openark-kit

MySQL utilities for everyday use

Shlomi Noach

openark.org

O'Reilly MySQL Conference & Expo 2011

What is the openark-kit?

- *openark-kit* it a set of tools designed to ease some common MySQL tasks
  - Some of the tools simply automate common tasks
  - Others allow for MySQL auditing
  - Others still introduce new functionality to MySQL.
- All tools are standalone python scripts.
- *openark-kit* is developed and supported on the Linux operating system. There are ports for BSD and OS/X.

Openark-kit is an open source project

- The toolkit is release under the permissive New BSD License.
- Currently hosted by Google Code. Downloads and documentation on:

  http://code.openark.org/forge/openark-kit

  http://code.google.com/p/openarkkit/
- Looking for contributors!

Origin of openark-kit

- *openark-kit* is inspired by popular Maatkit.
- It follows similar naming conventions, command line options names, distribution concept, all in the hope and purpose of making a familiar environment.

Some openark-kit tools

- We discuss a few *openark-kit* tools
- A tool is written to solve a problem.
  - What kind of problems do *openark-kit* tools solve?
- We discuss issues in:
  - Auditing
  - General maintenance
  - Security
  - Massive, blocking operations

oak-hook-general-log

- **Problem**: you wish to log queries which are not using indexes.
  - You set **log_queries_not_using_indexes=1**
  - The slow log gets swamped with queries over very small queries, irrelevant to your problem.
- **Problem**: you wish to audit all queries using temporary tables.
  - This does not mean they're not using indexes
- **Problem**: you wish to audit queries iterating over 100,000 rows.
- **Problem**: you wish to only audit queries using a specific table or a specific index.
- **Problem**: you wish to audit queries answering for a *combination* of the above requirements.
- **Problem**: you wish to audit logins / logouts.

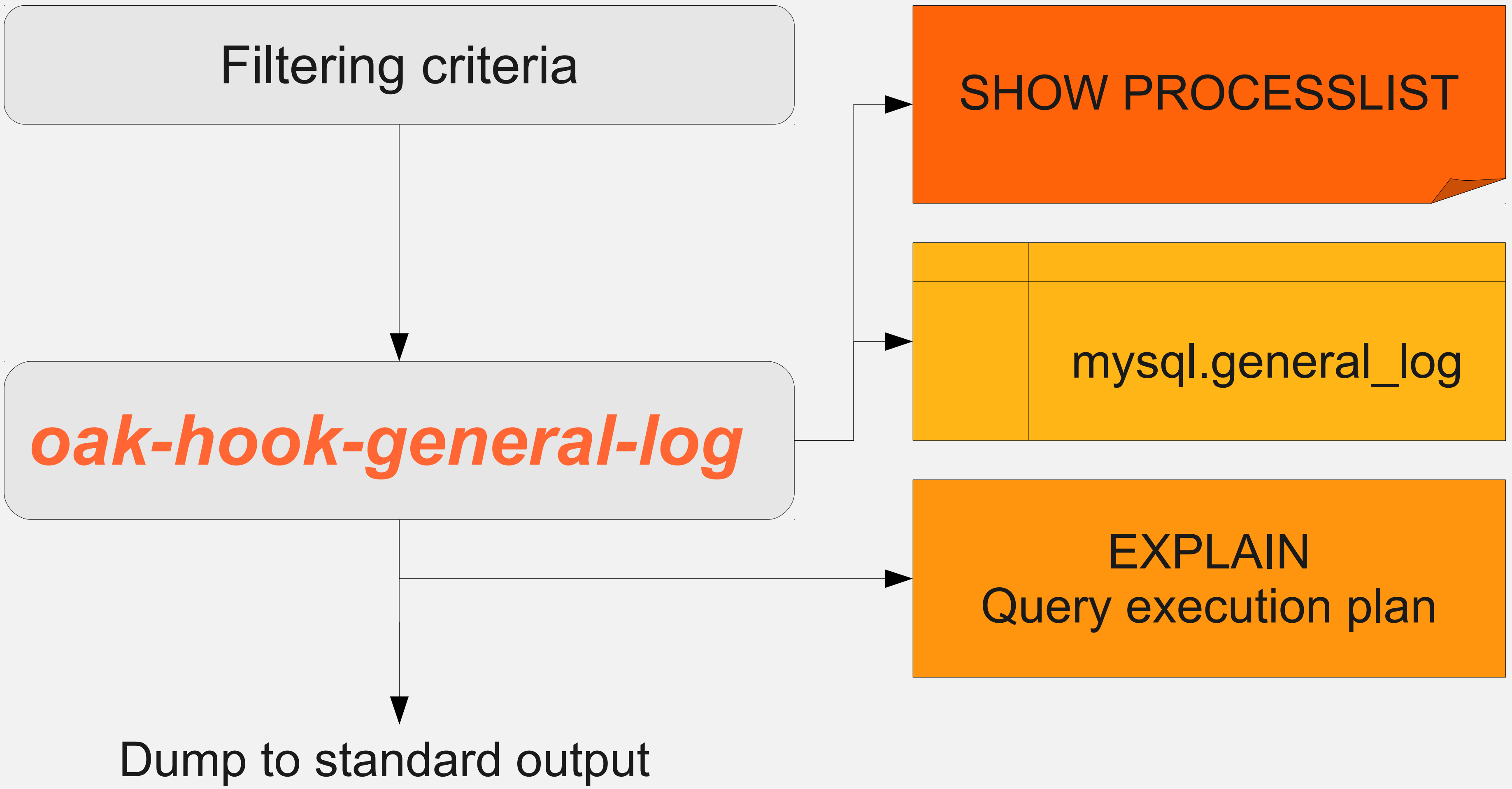
oak-hook-general-log

- *Percona Server* and *MariaDB* answer for most of the above (see Percona's Slow Query Log feature), or otherwise lay the basis for answering additional questions.
- Standard MySQL distribution does not answer for any of the above.
- *oak-hook-general-log* hooks up to a (>= 5.1) MySQL server and audits running queries in near real-time.
- Queries answering for input criteria are dumped to standard output.

oak-hook-general-log

- MySQL's *general log* contains much of the information required to solve above problems.
- General log can be directed at file or log tables, and is mostly turned off due to overwhelming amounts of entries.
- Sadly, it lacks some important information:
  - Both formats neglect to note the database context
  - File format only identifies account on login; but not on queries

oak-hook-general-log



oak-hook-general-log

- *oak-hook-general-log* enables *general log* on log tables for limited period.
- It cross-references log entries with the process list so as to identify database context (with fair chances of success due to asynchronous action).
- It rotates the **general_log** table so as to prevent it from filling up.
- It can evaluate query execution time *on-the-fly.*
- Whether to dump query or not may depend on output of query execution plan.

oak-prepare-shutdown

- Issue: MySQL must be restarted
  - Perhaps to make changes to variables such as **innodb_buffer_pool_size** to take effect.
  - Perhaps files should be moved around.
- **Problem**: restart takes a very long time.
- In the process of shutdown:
  - MySQL rejects any new incoming connections
  - But waits on all pending queries to complete
  - Then, InnoDB must flush dirty pages to disk

oak-prepare-shutdown

- *oak-prepare-shutdown* automates a popular solution:
  - Reduce **innodb_max_dirty_pages_pct** to zero.
  - Follow up on **Innodb_buffer_pool_pages_dirty** until no improvement is observed for 10 successive seconds.
- This allows MySQL to accept connections while flushing dirty pages.
- MySQL will be more I/O bound than before, but still there!

oak-prepare-shutdown

```
$ oak-prepare-shutdown && service mysql stop
-- innodb_buffer_pool_pages_dirty: 79278
-- innodb_buffer_pool_pages_dirty: 28113
-- innodb_buffer_pool_pages_dirty: 1284
-- No improvement from 1284
-- No improvement from 1284
-- No improvement from 1284
…
-- No improvement from 1284
-- No improvement from 1284
-- Found no improvement for 10 successive
attempts. Will now terminate
$ Stopping MySQL.........................[OK]
```

Security

- Openark kit provides with two tools to audit & control some security and privileges issues:
  - oak-security-audit
  - oak-block-account

MySQL security model

- The MySQL security model is a simple hierarchal set of rules.
- GRANTS and passwords are associated on a per-account basis, and are created over the following constructs:
  - Entire domain
  - Databases (schemata)
  - Tables
  - Columns
  - Routines

MySQL security model

- Missing from the model:
  - The *catalog* level, above the schemata level.
  - LDAP/Kerberos integration (*MySQL 5.5 now supports pluggable authentication)
  - Roles
  - ...
- Missing functionality makes for management overhead. More accounts must be created, and explicitly associated with privileges.
- People look for shortcuts, thereby relaxing security.

oak-security-audit

- Common shortcut pattern:

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,EXECUTE,FILE,LOCK TABLES
       ON xampp.* TO 'web_user'@'%.local';

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,EXECUTE,FILE,LOCK TABLES
       ON app.* TO 'web_user'@'%.local';

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,EXECUTE,FILE,LOCK TABLES
       ON interfaces.* TO 'web_user'@'%.local';
… the list goes on …

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,EXECUTE,FILE,LOCK TABLES
       ON analytics.* TO 'web_user'@'%.local';
… and on …

Too much to type. Take a shortcut:
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,EXECUTE,FILE,LOCK TABLES
       ON *.* TO 'web_user'@'%.local';
```

oak-security-audit

- Many are familiar with the *mysql_secure_installation* tool.
- *oak-security-audit* brings much more to the table. Among other tests, it will:
  - Check for non-local root accounts, anonymous users, wild card host accounts
  - Look for accounts with empty passwords (implies no password required)
  - Look for different users sharing identical passwords
  - Report non-root accounts with complete grants; administrative privileges; write access to the *mysql* schema
  - Test general settings: look for **sql_mode** settings, **old_passwords** use.

oak-security-audit

```
$ oak-security-audit

-- Auditing in strict level
-- The following users are assumed as root: root
--
-- Looking for non local 'root' accounts
-- -------------------------------------
-- Found 1 non local 'root' accounts. Recommended actions:
RENAME USER 'root'@'remote' TO 'root'@'localhost';
--
-- Looking for anonymous user accounts
-- -----------------------------------
-- Passed
--
-- Looking for accounts accessible from any host
-- ---------------------------------------------
-- Found 1 accounts accessible from any host. Recommended actions:
RENAME USER 'foo'@'%' TO 'foo'@'<specific host>';
-- ...
```

oak-block-account

- Most user authenticated systems have some form of user access blocking.
  - Due to repeating failed login attempts
  - Due to failed payment
  - Due to request for account freeze
- MySQL has no such notion.
- The mere fact an account exists allows for user login.
- No

```
GRANT/REVOKE login ON *.*
```

oak-block-account

- We may wish to temporarily block an account due to abuse suspicion.
- Perhaps we wish to disable some modules in our system, which are not completely under our control.
- *oak-block-account* works around this limitation by changing account's password in such way that:
  - The account becomes inaccessible (no password will gain access).
  - The block is easily reversible.
  - At any point it is easy to deduce whether an account is blocked or not.

oak-block-account

```
root@mysql-5.1.51> SELECT user, host, password FROM mysql.user;
+-------------+-----------+-------------------------------------------+
| user        | host      | password                                  |
+-------------+-----------+-------------------------------------------+
| shlomi      | localhost | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
| replication | 10.0.0.%  | *35CE0EA4EA6C6E962A01F70C121071AE5D38B517 |
+-------------+-----------+-------------------------------------------+


$ oak-block-account --account-user=shlomi
    --account-host=localhost --block


root@mysql-5.1.51> SELECT user, host, password FROM mysql.user;
+-------------+-----------+-------------------------------------------+
| user        | host      | password                                  |
+-------------+-----------+-------------------------------------------+
| shlomi      | localhost | 9DA2AC2DE76CD7ADD8654EE50192347BE7384BB6* |
| replication | 10.0.0.%  | *35CE0EA4EA6C6E962A01F70C121071AE5D38B517 |
+-------------+-----------+-------------------------------------------+
```

oak-chunk-update

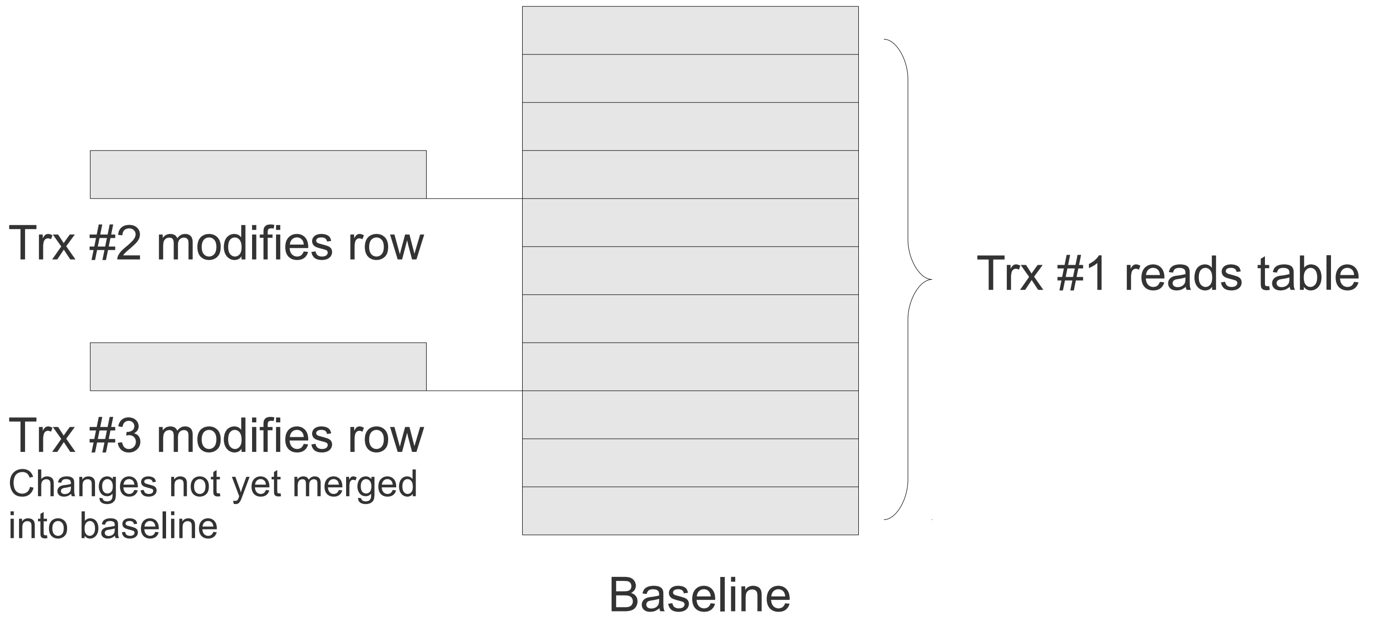
- **Problem**: company wishes to create a denormalized table, the combination of three large, static tables.
  - To populate new table, they issue:

```
INSERT INTO new_table
SELECT … FROM t1 JOIN t2 ON (…) JOIN t3 ON (…)
```

  - Three days later, they give up and hit *Ctrl*+C
- InnoDB transaction becomes huge.

oak-chunk-update

- InnoDB uses *MVCC* (Multi Version Concurrency Control) to manage concurrency, as well as allow for non blocking selects.
- A row may have concurrent versions of data.



oak-chunk-update

- Long running transactions make for increasing number of non-merged versions, and eventually to increased locks.
- A long running transaction may be aborted at the last moment, in which case it must be able to rollback. It must store original data while manipulating it.
- A long running transaction will have to resort to disk at some point.
- *oak-chunk-update* breaks down queries to smaller chunks, executed in smaller transactions, with optional sleep time.
- Also note Maatkit's *mk-archiver* tool, for archiving/purging table rows.

oak-chunk-update

- In simplest invocation magic begins with query modification:

```
$ oak-chunk-update --execute="INSERT INTO new_table
  SELECT … FROM t1 JOIN t2 ON (…) JOIN t3 ON (…)
  WHERE OAK_CHUNK(t1)"
```

- The tool will break translate this query into many queries of the form:

```
INSERT INTO new_table
  SELECT … FROM t1 JOIN t2 ON (…) JOIN t3 ON (…)
  WHERE (t1.col >= …) AND (t1.col < …)
```

oak-chunk-update

- How does it work?
- *oak-chunk-update* requires a UNIQUE KEY on one of the tables. PK is best, others possible.
- It will automatically split (chunk) the table into smaller portions, e.g. of 1,000 rows, in ascending key order.
- It will execute the query with WHERE clause limiting to said rows.
- The key may actually be compound (over several columns)

oak-chunk-update

- A scarier example:

```
$ oak-chunk-update -d sakila -e "UPDATE film_actor SET
last_update = DATE(last_update) WHERE OAK_CHUNK(film_actor)"

UPDATE film_actor SET last_update = DATE(last_update)
WHERE
  (((film_actor.actor_id > @unique_key_range_start_0) OR
  (((film_actor.actor_id = @unique_key_range_start_0)) AND
  (film_actor.film_id > @unique_key_range_start_1)))
AND
  ((film_actor.actor_id < @unique_key_range_end_0) OR
  (((film_actor.actor_id = @unique_key_range_end_0)) AND
  (film_actor.film_id < @unique_key_range_end_1)) OR
  ((film_actor.actor_id = @unique_key_range_end_0) AND
  (film_actor.film_id = @unique_key_range_end_1))))
```

oak-chunk-update

- Common usage:
  - Routine purging of old data
  - Copying data between tables
  - Updating data for a newly created column
  - Queries which are just too large for single transactions
- Benefits:
  - Smaller, faster transactions
  - Optional sleep time allows for spreading of total runtime, with chance for replication to catch up.
  - Optional hints to limit range, or auto-stop execution.

oak-online-alter-table

- **BIG problem**: you want to refactor a table; say, add a column:

```
ALTER TABLE forum_message ADD COLUMN is_private TINYINT;
```

- MySQL will lock down the table. No reads, no writes. Not even metadata.
- Effectively, on a “popular” table, this means database lockdown.

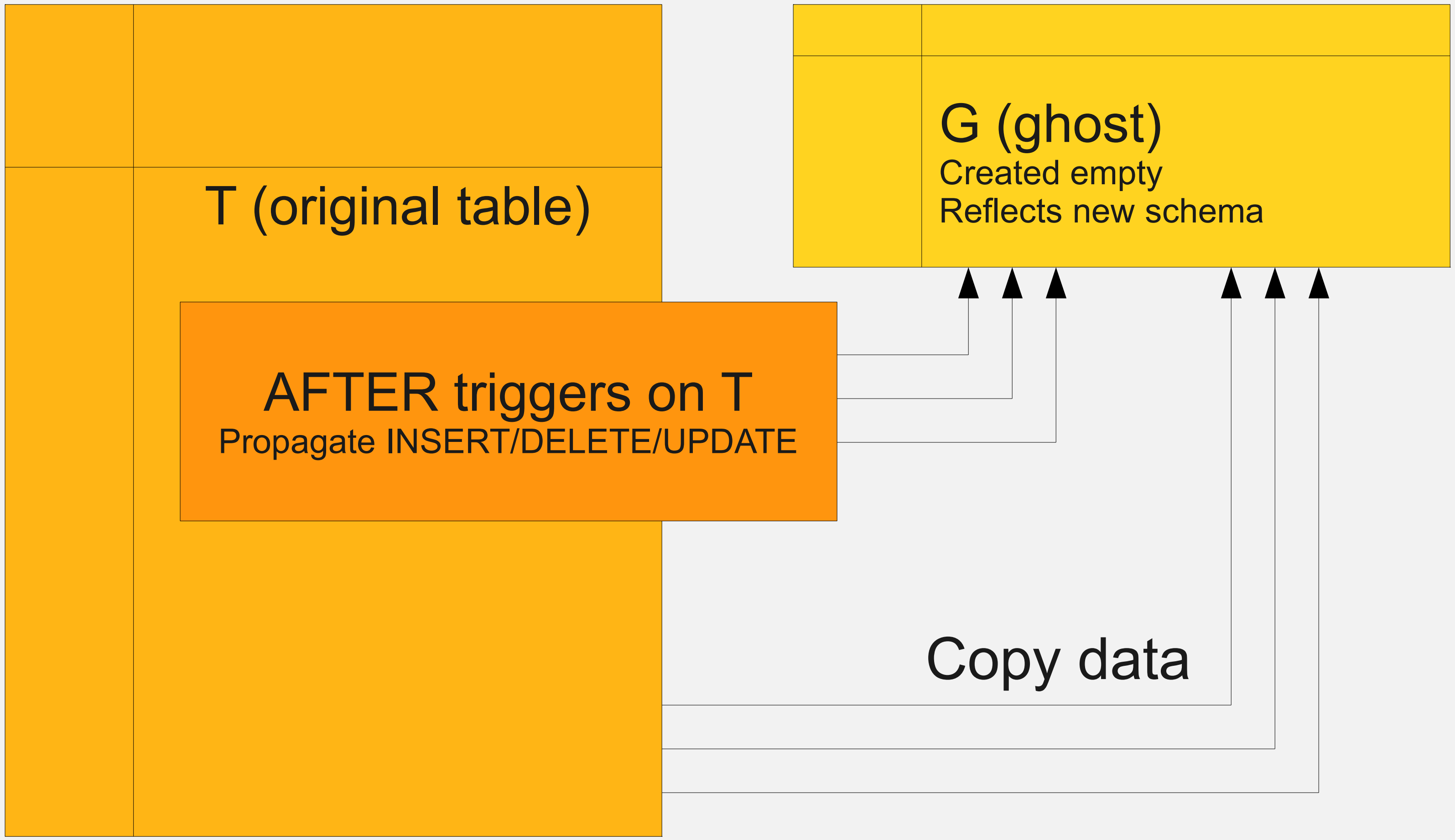
oak-online-alter-table

- Possible solution: use replication
  - Make **ALTER TABLE** on slave
  - Upgrade slave to master
  - Build new replication slave
- Better, use Master-Master replication
  - *MMM for MySQL* automates much of the plumbing.
- Cons:
  - You need additional servers
  - These servers will be (probably) inaccessible due to **ALTER TABLE** invocation.
  - Application will have to fail-over to secondary servers

oak-online-alter-table

- *oak-alter-table* uses similar approach of *oak-chunk-update* in breaking up your query into chunks.
- How can you split an **ALTER** statement?
  - By *simulating* it
  - Create a new, empty, “ghost” table
  - Execute **ALTER** on ghost table
  - Slowly synchronize between original table (T) and ghost table (G)
  - Throw away original table, rename ghost in its place.

oak-online-alter-table



oak-online-alter-table

- *"Slowly synchronize between original table (T) and ghost table (G)"*. How?
  - A lot of "magic" in onto play.
  - *oak-online-alter-table* creates "AFTER" triggers on table T.
  - Triggers propagate INSERT, UPDATE, DELETE statements onto G, in such way that they are ensured to succeed.
  - Getting range snapshot of T's unique key (i.e. PRIMARY KEY or other), the tool chunks that range (à la *oak-chunk-update*), and copies chunked rows to G.
  - Meanwhile, queries on T may actually modify or delete such rows. With relatively small, quick locks this concurrency problem can be solved.

oak-online-alter-table

- Note:
  - The tool is experimental!
  - Use of triggers makes for noticeable impact on overall performance.
  - Current limitation:
    - No support for foreign keys (can be lifted on child-side)
    - No AFTER triggers may exist (can be lifted in MySQL **5.1**)
  - Other alternatives exist today, based on this tool.

Other noteworthy tools

- *oak-repeat-query*: repeat execution of a given query until either:
  - No more rows are affected
  - Predefined time has passed
  - Predefined number of iterations has passed
- *oak-purge-master-logs*: safely purge master's binary logs after consulting with slaves' positions.
- *oak-show-limits*: show AUTO_INCREMENT "free space"
- More...

Thank you!

- I blog at http://openark.org
- Find open source projects on http://code.openark.org/forge/
- Do you wish to participate in *openark-kit* or other tools development?
  - Contact me at shlomi@[you-know-where].org
- Questions?