

Первый взгляд на MySQL 6

Кирилл Сухов

В апреле 2008 года ввиду активно обсуждавшейся новости о покупке компании MySQL AB корпорацией Sun Microsystems как-то без особого внимания прошло сообщение о том, что активно разрабатываемая версия СУБД 5.2 выделена в отдельную ветку и соответственно переименована в MySQL 6.0. Прошло уже полгода, и это изделие, носящее в данный момент статус альфа-версии, можно рассмотреть и оценить новые возможности и предстоящие изменения.

Что собой представляет самая передовая версия самого распространённого сервера баз данных? Было ли в нововведениях что-то такое, что заставило поменять первую цифру? Ниже приведён краткий обзор, охватывающий основные нововведения MySQL 6.

Новое хранилище данных Falcon

До последнего времени самыми популярными типами таблиц в MySQL остаются MyISAM и InnoDB. Первые отличаются высочайшей производительностью при простых запросах на выборку. Именно эти хранилища данных (а также их предшественники – таблицы ISAM) сделали данную СУБД столь популярной. Недостатком этих таблиц являются всё ещё малые возможности и ограничения. С таблицами InnoDB всё вроде бы гораздо лучше – там нет ограничений MyISAM (таких, например, как блокировки на уровне таблицы при вставке), поддерживаются все типы транзакций. Правда, за всё это приходится платить меньшей производительностью, но основная проблема не в этом.

В октябре 2005 года разработчик InnoDB, компания Innobase, бы-

ла приобретена корпорацией Oracle, и дальнейшее развитие этого движка лежит теперь в русле проприетарной лицензии.

Альфа-версия хранилища данных Falcon, призванного заменить InnoDB, должна была появиться в MySQL 5.2, но в связи с изменениями в нумерации версий доступна теперь в ветке 6.0.x.

При разработке нового хранилища данных ставилась задача преодолеть функциональные ограничения таблиц MyISAM, не потеряв при этом в производительности. Было запланировано реализовать следующий функционал:

- Хранение всех типов данных MySQL.
- Технология Multi Version Concurrency Control (MVCC), позволяющая строкам и таблицам обновляться без блокировки уровня строк. Практически это позволяет сильно уменьшить или полностью устранить блокировку при обновлениях.
- Система «гибких» (flexible) блокировок, включающая гибкую установку уровня блокировки и распознавание deadlock-ситуаций.
- Оптимизация под современные процессоры с поддержкой много-

поточности и оптимизации для работы с большим количеством оперативной памяти.

- Полная (полностью поддерживаемая ACID-стандарт) поддержка транзакций, поддержка конкурентных транзакций.
- Расширенные возможности работы с B-Tree-индексами.
- Возможность хранить информацию на диске в сжатом виде, осуществляя компрессию и декомпрессию «на лету».
- Интеллектуальное управление размещением данных на диске. Дисковое пространство под файлы данных и логов может быть автоматически перераспределено.
- Кэширование данных и индексов.
- Неявные точки сохранения (Implicit savepoints) – автоматические точки сохранения, гарантирующие целостность данных.

Полный список возможностей хранилища данных Falcon доступен в руководстве MySQL 6 (<http://dev.mysql.com/doc/refman/6.0/en/se-falcon-features.html>).

Немного о грустном. Далеко не все эти функции реализованы в текущей версии Falcon, и какие из них так и ос-

танутся только в проекте к моменту релиза шестой версии, пока неизвестно.

Хранилище данных Maria

Разработка этого движка велась одним из создателей MySQL, Майклом Вайдениусом (Michael Widenius), около трёх лет и была прямо вызвана действиями Oracle (по покупке Innobase). Тем не менее таблицы Maria стали логическим развитием таблиц MyISAM, расширенной версией которых данный движок по сути и является.

Главное, что отличает «Марию» от своего прародителя, это наличие средств сохранения целостности данных после краха. Реализовано это посредством ведения лога операции: после катастрофы производится откат результатов выполнения текущей операции или возврат в состояние до команды LOCK TABLES.

Использование лога операций даёт возможность восстановления состояния БД из любой точки лога, и это восстановление может быть произведено после любых изменений таблиц, включая операции CREATE/DROP/RENAME/TRUNCATE. Естественно, за использование лога приходится платить производительностью, поэтому планируется реализовать два режима операций – транзакционный и режим работы без отражения в логе транзакций. Последний может быть использован для некритичных данных.

Ведение лога позволяет создавать инкрементные бэкапы таблиц посредством его периодического копирования.

Кроме того, в новом движке введено расширение для всех типов данных MyISAM, формата rows-in-block, которое использует страничный механизм хранения данных, для кэширования данных в столбцах. Блочный механизм хранения строк позволяет кэшировать данные построчно.

Размер страницы данных в новом хранилище составит 8 Kb (против 1 Kb в MyISAM), что позволяет достичь более высокой производительности для индексов по полям фиксированного размера.

Резервное копирование и схема данных

В новой версии СУБД появились команды BACKUP DATABASE и RESTORE, предназначенные соответственно для резервного копирования и восстановления базы данных.

Наверное, это нововведение было не менее ожидаемым, чем новые хранилища данных. Первая из этих команд позволяет сделать резервную копию одной или нескольких баз данных на определенный момент времени, вторая восстанавливает состояния БД на указанный момент. Восстановление из резервной копии может быть проведено одной командой с использованием бинарного лога для прояснения момента возможной потери данных. Выполнение команд BACKUP DATABASE и RESTORE может быть проведено без перевода базы данных в режим offline или отключения работающих клиентов. Разумеется, BACKUP DATABASE блокирует такие операции, как drop table, но общее число блокируемых команд сведено к минимуму. В общем случае все блокировки касаются операторов языка определения данных – Data Definition Language (DDL). Команда RESTORE блокирует несколько больше операций, так как она перезаписывает данные и контекст работы базы данных.

Важно, что команды BACKUP DATABASE и RESTORE работают независимо от типа таблиц. Рассмотрим примеры работы этих команд.

Создание резервной копии:

```
BACKUP DATABASE test_db TO '/tmp/backupfile';
```

Восстановление:

```
RESTORE FROM '/tmp/backupfile';
```

Работа с несколькими базами данных:

```
BACKUP DATABASE db1, db2 TO '/tmp/db1-db2.backup';
BACKUP DATABASE * TO '/tmp/all.backup';
```

В команде BACKUP DATABASE предусмотрена опция WITH COMPRESSION, позволяющая получать резервные копии в архивированном виде. Опция COMPRESSION_ALGORITHM позволяет указать алгоритм сжатия (по умолчанию – gzip, к сожалению, на настоящий момент он же и единственный доступный).

В настоящее время команды BACKUP DATABASE и RESTORE имеют следующие ограничения:

- BACKUP DATABASE не включает в резервную копию временные таблицы. Поддержка резервной копии табличного пространства в таблицах типа ограничена типом таблиц Falcon.
- Резервное копирование не касается баз данных MySQL и INFORMATION_SCHEMA.

При использовании этих команд надо ясно осознавать, что нет прямого способа расшифровать данные, полученные в результате резервного копирования, кроме их восстановления командой RESTORE. Таким образом, вы просто не имеете возможность узнать, что же именно у вас сохранилось.

Команды BACKUP DATABASE и RESTORE не могут быть использованы в хранимых процедурах, функциях или триггерах. Они также не могут быть использованы как подготовленные выражения (prepared statements).

Интересные изменения произошли в базе данных INFORMATION_SCHEMA, которая впервые появилась в этой версии СУБД. Это виртуальная БД, являющаяся набором представлений (views) к системной базе данных и предоставляющая доступ к метаданным баз данных, то есть к информации об их структуре, объектах (таблицах, представлениях, хранимых процедурах и т. д.) и типах данных. Теперь она дополнена таблицей PARAMETERS, содержащей данные о параметрах и возвращаемых значениях функций, а также о параметрах хранимых процедур. Информация о параметре представлена в той же форме, как и в поле param_list таблицы mysql.proc.

Второе дополнение касается таблицы ROUTINES, содержащей информацию о хранимых процедурах и функциях. В неё добавлены новые поля:

- DATA_TYPE;
- CHARACTER_MAXIMUM_LENGTH;
- CHARACTER_OCTET_LENGTH;
- NUMERIC_PRECISION;

- NUMERIC_SCALE;
- CHARACTER_SET_NAME;
- COLLATION_NAME.

Все они содержат информацию о значении, возвращаемом функцией.

Поддержка XML

Пока ещё скромный XML-функционал дополнен новой командой LOAD XML, загружающей данные из XML-файла в таблицу БД. Команда дополняет уже имеющуюся в MySQL функцию XML-вывода, пример использования:

```
shell> mysql --xml -e 'SELECT * FROM mytable' > file.xml
```

При выполнении команды LOAD XML INFILE по умолчанию тег <row> интерпретируется как описание записи таблицы БД, но этот параметр может быть изменён опцией ROWS IDENTIFIED BY.

Команда LOAD XML воспринимает три различных формата XML:

- Имя атрибута тега как имя колонки (поля) и значение атрибута как значение поля.

```
<row column1="value1" column2="value2" .../>
```

- Имя тега как имя поля и содержимое тега как значение.

```
<row>
<column1>value1</column1>
<column2>value2</column2>
</row>
```

- Атрибут name тега <field> как имя колонки и его содержимое как значение.

Рассмотрим пример работы команды (из руководства), файл test.xml:

```
<?xml version="1.0"?>
<list>
  <person person_id="1" fname="Pekka" lname="Nousiainen"/>
  <person person_id="2" fname="Jonas" lname="Oreland"/>
  <person person_id="3"><fname>Mikael</fname><lname>
    Ronström</lname></person>
  <person person_id="4"><fname>Lars</fname><lname>
    Thalmann</lname></person>
  <person><field name="person_id">5</field>
    <field name="fname">Tomas</field>
    <field name="lname">Ulin</field></person>
  <person><field name="person_id">6</field>
    <field name="fname">Martin</field>
    <field name="lname">Sköld</field></person>
</list>
```

Сначала необходимо создать таблицу:

```
CREATE TABLE person (
  person_id INT NOT NULL PRIMARY KEY,
  fname VARCHAR(40) NULL,
  lname VARCHAR(40) NULL,
  created TIMESTAMP
);
```

Теперь загружаем в таблицу значения из XML-файла (поменяв тег записи по умолчанию):

```
mysql> LOAD XML LOCAL INFILE 'person.xml'
-> INTO TABLE person
-> ROWS IDENTIFIED BY '<person>';
```

И смотрим результат:

```
Query OK, 6 rows affected (0.00 sec)
Records: 6 Deleted: 0 Skipped: 0 Warnings: 0
```

Что ещё?

- Введена поддержка дополнительных кодировок Unicode: utf16, utf32, и 4-байтной utf8.
- Появилась поддержка внешних комментариев для таблиц, колонок и индексов.
- Команда RESET SLAVE больше не сбрасывает параметры соединения, как это было с момента её введения.
- Устранена логическая ошибка – команда LOCK TABLES для транзакционных таблиц больше не подтверждает транзакцию автоматически.
- Кроме того, ряд изменений и дополнений коснулся кластерного варианта сервера MySQL.

Полный список нововведений можно посмотреть в документации (<http://dev.mysql.com/doc/refman/6.0/en/mysql-nutshell.html>).

Что останется в прошлом?

Как всегда, при переходе к новой версии продукта чем-то приходится жертвовать, а кое-что и сознательно отвергать. Ниже приведён список устаревших конструкций, которым не нашлось места в новой версии сервера и их замены:

- Ушла в прошлое системная переменная table_type (теперь есть аналог storage_engine).
- Опция TYPE для хранилища данных, применяемая в командах CREATE TABLE и ALTER TABLE, заменена опцией ENGINE.
- Вместо SHOW TABLE TYPES теперь предполагается использовать SHOW ENGINES.
- Вместо переменной log_bin_trust_routine_creators теперь используем log_bin_trust_function_creators.
- Отменён формат команды TIMESTAMP(N).
- Туда же попали команды SHOW INNODB STATUS и SHOW MUTEX STATUS (замена – SHOW ENGINE INNODB STATUS).
- Не используются больше LOAD TABLE ... FROM MASTER и LOAD DATA FROM MASTER.
- Отменена команда SHOW PLUGIN. Теперь следует использовать SHOW PLUGINS.
- Отменены все опции установки параметров репликации вида: --master-xxx, вместо них теперь используется команда CHANGE MASTER.

Вот собственно и всё, что нам пока обещано в новой версии популярной СУБД. Впрочем, можно заметить, что этого уже очень немало. Разработчики MySQL делают всё, чтобы их продукт не только не отставал от времени, но и стремительно догонял своих более солидных конкурентов.

Подождём выхода релиза MySQL 6 и оценим, насколько это им удалось. 