Новые возможности MySQL 5.1 и 6.0

Дмитрий Ленев, Константин Осипов
Software Developers, MySQL

Корпоративные базы данных-2008

Фундамент:

существующие возможности 5.0

- **хранимые процедуры, функции, триггеры**
- **представления (views)**
- **INFORMATION_SCHEMA.* таблицы**
- **типы таблиц/storage engines**
- **NDB – in-memory/shared nothing кластер**

Статус **5.1** и **6.0**

- 5.1
  - Release Candidate, устранение ошибок
  - Release/GA в ближайщее время (Q2/08)
- 6.0
  - Alpha (Falcon-beta), активная разработка новых возможностей
  - Release/GA в конце года (Q4/08)

Основные направления разработки **5.1**

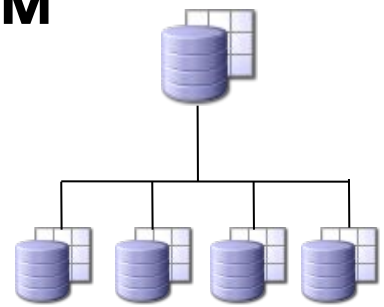
- **удобство администрирования** (partitioning, events, log tables)
- **высокая доступность** (NDB replication, NDB disk-data, row-based replication)
- **большие объёмы данных** (partitioning, NDB disk data)
- **производительность** (partitioning)

Удобство администрирования:

- **partitioning**
- **диспетчер событий (events)**
- **новые возможности аудита и отладки (information_schema.processlist, log tables)**

Partitioning: данные и индексы

- предназначен для управления большими объемами данных
- повышение скорости доступа к данным
- простота управления дисковым пространством



- поддержка range, hash, key, list и составных способов разбиения
- все storage engines

Partitioning: примеры

- разбиение по диапазону – подходит для хранения исторической информации или других слабо связанных данных

```
CREATE TABLE RANGE_BY_DATE(
    CUSTOMER_NUMBER int                 NOT NULL,
    CUSTOMER_ORDER VARCHAR(50)          NOT NULL,
    CUSTOMER_ORDER_DATE DATETIME        NOT NULL)
PARTITION BY RANGE(YEAR(CUSTOMER_ORDER_DATE))
(
PARTITION P1 VALUES LESS THAN (2000),
PARTITION P2 VALUES LESS THAN (2003),
PARTITION P3 VALUES LESS THAN (2005),
PARTITION P4 VALUES LESS THAN MAXVALUE
);
```

Partitioning: примеры

- по значению хэш-функции – для равномерного распределения данных среди нескольких физический устройств и повышения эффективности ввода-вывода. Хэш функция должан быть выбрана верно!

```
CREATE TABLE HASH_EXAMPLE (col1 INT, col2 CHAR(5),
   col3 DATE)
PARTITION BY HASH(col1)
(
PARTITION P1 DATA DIRECTORY = '/.../mysql51/data',
PARTITION P2 DATA DIRECTORY = '/.../mysql51/data2',
PARTITION P3 DATA DIRECTORY = '/.../mysql51/data3',
PARTITION P4 DATA DIRECTORY = '/.../mysql51/data4'
);
```

Partitioning: примеры

- по первичному ключу – используется если данные ключа распределены равномерно.

```
CREATE TABLE HASH_EXAMPLE (col1 INT primary key,
  col2 CHAR(5), col3 DATE)
PARTITION BY KEY(col1)
(
PARTITION P1 DATA DIRECTORY = '/.../mysql51/data',
PARTITION P2 DATA DIRECTORY = '/.../mysql51/data2',
PARTITION P3 DATA DIRECTORY = '/.../mysql51/data3',
PARTITION P4 DATA DIRECTORY = '/.../mysql51/data4'
)
;
```

Partitioning: примеры

- по списку значений – позволяет с точностью указать распределение данных для каждого значения

```
CREATE TABLE LIST_BY_AREA
(
    STORE_NUMBER   int                    NOT NULL,
    STORE_LOCATION int                    NOT NULL,
    ROLLUP_DATE    DATE                   NOT NULL,
    STORE_RECEIPTS DECIMAL(10,2)          NOT NULL)
PARTITION BY LIST(STORE_LOCATION)
(
PARTITION P1 VALUES IN (1,2),
PARTITION P2 VALUES IN (3),
PARTITION P3 VALUES IN (4,5)
)
```

Partitioning: примеры

- возможность создания вложенных разбиений

```
CREATE TABLE SUB_EXAMPLE
(
    CUSTOMER_NUMBER INT              NOT NULL,
    CUSTOMER_ORDER VARCHAR(50)       NOT NULL,
    CUSTOMER_ORDER_DATE DATETIME     NOT NULL,
    CUSTOMER_SERVICE_REGION INT      NOT NULL)
PARTITION BY RANGE(YEAR(CUSTOMER_ORDER_DATE))
SUBPARTITION BY HASH(CUSTOMER_SERVICE_REGION)
(
PARTITION P1 VALUES LESS THAN (2000) (SUBPARTITION S0,SUBPARTITION S1),
PARTITION P2 VALUES LESS THAN (2003) (SUBPARTITION S2,SUBPARTITION S3),
PARTITION P3 VALUES LESS THAN (2005) (SUBPARTITION S4,SUBPARTITION S5),
PARTITION P4 VALUES LESS THAN MAXVALUE (SUBPARTITION S6,SUBPARTITION S7)
)
;
```

Partitioning: мета-информация

```
mysql> desc partitions;
+-------------------------------+-------------+------+-----+---------+-------+
| Field                         | Type        | Null | Key | Default | Extra |
+-------------------------------+-------------+------+-----+---------+-------+
| TABLE_CATALOG                 | varchar(512) | YES  |     | NULL    |       |
| TABLE_SCHEMA                  | varchar(64)  | NO   |     |         |       |
| TABLE_NAME                    | varchar(64)  | NO   |     |         |       |
| PARTITION_NAME                | varchar(64)  | YES  |     | NULL    |       |
| SUBPARTITION_NAME             | varchar(64)  | YES  |     | NULL    |       |
| PARTITION_ORDINAL_POSITION    | bigint(21)   | YES  |     | NULL    |       |
| SUBPARTITION_ORDINAL_POSITION | bigint(21)   | YES  |     | NULL    |       |
| PARTITION_METHOD              | varchar(12)  | YES  |     | NULL    |       |
| SUBPARTITION_METHOD           | varchar(5)   | YES  |     | NULL    |       |
| PARTITION_EXPRESSION          | longtext     | YES  |     | NULL    |       |
| SUBPARTITION_EXPRESSION       | longtext     | YES  |     | NULL    |       |
| PARTITION_DESCRIPTION         | longtext     | YES  |     | NULL    |       |
| TABLE_ROWS                    | bigint(21)   | NO   |     | 0       |       |
| AVG_ROW_LENGTH                | bigint(21)   | NO   |     | 0       |       |
| DATA_LENGTH                   | bigint(21)   | NO   |     | 0       |       |
| MAX_DATA_LENGTH               | bigint(21)   | YES  |     | NULL    |       |
| INDEX_LENGTH                  | bigint(21)   | NO   |     | 0       |       |
| DATA_FREE                     | bigint(21)   | NO   |     | 0       |       |
| CREATE_TIME                   | datetime     | YES  |     | NULL    |       |
| UPDATE_TIME                   | datetime     | YES  |     | NULL    |       |
| CHECK_TIME                    | datetime     | YES  |     | NULL    |       |
| CHECKSUM                      | bigint(21)   | YES  |     | NULL    |       |
| PARTITION_COMMENT             | varchar(80)  | NO   |     |         |       |
| NODEGROUP                     | bigint(21)   | NO   |     | 0       |       |
| TABLESPACE_NAME               | varchar(64)  | NO   |     |         |       |
```

Partitioning: мета-информация

```
mysql> insert into
    test.RANGE_BY_DATE
    VALUES(1,'TEST',NOW());

mysql> insert into
    test.RANGE_BY_DATE
    VALUES(2,'TEST',NOW());

mysql> insert into
    test.RANGE_BY_DATE
    VALUES(3,'TEST',NOW());

mysql> select * from
    partitions where
    table_name =
    'RANGE_BY_DATE'\G
```

```
*********************** 4. row ***********************
                TABLE_CATALOG: NULL
                 TABLE_SCHEMA: test
                   TABLE_NAME: RANGE_BY_DATE
              PARTITION_NAME: P4
           SUBPARTITION_NAME: NULL
  PARTITION_ORDINAL_POSITION: 4
SUBPARTITION_ORDINAL_POSITION: NULL
            PARTITION_METHOD: RANGE
         SUBPARTITION_METHOD: NULL
       PARTITION_EXPRESSION: YEAR(CUSTOMER_ORDER_DATE)
    SUBPARTITION_EXPRESSION: NULL
      PARTITION_DESCRIPTION: MAXVALUE
                 TABLE_ROWS: 3
             AVG_ROW_LENGTH: 24
                DATA_LENGTH: 72
            MAX_DATA_LENGTH: 281474976710655
               INDEX_LENGTH: 1024
                  DATA_FREE: 0
                CREATE_TIME: 2006-02-06 13:12:10
                UPDATE_TIME: 2006-02-06 13:23:36
                 CHECK_TIME: NULL
                   CHECKSUM: NULL
          PARTITION_COMMENT: default
                  NODEGROUP: 0
            TABLESPACE_NAME: default
```

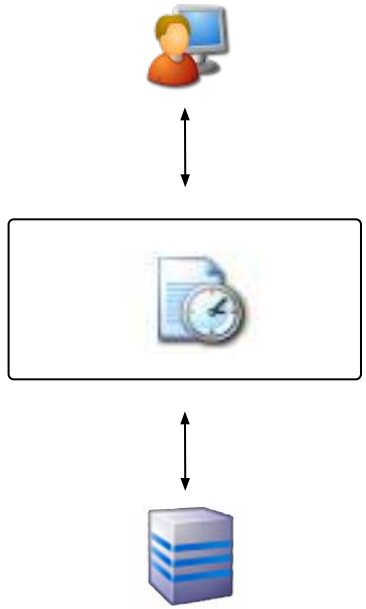
Partitioning: производительность

```
mysql> CREATE TABLE part_tab
    ->  (  c1 int ,c2 varchar(30) ,c3 date )
    ->      PARTITION BY RANGE (year(c3)) (PARTITION p0 VALUES LESS THAN (1995),
    ->      PARTITION p1 VALUES LESS THAN (1996) , PARTITION p2 VALUES LESS THAN (1997) ,
    ->      PARTITION p3 VALUES LESS THAN (1998) , PARTITION p4 VALUES LESS THAN (1999) ,
    ->      PARTITION p5 VALUES LESS THAN (2000) , PARTITION p6 VALUES LESS THAN (2001) ,
    ->      PARTITION p7 VALUES LESS THAN (2002) , PARTITION p8 VALUES LESS THAN (2003) ,
    ->      PARTITION p9 VALUES LESS THAN (2004) , PARTITION p10 VALUES LESS THAN (2010),
    ->      PARTITION p11 VALUES LESS THAN MAXVALUE );
mysql> create table no_part_tab (c1 int,c2 varchar(30),c3 date);
*** Load 8 million rows of data into each table ***
mysql> select count(*) from no_part_tab where c3 > date '1995-01-01' and c3 < date '1995-12-31';
+----------+
| count(*) |
+----------+
|   795181 |
+----------+
1 row in set (38.30 sec)
mysql> select count(*) from part_tab where c3 > date '1995-01-01' and c3 < date '1995-12-31';
+----------+
| count(*) |
+----------+
|   795181 |
+----------+
1 row in set (3.88 sec)
```

10x ускорение!

Диспетчер событий

- новый тип объекта – событие, EVENT
- позволяет создавать единоразовые или повторяющиеся задачи
- позволяет выполнить запрос, блок SQL или хранимую процедуру
- использует нити для выполнения



- имеется возможность остановить выполняющуюся задачу

Events: пример

- реорганизация таблиц каждое воскресенье в 6 утра

```
DELIMITER //
CREATE EVENT OPTIMIZE_TABLES
ON SCHEDULE EVERY 1 WEEK
STARTS '2006-07-23 6:00:00'
ON COMPLETION PRESERVE
DO
BEGIN
OPTIMIZE TABLE test.table1;
OPTIMIZE TABLE test.table2;
END
//
```

Events: пример

- DBA решает что с 2007 года таблица более не нужна.

```
DELIMITER //
CREATE EVENT DROP_TABLES
ON SCHEDULE
AT TIMESTAMP '2007-01-01 0:01:00'
ON COMPLETION NOT PRESERVE
DO
BEGIN
DROP TABLE test.table2;
END
//
```

Новые возможности аудита

- таблица PROCESSLIST
- таблица general_log, slow_log
- CSV или MyISAM формат *_log таблиц

Таблица INFORMATION_SCHEMA.PROCESSLIST

```
mysql> desc processlist;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| ID      | bigint(4)    | NO   |     | 0       |       |
| USER    | varchar(16)  | NO   |     |         |       |
| HOST    | varchar(64)  | NO   |     |         |       |
| DB      | varchar(64)  | YES  |     | NULL    |       |
| COMMAND | varchar(16)  | NO   |     |         |       |
| TIME    | bigint(4)    | NO   |     | 0       |       |
| STATE   | varchar(30)  | YES  |     | NULL    |       |
| INFO    | varchar(100) | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
```

Лог-таблицы: структура

```
mysql> desc slow_log;
+---------------+--------------+------+-----+-------------------+-------+
| Field         | Type         | Null | Key | Default           | Extra |
+---------------+--------------+------+-----+-------------------+-------+
| start_time    | timestamp    | YES  |     | CURRENT_TIMESTAMP |       |
| user_host     | mediumtext   | NO   |     |                   |       |
| query_time    | time         | NO   |     |                   |       |
| lock_time     | time         | NO   |     |                   |       |
| rows_sent     | int(11)      | NO   |     |                   |       |
| rows_examined | int(11)      | NO   |     |                   |       |
| db            | varchar(512) | YES  |     | NULL              |       |
| last_insert_id| int(11)      | YES  |     | NULL              |       |
| insert_id     | int(11)      | YES  |     | NULL              |       |
| server_id     | int(11)      | YES  |     | NULL              |       |
| sql_text      | mediumtext   | NO   |     |                   |       |
+---------------+--------------+------+-----+-------------------+-------+
mysql> desc general_log;
+--------------+------------+------+-----+-------------------+-------+
| Field        | Type       | Null | Key | Default           | Extra |
+--------------+------------+------+-----+-------------------+-------+
| event_time   | timestamp  | YES  |     | CURRENT_TIMESTAMP |       |
| user_host    | mediumtext | YES  |     | NULL              |       |
| thread_id    | int(11)    | YES  |     | NULL              |       |
```

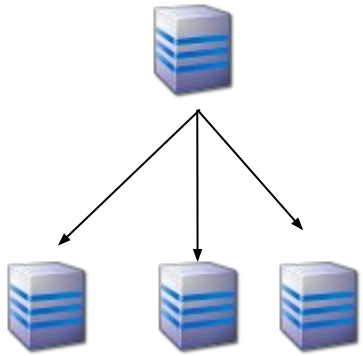
Лог-таблицы: примеры

```
mysql> select query_time, rows_examined, sql_text
    -> from slow_log
    -> order by query_time desc
    -> limit 1\G
*************************** 1. row ***************************
query_time: 00:00:49
rows_examined: 9935
sql_text: SELECT c_custkey, c_name, SUM(l_extendedprice * (1 -
   l_discount)) AS REVENUE,    c_acctbal, n_name, c_address, c_phone,
   c_comment FROM dss_customer, dss_order, dss_lineitem, dss_nation
   WHERE c_custkey = o_custkey     AND l_or derkey = o_orderkey
   AND o_orderdate >= '1993-10-01'          AND o_orderdate <
   '1994-1-01'   AND l_returnflag = 'R` AND c_nationkey = n_nationkey
   GROUP BY c_custkey, c_name, c_acctbal, c_phone,   n_name,
   c_address, c_comment ORDER BY REVENUE DESC
1 row in set (0.00 sec)
```

Высокая доступность(HA)

- новый формат репликации: row-based
- NDB cluster: данные на диске
- NDB cluster: репликация

Новый формат репликации: **row-based**



- опционален – сохранён старый формат
- любые сценарии репликации
- наиболее надёжный
- распространён в большинстве коммерческих СУБД
- возможен смешанный режим, сочетающий преимущества двух форматов

Кластер: данные на диске

- таблицы могут храниться на диске
- управление дисковым пространством при помощи tablespaces
- индексы хранятся в памяти

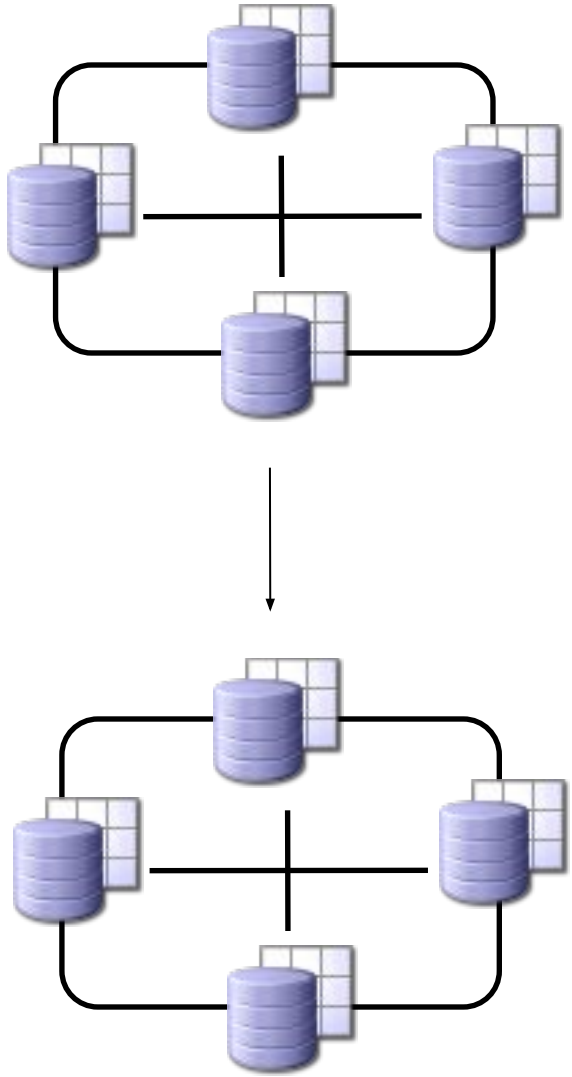
Кластер: определение табличных пространств

```
CREATE TABLESPACE ts1
ADD DATAFILE 'datafile.dat'
USE LOGFILE GROUP lg1
INITIAL_SIZE 12M
ENGINE NDB;


CREATE TABLE t1
(pk1 INT NOT NULL PRIMARY KEY,
   b INT NOT NULL,
   c INT NOT NULL
)
TABLESPACE ts1 STORAGE DISK
ENGINE NDB;
```

Кластер: репликация

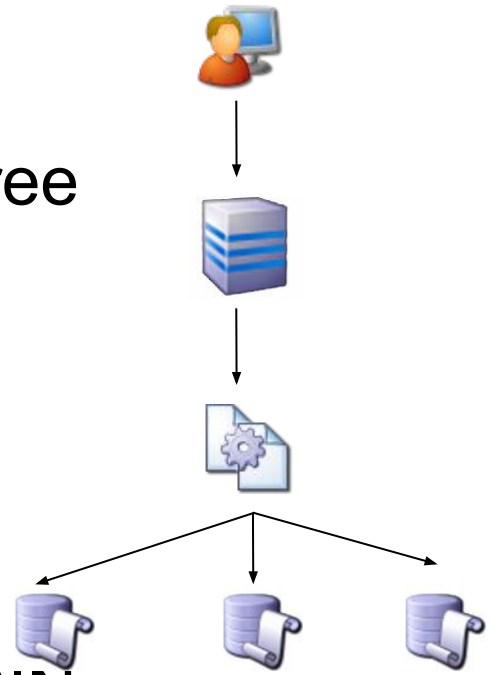
- возможность асинхронной репликации данных из одного кластера в другой
- ограничение: нет поддержки CREATE/ALTER/DROP TABLE



Новые возможности манипуляций с данными:

- **улучшение полнотекстового поиска**
- **XML: поддержка XPath**

Полнотекстовый поиск: **plug-in** архитектура



- пример PLUG-IN парсера включён в main tree
- поддержка списков стоп-слов
- возможность игнорировать регистр
- возможность влиять на релевантность
- режим BOOLEAN теперь по умолчанию
- SHOW PLUGIN, INSTALL/UNINSTALL PLUGIN
- --plugin_dir=path, plugin.h
- документация

Поддержка **XML Xpath**

- XML документ доступен пользователю в виде дерева узлов
- возможность доступа к значениям без пересылки на клиент
- extractValue(), updateXML()

XPath: пример

```
mysql> SELECT extractValue(doc,'/book/author/initial') FROM x;
+-----------------------------------------+
| extractValue(doc,'/book/author/initial') |
+-----------------------------------------+
| CJ |
| J  |
+-----------------------------------------+
2 rows in set (0.01 sec)

mysql> SELECT extractValue(doc,'/book/child::*') FROM x;
+-----------------------------------------+
| extractValue(doc,'/book/child::*') |
+-----------------------------------------+
| A guide to the SQL standard |
| SQL:1999 |
+-----------------------------------------+
2 rows in set (0.00 sec)
```

Пример работы с RSS: извлечение заголовков

```
mysql> select ExtractValue(raw_xml, "/*/channel/title") from
sites_log LIMIT 7;
+-----------------------------------------------------+
| ExtractValue(raw_xml, "/*/channel/ title") |
+-----------------------------------------------------+
| The Motley Fool |
| Nanodot: Nanodot: Nanotechnology News and Discussion of
Emerging Technologies |
| Slashdot: Science |
| Slashdot |
| Slashdot: Book Reviews |
| Slashdot: Features |
| Slashdot: Interviews |
+-----------------------------------------------------+
7 rows in set (0.00 sec)

mysql> INSERT into titles select ExtractValue(raw_xml,
"/*/channel/title") from sites_log;
```

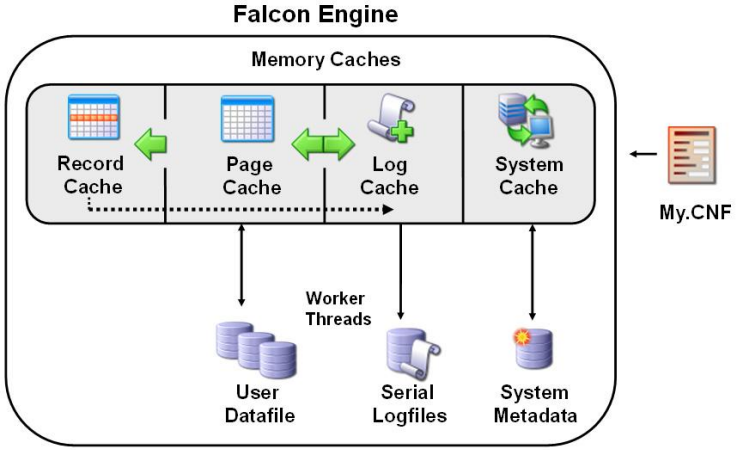
Основные направления разработки **6.0**

- **Falcon storage engine**
- **неблокирующие операции** **(online backup, online ALTER in NDB)**
- **производительность** **(оптимизатор)**

Falcon

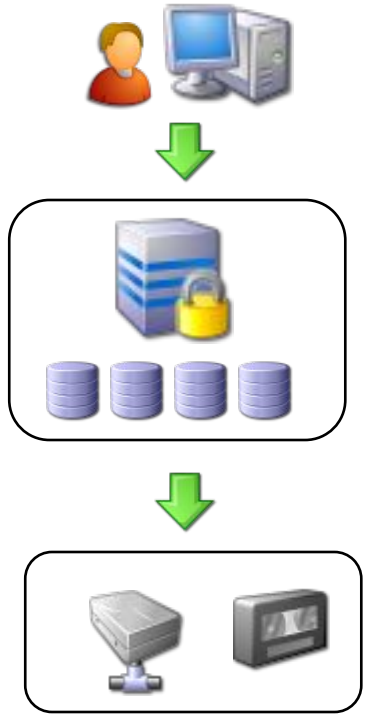
Новый storage engine который:

- Ориентирован на OLTP
- Оптимизирован для современного оборудования
- Реализует ACID транзакции
- Использует MVCC
- Хорошо масштабируется на SMP/Multi-core
- Не замена InnoDB
- Но должен превосходить InnoDB на типичных для MySQL задачах



Online backup

- Поддерживает все основные storage engines
- Не блокирует DML для engines
  - Имеющих native driver (метод backup специфический для engine, e.g. MyISAM)
  - Поддерживающих consistent snapshot (транзакционники – Falcon, InnoDB)
- Управление посредством SQL
- Поддержка PITR
- Возможность расширения за счет plugins



NDB: Online ALTER TABLE

Поддержка неблокирующих для чтения и обновления DDL операции:

- ADD COLUMN
- ADD INDEX
- DROP INDEX

Улучшения в оптимизаторе

Улучшения в реализации подзапросов:
- materialization
- semi-join (различные стратегии)
- FROM-flattening

Другие улучшения (?)

Спасибо!