MySQL

MySQL: плюсы и минусы

По нынешним временам MySQL — вещь популярная. Особенно часто эту СУБД применяют в интернет-системах. Но вот что интересно, если задать разработчику вопрос: «Почему вы используете MySQL?» — ответом в большинстве случаев будет: «Потому что он бесплатный». Эта логика всегда казалась мне странной. И я поясню почему.

Допустим, вам нужно приобрести автомобиль. Что бы вы взяли: жигули-девятку или пустую банку из-под кока-колы?

Вроде бы бредовое сравнение, но подождите, в этом вопросе имеется подводный камень. А именно: мы не определили цель приобретения. Если вы хотите использовать автомобиль как средство передвижения, то придется раскошелиться на жигули. А вот если хранить бензин в партиях до 300 мл, возможно, банка будет подходить гораздо больше.

Добавляем, удаляем...

Предупреждаю сразу — не надо искать в бесплатной банке аналогий с MySQL: пример приведен исключительно для того, чтобы показать, насколько неоднозначным может быть процесс принятия решения. К тому же MySQL вовсе не бесплатен, как уверены некоторые. На самом деле коммерческое использование этого продукта стоит денег, и немалых. А некоммерческое...

И тут я должен снова отвлечься, почему-то по жизни мне часто встречаются люди, определяющие стоимость продукта суммой, в которую обходится его установка. Но когда дело доходит до частностей, выясняется, что эксплуатация даже самого бесплатного продукта стоит денег. Это и зарплата системного администратора, и потери из-за ошибок, выяснить природу которых не удастся, ибо техподдержкой бесплатных продуктов, как правило, никто не занимается, и «кривизна», на которую махнули рукой разработчики, так как у них сейчас другие дела и им не до вашего бизнеса, своих забот хватает.

И тем не менее даже это многих не смущает. Может быть, они не так уж и не правы? Может быть. Поэтому давайте посмотрим, что же мы имеем на самом деле.

Попробуем просто поработать с MySQL как с СУБД и сделать с его помощью хоть что-нибудь. Для примера возьмем простейшую, хрестоматийную задачку о складе. Допустим, мы храним некие товары, которые доставляют поставщики. В первом приближении для решения этой задачи достаточно двух таблиц: GOODS и SUPPLIERS. Между ними имеется связь «один ко многим», определяющая конкретного поставщика по его идентификатору в таблице товаров. Другими словами, это внешний ключ, обеспечивающий целостность информации. Теперь возьмем MySQL и попробуем воспроизвести эту схему. И сразу же начнутся сюрпризы.

Вначале мы с удивлением обнаружим, что MySQL внешние ключи не поддерживает. Это означает, что любой желающий, имеющий доступ к БД, сможет удалить сведения о любом поставщике независимо от того, имеются ли его товары на складе. В результате записи в таблице GOODS «подвиснут»: они будут указывать на поставщика, сведений о котором нет.

Авторы MySQL рекомендуют бороться с этой напастью правильно написанными процедурами, реализующими конкретную бизнес-логику. И все было бы хорошо, если бы не призрак «человека с консолью», который, без всякого понятия о существовании правильных процедур, случайно почистит таблицу введением команды DELETE FROM.

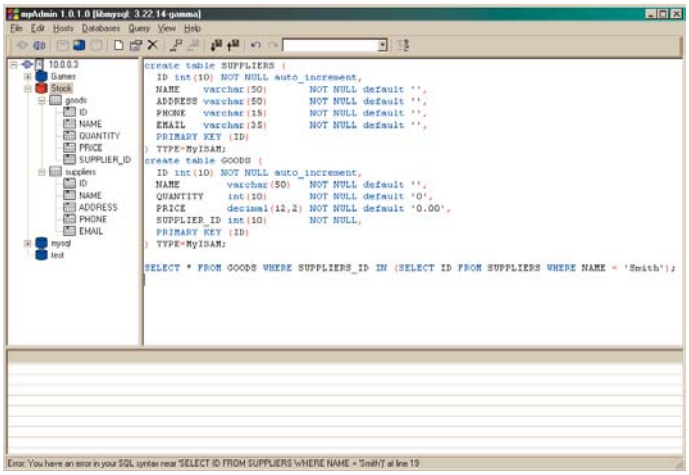
Как известно, против лома нет приема, но до чего же обидно, что в данном случае и лома-то не потребуется.

Но не будем грустить: допустим, наша система надежно защищена от человека с консолью, вокруг сервера стоят вооруженные люди, так что враг не пройдет.

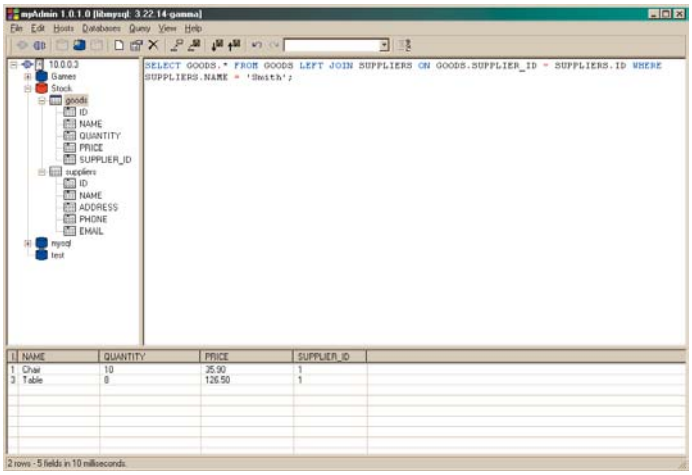
> Не стоит забывать, что тот **тип таблиц**, который MySQL использует по умолчанию, **не поддерживает трансакций**

Итак, мы в безопасности, давайте писать процедуру. Но как мне, наверное, подскажут, процедуру не простую, а хранимую, а еще лучше триггер. Триггер, как известно, срабатывает при наступлении какого-либо события и в данном случае вполне мог бы предотвратить удаление связанных записей... Мог бы. Если бы не одно маленькое но: MySQL не поддерживает триггеров. И хранимых процедур тоже.

Ну хорошо. Не триггер, так программная логика. Мы напишем модуль... на PHP, например (а почему нет?), в котором реализуем такой замысел: «Хотите удалить запись из таблицы поставщиков? Удаляйте сначала »



▲ SELECT из SELECT MySQL считает синтаксической ошибкой



▲ Чтобы выполнить SELECT из SELECT, необходимо преобразовать запрос с использованием JOIN

» все связанные записи из таблицы товаров!» А потом уже поставщика. В смысле, информацию о нем. Логично? Да не то слово.

Поэтому пишем:

```
DELETE FROM GOODS WHERE SUPPLIER_ID =
ID_TODELETE;
DELETE FROM SUPPLIERS WHERE ID =
ID_TODELETE;
```

Но есть один скользкий момент: если после первого DELETE с сервером (не дай бог!) что-то случится, будет беда. Эти два выражения должны обязательно отработать вместе. Или не отработать вообще.

Понятное дело, организовать такой процесс для нас не составляет никакого труда: трансакция, BEGIN, COMMIT и ROLLBACK для нас не пустые слова, и мы их сейчас употребим. Вот только… тот тип таблиц, который MySQL использует по умолчанию, не поддерживает трансакций. Поэтому прежде чем использовать указанный механизм, посмотрите, с таблицами какого типа вы работаете. И не включен ли у вас случайно AutoCommit. Впрочем, о последнем не забывайте и при работе с любой другой СУБД.

Теперь о типах таблиц. MySQL поддерживает их аж шесть различных типов. Это ISAM, HEAP, MERGE, MyISAM, BDB и InnoDB. Четыре первых типа не поддерживают механизм трансакций, причем тип MyISAM используется по умолчанию, а также в тех случаях, когда мы создаем такой тип таблицы, который серверу неизвестен.

Еще один интересный момент: возможность поддержки таблиц типа InnoDB и BDB, умеющих обрабатывать трансакции, зависит от опций, с которыми скомпилирован сервер. Так что если для вас этот вопрос важен, разберитесь с ним сразу, пока поезд еще не ушел.

Перечисленные выше особенности накладывают особый отпечаток и на всю остальную логику. Так, неправильно реализованные процедуры вставки позволят завести информацию о товарах несуществующего поставщика, а один некорректный UPDATE даст возможность вволю поломать голову над тем, например, почему у всего вашего бесконечного списка товаров вдруг оказался один-единственный поставщик. В общем, будьте бдительны.

Проблема выбора

Будем считать, что худо-бедно мы справились с INSERT, DELETE и UPDATE. Теперь наступает очередь SELECT. Казалось бы, уж тут-то каких ждать сюрпризов?

Но вы, наверное, уже поняли, что они будут. И это хорошо. Хорошо не то, что будут сюрпризы, а то, что вы это поняли. Давайте-ка покажем информацию пользователю.

Допустим, нас просят показать все товары, от всех поставщиков, которых зовут Smith. Да ради бога, смотрите, пожалуйста, пользуйтесь:

```
SELECT * FROM GOODS WHERE
SUPPLIERS_ID IN (SELECT ID FROM
SUPPLIERS WHERE NAME = 'Smith');
```

Красиво получается? На самом деле не очень. SELECT из SELECT (то есть выбор из выбранного) не пройдет: MySQL этого не умеет.

И что делать? У вас опустились руки? Не волнуйтесь: выход есть. Хотя уж очень он мне напоминает выход на улицу через окно. Надо, как говорит документация, приводить запросы вот к такой форме:

```
SELECT GOODS.* FROM GOODS LEFT JOIN
SUPPLIERS ON GOODS.SUPPLIER_ID =
SUPPLIERS.ID WHERE SUPPLIERS.NAME =
'Smith';;
```

Пустячок, в общем-то… но почему-то неприятно.

Также неприятно и отсутствие гарантий на то, что установленная версия MySQL поддерживает функции RIGHT и INNER JOIN. И хотя последние версии уже знают, что это такое, убедитесь, понимает ли все это та, с которой вы работаете…

Ладно, теперь, для пользы дела, было бы неплохо сохранить этот (или не этот) замечательный SELECT в качестве представления (VIEW) и обращаться потом к нему… Но как все уже поняли, MySQL не поддерживает VIEW.

И тут меня, видимо, спросят: «Доколе ж?!» — на что я отвечу: «Аккурат до сих пор». Ибо на этом операции с дегтем заканчиваются, и в следующем номере журнала мы поговорим о бочке меда, в которую он лился.

Как видите, у MySQL есть проблемы. С другой стороны, это одна из тех разработок, где они оперативно решаются.

■ ■ ■ Павел Давыдов

Ссылки

Разработчик ▸ The MySQL AB Company

Сайт ▸ http://www.mysql.com

Условия распространения ▸ freeware