запрос к базе данных (возможно, даже не один), что преподаватель, не знающий языка SQL просто не в состоянии сделать. Автоматизировать этот процесс тоже не представляется возможным, поскольку заранее даже примерно нельзя определить, каким будет этот запрос.

Решение данной задачи довольно просто. Например, при регистрации преподавателем нового предмета на жестком диске сервера создается папка с уникальным именем. Настройки безопасности файловой системы позволяют ограничить доступ к этой папке, чтобы обеспечить сохранность материалов. В корне папки находится ini-файл, описывающий конфигурацию предмета, разрешение на доступ к тем или иным его материалам и всю остальную необходимую информацию. Большинство материалов предмета хранится в виде HTML документов, поскольку данный формат позволяет использовать самые разнородные данные и организовывать ссылки между ними. Другие материалы (тесты, демонстрации опытов) также находятся внутри папки предмета и описываются конфигурационным файлом. (Представление материалов предмета в виде ссылок на внешние источники запрещено в целях обеспечения дополнительно безопасности.)

Предметы, например, могут создаваться с помощью специальной программы, которая позволяет сгенерировать на выходе архивный инсталляционный пакет. Этот пакет преподаватель загружает на сервер через свою учетную запись в Web-системе, после чего он автоматически распаковывается в соответствующий каталог и, в случае удачной установки, удаляется. В противном случае модуль автоматического информирования системы оповещает администратора, и тот устанавливает предмет «вручную».

Если автор предмета вносит в него изменения, то генерируется инсталляционный пакет (patch), содержащий в себе только измененные материалы, а не весь предмет. Объем этого патча значительно меньше основного инсталляционного пакета. При этом контрольные суммы модифицированных материалов меняются, чтобы можно было отслеживать их дальнейшие изменения. В случае если процент внесенных изменений выше определенной отметки (по умолчанию 75%), заново создается весь установочный пакет. При этом в обоих случаях процедура установки идентична описанной выше.

В ходе разработки возник еще один вопрос, решение которого потребовало достаточно больших усилий. Это способ хранения информации о тестах и контрольных работах. Сам текст вопроса и ответов хранится в HTML-шаблоне. Но в какой форме хранить ответ? Естественно, что номер ответа или его значение нельзя хранить в открытом текстовом виде внутри ini-файла, поскольку пользователь сможет просмотреть эти значения. (Здесь нужно сделать одну оговорку: если пользователю удастся просмотреть ini-файл, то значит, безопасность сервера должным образом не настроена, и, скорее всего, он сможет также просмотреть или даже изменить исходные коды системы, что может привести к сбоям в работе или к ее полному краху. Конечно, для того чтобы сделать это, злоумышленник должен обладать достаточно высоким уровнем знаний в области операционных систем, серверных систем и программирования на языках PHP и JavaScript.) Самый простой и надежный способ избежать этого – хранить hash суммы ответов, а не их значения. Hash сумма каждого ответа особым образом генерируется по алгоритму md5 (или другому ассиметричному алгоритму) из контрольной суммы теста, номера вопроса и значения ответа [8]. Для теста значение ответа – это его порядковый номер. Для числового ответа – число. Для текстового – «стандартизированная» строка (стандартная кодировка, удаленные пробелы, прописные буквы).

При запуске тестирования генерируется последовательность вопросов, исходя из настроек предмета и конкретного теста. Студент выбирает номер ответа либо вводит его значение, нажимает кнопку «Ответить». На первый взгляд проблема решена. Но оказалось не все так просто. Как быть, если значение может быть неточным, округленным или вопрос имеет несколько правильных ответов?

Рассмотрим самый распространенный случай: ответ задачи равен 3,66 в периоде. Создать контрольную сумму периода невозможно, поэтому мы задаем рамки правильного ответа от 3,6 до 3,7. Каждый из ответов получает свою контрольную сумму. Но мы не можем их сравнить, поскольку md5 не имеет обратного хода, а его результаты абсолютно не схожи даже при очень схожих исходных значениях. Можно воспользоваться симметричным алгоритмом, например DES, но в этом случае результат шифрования будет разительно отличаться от md5 суммы, то есть придется писать ряд дополнительных функций для работы с таким вариантом ответа. К тому же язык PHP в стандартной установке не содержит симметричных алгоритмов. Библиотека Mcrypt, поставляемая вместе с дистрибутивом PHP, дает огромные возможности симметричного шифрования, но на большинстве серверов она не установлена, а ее установка требует некоторых специальных знаний [8].

При разработке системы необходимо ориентироваться на минимальные требования к программному и аппаратному обеспечению, поэтому вариант с дополнительной установкой библиотеки Mcrypt отпадает сам собой. В частности, нами вместо этого был разработан уникальный алгоритм, получивший название «Pseudo MD5». Его суть в том, что исходя из входных данных и ключа, он генерирует контрольную сумму, с виду

ничем не отличающуюся от результата работы md5, но при этом имеющую возможность расшифровки. Единственным его недостатком является тот факт, что максимальная длина входных данных ограничивается 30 знаками. Но для числового ответа на задачу этого вполне достаточно.

В результате внедрения в систему Pseudo MD5 стало возможным давать в виде ответа одно или несколько значений, неточные значения или значения в промежутке.

В заключении надо заметить, что все выше сказанное о хранении данных на сервере напрямую зависит от его настроек безопасности. Дело в том, что язык PHP не предусматривает предкомпиляции, и его код выполняется «на лету». Файлы исходного кода хранятся на сервере в текстовом виде, что позволяет их легко изменять. В свете этого, настройки сервера являются последней, а лучше сказать, первой линей обороны. Если злоумышленник получил доступ к файловой системе сервера, то все наши способы защиты информации становятся бесполезными. Такой пользователь, обладая некоторыми знаниями в области программирования, сможет вносить любые изменения в базу данных, при этом администраторы даже не будут об этом знать.

В большинстве случаев в качестве сервера используется программа Apache. Запретить доступ к данным определенной директории можно очень просто, задав для нее в конфигурационном файле httpd.conf директиву AllowDenay none. В принципе этого достаточно, чтобы защитить наш исходный код. Конечно, способов взлома серверов и баз данных множество, к тому же, некоторые используют в качестве сервера IIS, Samba, X-serv или другие подобные программы, каждая из которых имеет свою методологию защиты данных. Но это уже тема отдельной статьи, а скорее целой научной работы.

Литература:

1. Веллинг Л., Томсон Л. Разработка Web-приложений с помощью PHP и MySQL, 2-ое издание. - М.: Издательский дом «Вильямс», 2003 - 794 с.
2. Bruce J. McLaren. Working With dBASE IV and dBASE III Plus». - New Jersey: Addison-Wesley, 1992 – 449 с.
3. С. Фейерштейн, Б. Прибыл. Oracle PL/SQL для профессионалов. - СПб.: «Питер», 2004 – 941 с.
4. Э. Таненбаум, М. Ван Стен. -Распределенные системы: принципы и парадигмы. - СПб.: «Питер», 2003 - 880 с.
5. Jim Gray. The Revolution in Database Architecture // Microsoft Research, 2004, Technical Report MSR-TR-2004-31, ftp://ftp.research.microsoft.com/pub/tr/TR-2004-31.pdf
6. Кузнецов С. Крупные проблемы и текущие задачи исследований в области баз данных. - Институт системного программирования РАН, 2005, http://utc.jinr.ru/database/articles/problems/
7. Голубятников И. В. Основные принципы проектирования и применения мультимедийных обучающих систем. - М.: «Машиностроение», 1999 - 318 с.
8. Петров А. А. Компьютерная безопасность. Криптографические методы защиты. - М.: ДМК, 2000 – 448 с.