Valuable MariaDB Features That Somebody Paid for

Igor Babaev, Principal MariaDB Architect
Henrik Ingo, COO

Monty Program sponsor talk at MySQL User Conference 2010
Topics

- Table elimination
- mysqlbinlog improvements
- MyISAM keycache performance improvements
- Future work
Table elimination

- Optimization for queries over highly-normalized data
- Present in “big” databases, like Oracle, SQL Server
- Basic idea
  Detect outer joins that have “unused” inner sides and delete those inner sides

- SELECT tbl1.*
  FROM tbl1 LEFT JOIN tbl2 ON tbl1.id = tbl2.primary_key
  WHERE condition(tbl1.*)
- It is guaranteed that for each record of tbl1
  - tbl2 will have not more than one match (tbl2.primary_key=..)
  - If tbl2 has no match, LEFT JOIN will generate a NULL-record
Highly-normalized data:

actor(name, date_of_birth, rating)

is stored as:

create table ac_anchor(AC_ID int primary key);

cREATE TABLE ac_name(AC_ID int, ACNAM_name char(N), primary key(AC_ID));

cREATE TABLE ac_dob(AC_ID int, ACDOB_birthdate date, primary key(AC_ID));

cREATE TABLE ac_rating(AC_ID int, ACRAT_rating int, ACRAT_fromdate date, primary key(AC_ID, ACRAT_fromdate));
Then select back:

cREATE VIEW actors AS
SELECT
    ac_anchor.AC_ID, ACNAM_Name, ACDOB_birthdate, ACRAT_rating
FROM
    ac_anchor
LEFT JOIN ac_name ON ac_anchor.AC_ID=ac_name.AC_ID
LEFT JOIN ac_dob ON ac_anchor.AC_ID=ac_dob.AC_ID
LEFT JOIN ac_rating ON (ac_anchor.AC_ID=ac_rating.AC_ID AND
    ac_rating.ACRAT_fromdate =
    (SELECT MAX(sub.ACRAT_fromdate)
     FROM ac_rating sub
     WHERE sub.AC_ID=ac_rating.AC_ID))

SELECT ACRAT_rating FROM actors WHERE ACNAM_name='Gary Oldman';
Functional Dependencies (1)

```
select ac_anchor.AC_ID, ACNAM_Name, ACDOB_birthdate, ACRAT_rating
from ac_anchor
  left join ac_name on ac_anchor.AC_ID=ac_name.AC_ID
  left join ac_dob on ac_anchor.AC_ID=ac_dob.AC_ID
  left join ac_rating on (ac_anchor.AC_ID=ac_rating.AC_ID and
  ac_rating.ACRAT_fromdate =
    (select max(sub.ACRAT_fromdate) from ac_rating sub
     where sub.AC_ID=ac_rating.AC_ID))
  where ACNAM_name='Gary Oldman'
=>
select ac_anchor.AC_ID, ACNAM_Name, ACDOB_birthdate, ACRAT_rating
from ac_anchor
  left join ac_name on ac_anchor.AC_ID=ac_name.AC_ID
  left join ac_dob on ac_anchor.AC_ID=ac_dob.AC_ID
  left join ac_rating on (ac_anchor.AC_ID=ac_rating.AC_ID and
  ac_rating.ACRAT_fromdate =
    (select max(sub.ACRAT_fromdate) from ac_rating sub
     where sub.AC_ID=ac_rating.AC_ID))
  where ACNAM_name='Gary Oldman'

ac_ancor.AC_ID  ->  ac_rating.AC_ID           (1)
ac_rating.AC_ID ->  ac_rating.ACRAT_fromdate  (2)

=>  ac_ancor.AC_ID  ->  ac_rating.primary_key(AC_ID,ACRAT_fromdate)
```
Check: columns of the inner table(s) to be eliminated are:

- functionally dependent of some columns of the outer table(s).

Types of dependencies:

- \( a \rightarrow b: \ t2.b = t..a \text{ or } t2.b = t1.a + 1 \)
- \((a,b) \rightarrow c: \ t2.full\_name = \text{concat}(t1.first\_name, t1.last\_name)\)
- \( a \rightarrow (b,c): (t2.b, t2.c) = (\text{select max(b), max(c) from t where t.a=t1.a})\)
- \( t.pk1,..,t.pkm \rightarrow t.c1,...,t.cn \)

Algorithms:

1. Simple Wave Algorithm:
   - Using the basic dependencies iteratively expand the set \( C \) of the columns functionally dependent on the columns from outer tables \( C_0 \) until the set stabilizes.
   - Complexity: \( O(#D*#A) \)

2. More Sophisticated Algorithm:
   - Every time a new column \( c \) is added to the set \( C \) decrement the counters of 'unknown' columns in the dependencies that contain \( c \) as an argument. Apply a dependency when its counter of 'unknown' arguments becomes 0.
   - Complexity: \( O(#E) \)
Table elimination – examples

explain select ACRAT_rating, ACDOB_birthdate from actors where ACNAM_name='Gary Oldman';
+-----------------------------+---------------------------------+-----------------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+
<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>possible_keys</th>
<th>key</th>
<th>key_len</th>
<th>ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>ac_anchor</td>
<td>index</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>NULL</td>
</tr>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>ac_name</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>ac_anchor.AC_ID</td>
</tr>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>ac_dob</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>ac_anchor.AC_ID</td>
</tr>
<tr>
<td>3</td>
<td>DEPENDENT SUBQUERY</td>
<td>sub</td>
<td>ref</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>ac_rating.AC_ID</td>
</tr>
</tbody>
</table>

explain select ACRAT_rating from actors where ACNAM_name='Gary Oldman';
+-----------------------------+---------------------------------+-----------------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+
<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>possible_keys</th>
<th>key</th>
<th>key_len</th>
<th>ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>ac_anchor</td>
<td>index</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>NULL</td>
</tr>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>ac_name</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>ac_anchor.AC_ID</td>
</tr>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>ac_dob</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>ac_anchor.AC_ID</td>
</tr>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>ac_rating</td>
<td>ref</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>ac_anchor.AC_ID</td>
</tr>
<tr>
<td>3</td>
<td>DEPENDENT SUBQUERY</td>
<td>sub</td>
<td>ref</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>ac_rating.AC_ID</td>
</tr>
</tbody>
</table>

explain select ACDOB_birthdate from actors where ACNAM_name='Gary Oldman';
+-----------------------------+---------------------------------+-----------------------------+-------------------+-------------------+-------------------+-------------------+-------------------+-------------------+
<table>
<thead>
<tr>
<th>id</th>
<th>select_type</th>
<th>table</th>
<th>type</th>
<th>possible_keys</th>
<th>key</th>
<th>key_len</th>
<th>ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>ac_anchor</td>
<td>index</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>NULL</td>
</tr>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>ac_name</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>ac_anchor.AC_ID</td>
</tr>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>ac_dob</td>
<td>eq_ref</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>ac_anchor.AC_ID</td>
</tr>
<tr>
<td>1</td>
<td>PRIMARY</td>
<td>ac_rating</td>
<td>ref</td>
<td>PRIMARY</td>
<td>PRIMARY</td>
<td>4</td>
<td>ac_anchor.AC_ID</td>
</tr>
</tbody>
</table>
With table elimination, you can:

- Do normalization on optional/historic data
- Create a denormalized view with LEFT JOINs
- Use this view and get something like “index only” scans.
**mysqlbinlog cmdline tool**

**Master/slave replication**

- **Use cases**
  - Human readable output of contents in binlog
  - Valid SQL, can replay statements
  - Point in Time recovery
  - "Manual replication" to other DB
  - Hacks: You can grep, awk, sed, perl massage the output

**mysqlbinlog cmd line tool**

- SQL
- mysqlbinlog → [grep] → SQL

MariaDB

Monty Program
Customer issues with mysqlbinlog

- MySQL bug #46640 (MWL:50)
  - mysqlbinlog 5.1 output does not apply on MySQL 5.0 server. (BINLOG statement, used for RBR)
  - A "format description" BINLOG statement includes a server_id setting from the original server, which overrides server_id at "Other DB". This should not happen, server_id should remain constant per server. (Prevents circular replication problems.)

- --rewrite-db=name (MWL:36)
  - Background: for statement based replication, customer has scripts to awk statement like "USE dbname" so that the database on target server has a different name. With RBR dbname.tblname is explicitly encoded in BINLOG statement.
  - Corner cases like ALTER DATABASE too!

- Don't write certain tables to binlog. (MWL:40)
  - After extensive architecture study, customer realizes that RBR and TEMPORARY tables does exactly this in MySQL 5.1.
Fixing bug #46640 (MWL:50)

- Root problem: too much sharing of code
  - mysqlbinlog is cmdline utility, produces SQL
  - Replication SQL thread runs in server process, applies binlog events
- Leads to using flags and conditions (if context is a, do x, if b do y...). Or not using them...
  - thd->rli_fake= new Relay_log_info;
  - Sharing code paths in sql/log_event.cc, sql/log_event_old.cc, sql/slave.cc
- Binlog statements are rather opaque...
  - BINLOG '3u9kSBMUAAYQAAAJEBA ...
  - Where is the server_id?
- Fix: Move (replication related) code paths away to be in SQL slave thread only so server_id is not set from MySQL server cmd line.
  (But we still share code, root problem remains?)
- Support new option in client_priv.h, mysqlbinlog.cc
  - Add OPT_REWRITE_DB, parse 'db-from->db-to' syntax, allocate memory...

- RBR: Row operation events are preceded by Table map event(s) which maps table id(s) to database and table names. So, it's enough to support rewriting database name in a Table map.
  - case TABLE_MAP_EVENT:
  - Table_map_log_event::rewrite_db()

- SBR: Limited to rewrite "USE dbname"
  - Ignores INSERT INTO otherdbname.tblname VALUES...
  - Considered a feature, other *-ignore-* options work the same.
  - Query_log_event, Load_log_event, Execute_load_query_log_event, Create_file_log_event
  - print_use_stmt() will figure out correct "USE dbname" to output
mysqlbinlog conclusions

- 2 bugs fixed
- Use --rewrite-db to stream binlog to different or renamed database
- Sometimes you need consulting, not new features :-)
- Omitting some tables from binlog gives performance improvement on master due to less fsync's in writing binlog.

Possible future mysqlbinlog work
- 37: Add an option to mysqlbinlog to produce SQL script with fewer roundtrips
- 38: Make mysqlbinlog not to output unneeded statements when using --database (BUG#23890, BUG#23894)
- 41: Add a mysqlbinlog option to filter certain kinds of statements (filter on INSERT, UPDATE...)
- 45: Add a mysqlbinlog option to produce succint output
- 46: Change BINLOG statement syntax to be human-readable
- 47: Store in binlog text of statements that caused RBR events
- 49: Make --replicate-(do,ignore)-(db,table) behaviour for RBR identical to that of SBR
MyISAM keycache performance improvements

- Keycache = index cache for MyISAM tables
- Before any DDL, flush cache to disk
- \( \rightarrow \) This includes DROP TABLE and DROP DATABASE!

- Big unnecessary performance hit for customer creating and dropping large tables every day
- Fix: Don't flush data to disk only to delete it!
- 24 h work on DROP TABLE, 3 h DROP DATABASE

- Win: much less disk io at customer
Possible future work

- `index_merge` (fair choice between `index_merge` union and range access)
- Streaming `UNION ALL` (without temp table)
- IPv4 datatype
- Microsecond datatype
- Storage Engine API improvements