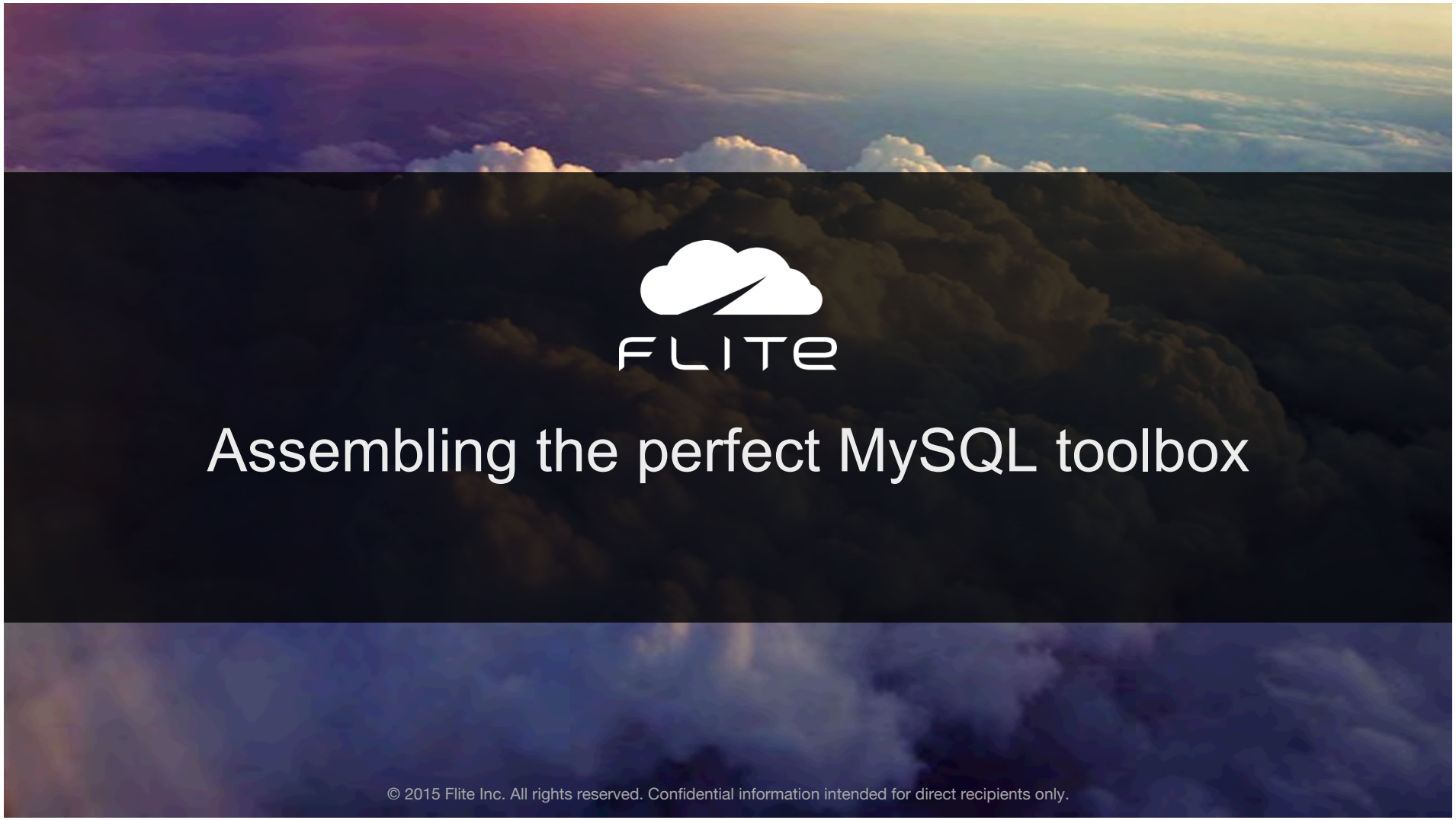


Assembling the perfect MySQL toolbox

ABOUT FLITE

Flite enables customers to build, traffic, and measure interactive, content-rich ads at scale

Founded: 2006

Investors: Sequoia Capital, HWVP, General Catalyst, Iris Capital

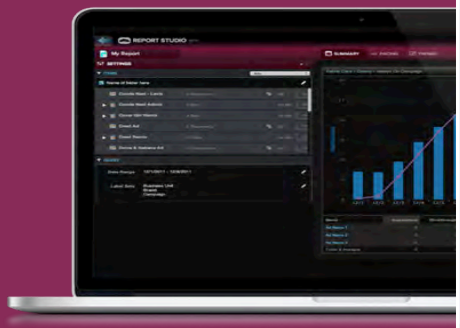
Headquarters: San Francisco, with offices in New York City, Chicago, and Boston

FLITE

FLITE PLATFORM

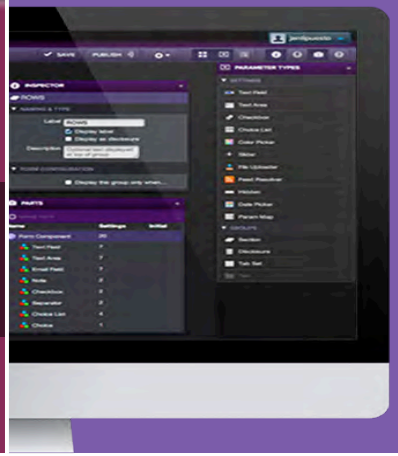
BUILD

TRAFFIC



MEASURE

EXTEND



SAMPLE CUSTOMERS

FLITE

MYSQL AT FLITE

PERCONA SERVER 5.6

- Running at AWS
- Almost all InnoDB
- Multi-region replication

OTHER DATA STORES

- **Hive** (for batch metrics)
- **Redis** (for realtime metrics)
- **Cassandra** (for data captured within ad units)

FLITE

ABOUT ME

Who am I?

1. Database Architect at **Flite** since 2007
2. **@iowalker** on twitter
3. Blog at **mechanics.flite.com**

FLITE

WHAT'S IN MY TOOLBOX?

FLITE

WHAT'S IN MY TOOLBOX?

PERCONA
TOOLKIT

FLITE

FLITE

PERCONA TOOLKIT

ORIGINALLY "MAATKIT" BY BARON SCHWARTZ

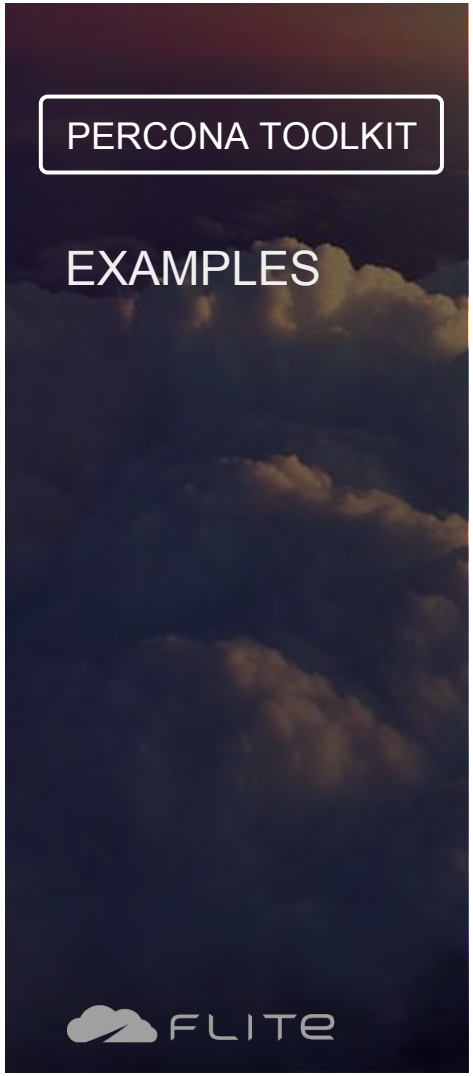
MAINTAINED BY PERCONA

FLITE

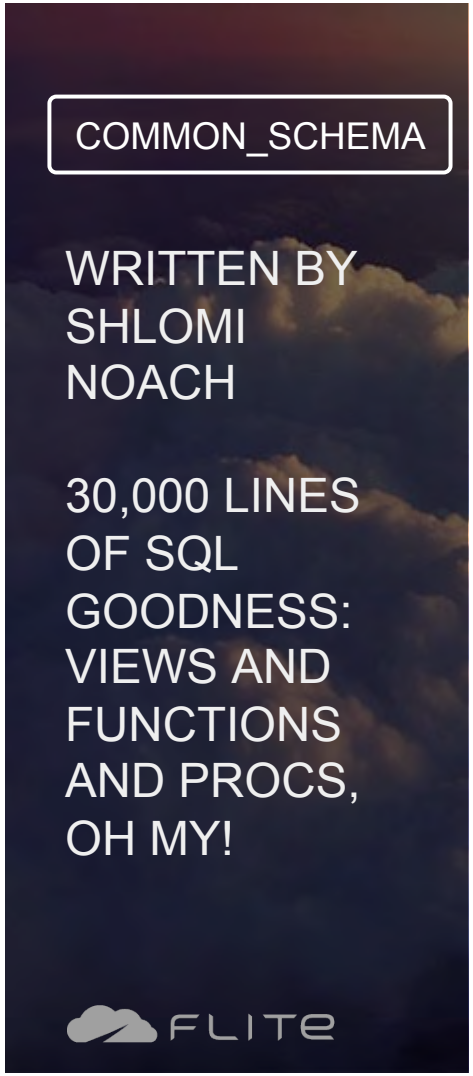
32 TOOLS (AS OF VERSION 2.2.14)

pt-align
pt-archiver ⚠
pt-config-diff
pt-deadlock-logger
pt-diskstats
pt-duplicate-key-checker
pt-fifo-split
pt-find
pt-fingerprint
pt-fk-error-logger
pt-heartbeat
pt-index-usage
pt-ioprofile ⚠
pt-kill
pt-mext
pt-mysql-summary
pt-online-schema-change
pt-pmp ⚠
pt-query-digest
pt-show-grants
pt-sift
pt-slave-delay
pt-slave-find
pt-slave-restart
pt-stalk
pt-summary
pt-table-checksum
pt-table-sync
pt-table-usage
pt-upgrade
pt-variable-advisor
pt-visual-explain

Free download at http://www.percona.com/software/percona-toolkit.




```
# capture 10 minutes of queries and profile them
pt-query-digest --processlist h=localhost --interval=0.01
--run-time 10m --output=slowlog > /tmp/slowlog.txt
pt-query-digest --report-format=profile /tmp/slowlog.txt

# kill and print all queries lasting over 2 minutes from
a certain user
pt-kill --match-user bad_user --match-command Query --
busy-time 120 --victims all --kill-query –print

# copy config files from two servers and compare them
scp db01.acme.com:/etc/my.cnf ~/my.cnf.db01
scp db02.acme.com:/etc/my.cnf ~/my.cnf.db02
pt-config-diff ~/my.cnf.db01 ~/my.cnf.db02
```



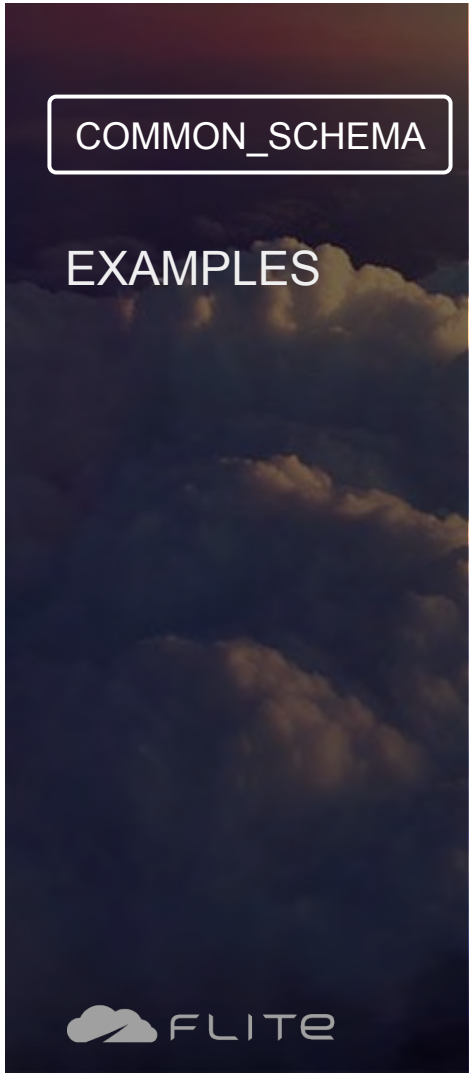
VIEWS LIBRARY
A set of powerful views, typically operating and presenting server metadata.

ROUTINES LIBRARY
A set of complementary routines to MySQL, including metadata handling.

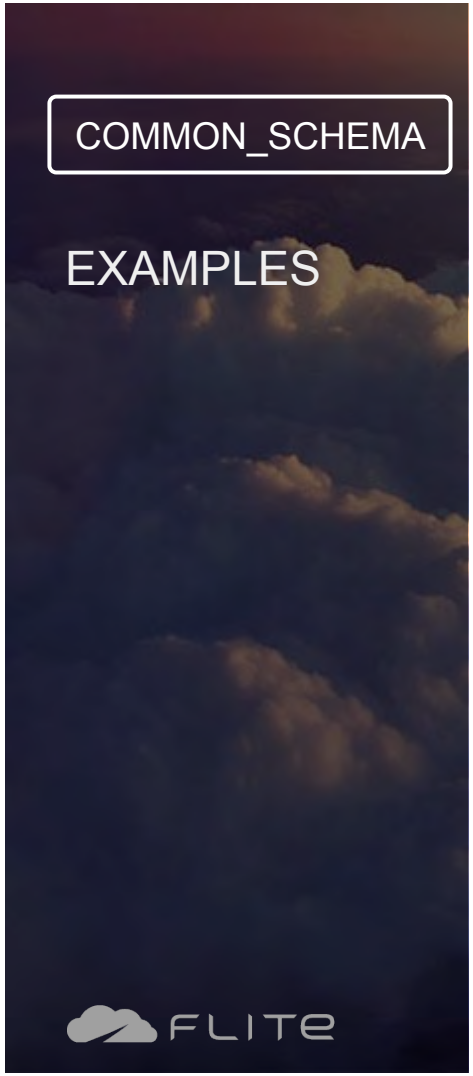
QUERYSCRIPT
A programming language aimed for SQL scripting, seamlessly combining scripting power such as flow control & variables with standard SQL statements or RDBMS-specific commands.

RDEBUG
A debugger and debugging API for MySQL stored routines

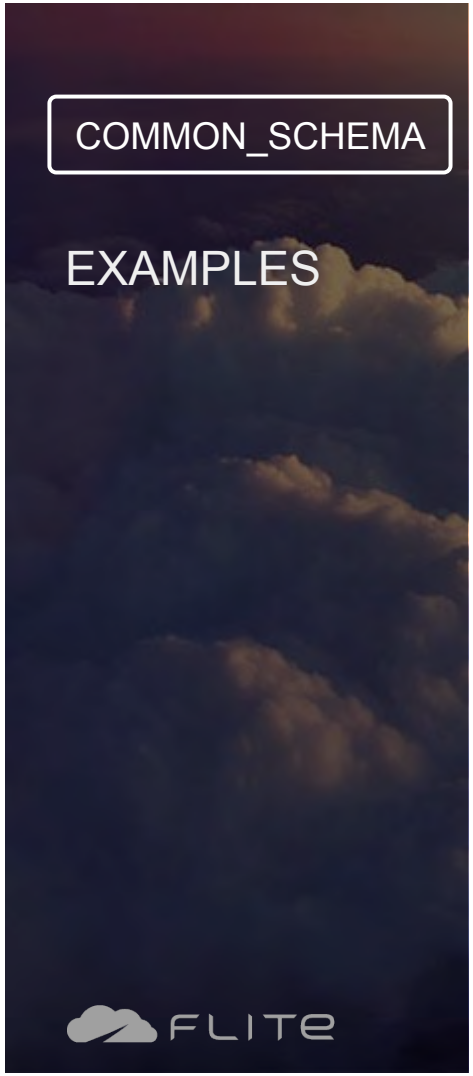
Learn more at:
https://code.google.com/p/common-schema/



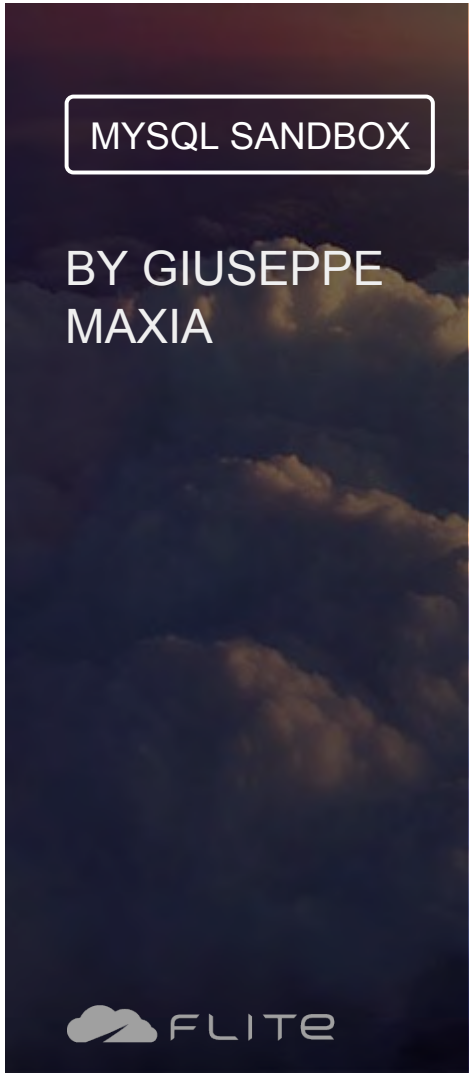

```
-- Update row format for all InnoDB tables in a given
schema
call common_schema.run("foreach($table, $schema, $engine:
table in sakila)
{
  if ($engine = 'InnoDB')
    ALTER TABLE :$schema.:$table ENGINE=InnoDB
ROW_FORMAT=Compact;
}");

-- insert a bunch of random numbers into a table
call common_schema.run("foreach($i: 1:10)
  insert into test.random_numbers select rand() from
common_schema.numbers;
");
```




```
DELIMITER $$

DROP FUNCTION IF EXISTS easter_day $$
CREATE FUNCTION easter_day(dt DATETIME) RETURNS DATE
DETERMINISTIC
NO SQL
SQL SECURITY INVOKER
COMMENT 'Returns date of easter day for given year'

BEGIN
    DECLARE p_year    SMALLINT DEFAULT YEAR(dt);
    DECLARE a    SMALLINT DEFAULT p_year % 19;
    DECLARE b    SMALLINT DEFAULT p_year DIV 100;
    DECLARE c    SMALLINT DEFAULT p_year % 100;
    DECLARE e    SMALLINT DEFAULT b % 4;
    DECLARE h    SMALLINT DEFAULT (19*a + b - (b DIV 4) - ((b - ((b + 8) DIV 25) +
1) DIV 3) + 15) % 30;
    DECLARE L    SMALLINT DEFAULT (32 + 2*e + 2*(c DIV 4) - h - (c % 4)) % 7;
    DECLARE v100 SMALLINT DEFAULT h + L - 7*((a + 11*h + 22*L) DIV 451) + 114;

    RETURN STR_TO_DATE(
                CONCAT(
                    p_year
                ,   '-'
                ,    v100 DIV 31
                ,   '-'
                ,   (v100 % 31) + 1
                )
            ,   '%Y-%c-%e'
            );
END $$

DELIMITER ;
```




```
mysql> -- when is easter next year?
mysql> select common_schema.easter_day('2016-01-01') as easter_2016;
+-------------+
| easter_2016 |
+-------------+
| 2016-03-27  |
+-------------+
1 row in set (0.05 sec)

mysql> -- what was the most common easter date in the 20th century?
mysql> select
right(common_schema.easter_day(concat(1900+n,'-01-01')),5) as
easter, count(*)
    -> from common_schema.numbers
    -> where n between 0 and 99
    -> group by easter
    -> order by count(*) desc
    -> limit 1;
+--------+----------+
| easter | count(*) |
+--------+----------+
| 04-12  |        5 |
+--------+----------+
1 row in set (0.01 sec)
```

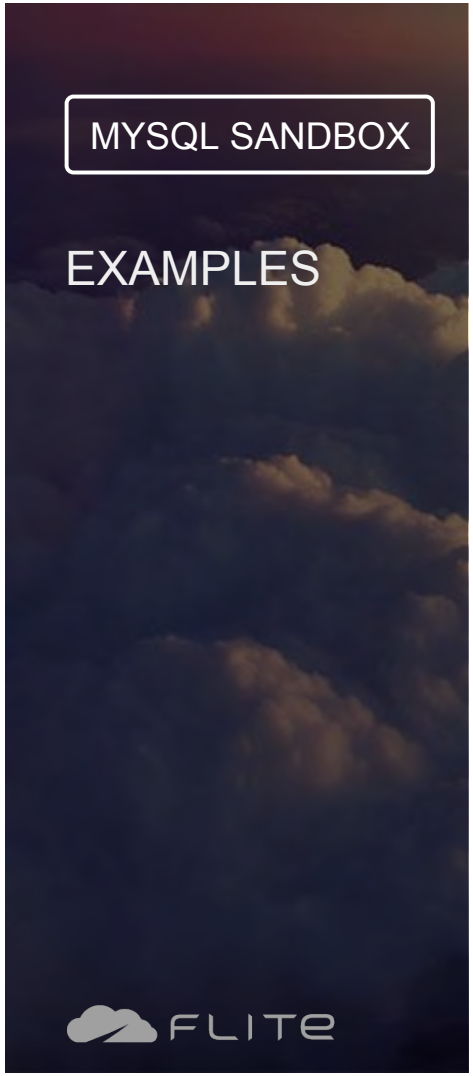


A tool for quickly and easily installing one or more MySQL Servers.

Get started in 2 lines:

```
$ sudo su -
# cpan MySQL::Sandbox
```

More info at:
http://mysqlsandbox.net/




```
# Create a MySQL 5.6 master with 2 slaves and connect to the master
$ make_replication_sandbox ~/Downloads/mysql-5.6.22-osx10.8-x86_64.tar.gz
# executing "clear" on /Users/ikewalker/sandboxes/rsandbox_mysql-5_6_22
executing "clear" on slave 1
executing "clear" on slave 2
executing "clear" on master
installing and starting master
installing slave 1
installing slave 2
starting slave 1
. sandbox server started
starting slave 2
.. sandbox server started
initializing slave 1
initializing slave 2
replication directory installed in $HOME/sandboxes/rsandbox_mysql-5_6_22

$ cd $HOME/sandboxes/rsandbox_mysql-5_6_22
$ ./m
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.6.22-log MySQL Community Server (GPL)
…

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

master [localhost] {msandbox} ((none)) >
```

MORE TOOLS

1. MySQL Utilities
2. swanhart-tools
3. Innotop
4. MySQL sys schema (fka ps_helper)
5. Pmysql
6. Securich

FLITE

ASSEMBLE YOUR TOOLBOX LIKE A B.O.S.S.

- **B**uy the right tool
- **O**wn the tool
- **S**ell the tool to others
- **S**uccess!

FLITE

BUY THE RIGHT TOOL

- **Buy it? I thought these tools were free???**

FLITE

BUY THE RIGHT TOOL

Do some research

- **Invest the time**
- **Make the commitment**

FLITE

OWN THE TOOL

- **Own it? How can I own something that's free?**

OWN THE TOOL

Learn how the tool works

1. **Try it** in a test environment
2. Look at the **code**
3. **Submit feedback** to the tool's creator/maintainer

FLITE

SELL THE TOOL

Sell it? I’m not a salesperson!

SELL THE TOOL

- **Show** your co-workers how you are using the tool
- **Blog** about the tool and what you did with it
- **Share your knowledge** on a mailing list or Q&A site like Stack Overflow

FLITE

SUCCESS!

Your problem is solved and you've expanded your toolbox, now move on to the next problem…

FLITE

YOU’RE THE B.O.S.S.!

FLITE

EXAMPLE #1

THE GREAT WIDGETBOX PURGE

1. Flite evolved from a product called **Widgetbox**.
2. **80% of the data** in MySQL was legacy Widgetbox data
3. How can I delete that data **safely, easily, and efficiently**?

FLITE

BUY IT

I considered using pt-archiver, but after reading a blog post by Baron Schwartz I decided to use common_schema instead

http://www.xaprb.com/blog/2013/01/28/deleting-millions-of-rows-in-small-chunks-with-common_schema/

FLITE

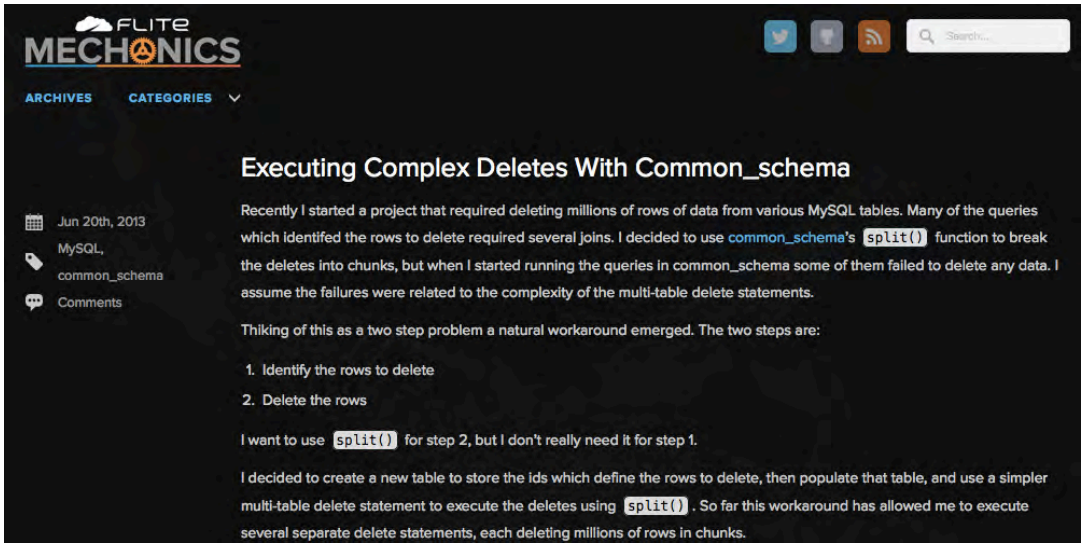
OWN IT

- I ran tests in our stage environments to understand the way common_schema chunks deletes
- I opened a bug report based on that experience, which Shlomi promptly fixed ☺

FLITE

SELL IT

I blogged about doing data purges with common_schema



FLITE MECHANICS

ARCHIVES CATEGORIES

Jun 20th, 2013
MySQL, common_schema
Comments

Executing Complex Deletes With Common_schema

Recently I started a project that required deleting millions of rows of data from various MySQL tables. Many of the queries which identifed the rows to delete required several joins. I decided to use common_schema's `split()` function to break the deletes into chunks, but when I started running the queries in common_schema some of them failed to delete any data. I assume the failures were related to the complexity of the multi-table delete statements.

Thiking of this as a two step problem a natural workaround emerged. The two steps are:

1. Identify the rows to delete
2. Delete the rows

I want to use `split()` for step 2, but I don't really need it for step 1.

I decided to create a new table to store the ids which define the rows to delete, then populate that table, and use a simpler multi-table delete statement to execute the deletes using `split()`. So far this workaround has allowed me to execute several separate delete statements, each deleting millions of rows in chunks.

http://mechanics.flite.com/blog/2013/06/20/executing-complex-deletes-with-common-schema/

FLITE

SUCCESS!

- After optimizing all of the tables, the database .ibd files shrank by 80%
- (ibdata1 is still big, time to find another tool and start the B.O.S.S. process again!) This might help: http://www.mysqlplus.net/2014/09/11/shrink-ibdata-file-trite/

FLITE

EXAMPLE #2

Check data consistency after the purge

1. After purging 80% of the database, how do I know that the data is consistent across all replicas?

FLITE

BUY IT

1. Sounds like a job for pt-table-checksum
2. (and pt-table-sync)
3. Time to create a DSNs table…

FLITE

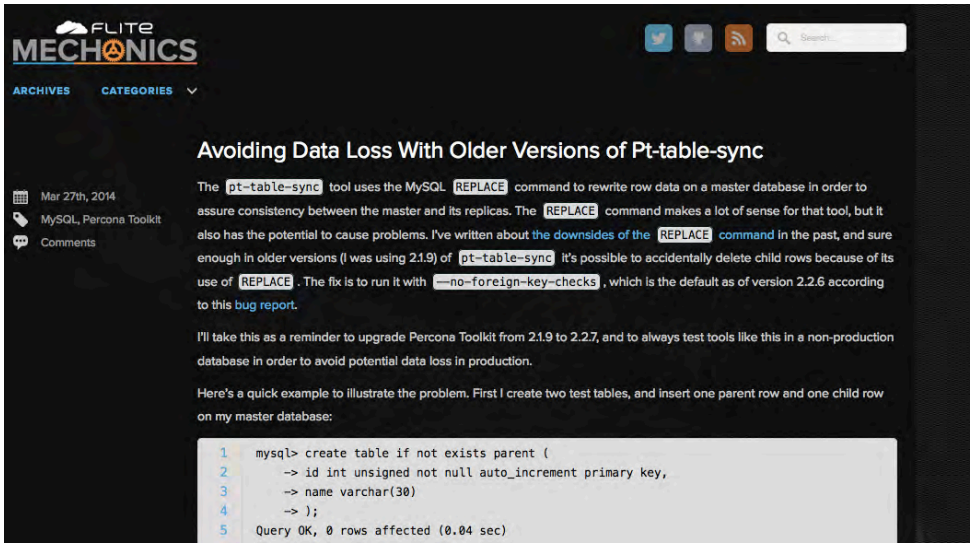
OWN IT

1. Lots of testing in stage.
2. More testing
3. Create a DSN table that knows about all replicas
4. Learn that schema differences can cause the tool to fail if it doesn't know about all replicas
5. Explicitly set the charset for heterogenous schema because of bug 1400472

FLITE

SELL IT

I blogged about the importance of --no-foreign-key-checks in pt-table-sync



FLITE MECHANICS

ARCHIVES CATEGORIES

Mar 27th, 2014
MySQL, Percona Toolkit
Comments

Avoiding Data Loss With Older Versions of Pt-table-sync

The `pt-table-sync` tool uses the MySQL `REPLACE` command to rewrite row data on a master database in order to assure consistency between the master and its replicas. The `REPLACE` command makes a lot of sense for that tool, but it also has the potential to cause problems. I've written about the downsides of the `REPLACE` command in the past, and sure enough in older versions (I was using 2.1.9) of `pt-table-sync` it's possible to accidentally delete child rows because of its use of `REPLACE`. The fix is to run it with `--no-foreign-key-checks`, which is the default as of version 2.2.6 according to this bug report.

I'll take this as a reminder to upgrade Percona Toolkit from 2.1.9 to 2.2.7, and to always test tools like this in a non-production database in order to avoid potential data loss in production.

Here's a quick example to illustrate the problem. First I create two test tables, and insert one parent row and one child row on my master database:

```
1  mysql> create table if not exists parent (
2      -> id int unsigned not null auto_increment primary key,
3      -> name varchar(30)
4      -> );
5  Query OK, 0 rows affected (0.04 sec)
```

http://mechanics.flite.com/blog/2014/03/27/avoiding-data-loss-with-older-versions-of-pt-table-sync/

FLITE

SUCCESS!

Now I can sync all of my replicas to the masters and make sure they stay consistent

FLITE

EXAMPLE #3

Patching pt-online-schema-change to work on tables with “before” triggers

1. pt-online-schema-change refuses to run on tables with any triggers, but it only creates “after” triggers.
2. I made a 3 line change to allow it to run on tables with “before” triggers
3. https://bugs.launchpad.net/percona-toolkit/+bug/1270328

FLITE

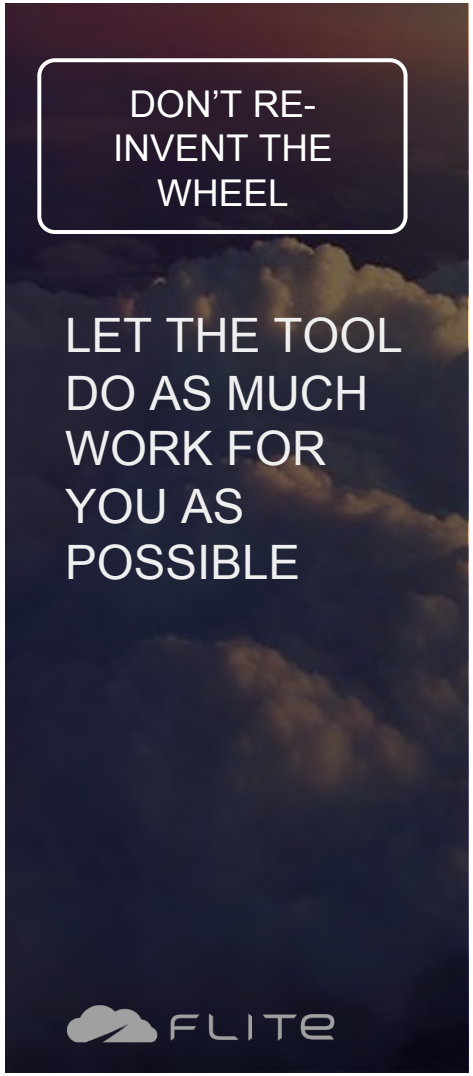
B.O.S.S. REMINDERS

REMINDERS

1. Don’t re-invent the wheel
2. Only use tools you understand
3. Test it before you use it in production

FLITE

DON’T RE-INVENT THE WHEEL



Examples:

pt-heartbeat
Use the built-in checks to monitor replication delay.

For Nagios alerts, combine pt-heartbeat with **pmp-check-mysql-replication-delay** (and pmp-check-mysql-replication-running).

pt-table-checksum
Use **pt-table-sync** with pt-table-checksum to resolve data differences.

ONLY USE TOOLS YOU UNDERSTAND

FLITE

ONLY USE TOOLS YOU UNDERSTAND

GET TO KNOW THE CODE
Tool code is typically **much more accessible** than server code.

NEVER RUN A TOOL WITH ONLY DEFAULT OPTIONS
Explicitly declare the options to force yourself to see them and think about them.

SWEAT THE SMALL STUFF
Small changes like running pt-kill with –kill versus –kill-query can make a difference

Examples:
- if pt-table-checksum doesn't know about all replicas it can break replication
- pt-table-sync can delete child data because of cascade delete

TEST TOOLS OUTSIDE OF PRODUCTION

Don’t test in prod (duh)

1. Test it in **dev, qa, stage**, etc
2. **MySQL Sandbox** is super useful for this
3. If you are in the cloud, spin up a new **disposable** DB instance with your production data set for testing

FLITE

MySQL 5.6

DO WE STILL NEED pt-slave-delay?

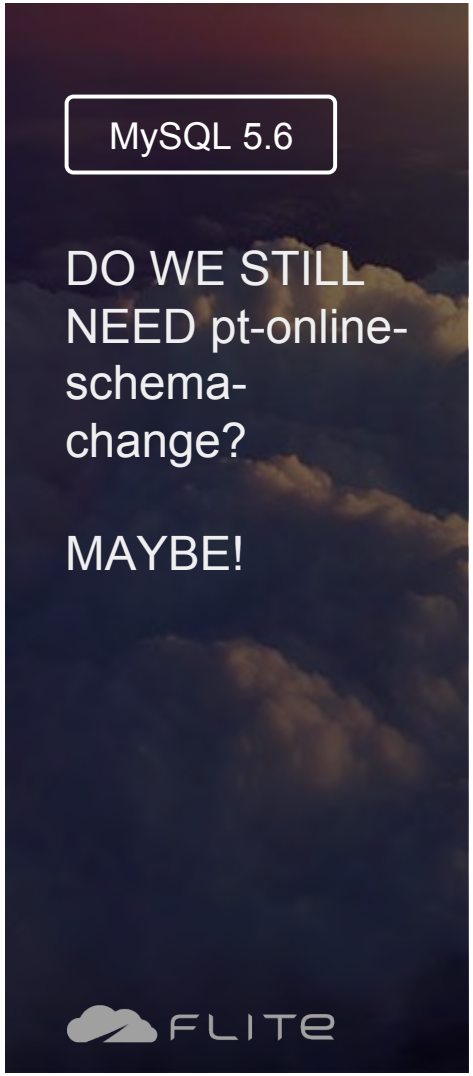
NO!

FLITE

```
CHANGE MASTER TO MASTER_DELAY = N;
```

https://dev.mysql.com/doc/refman/5.6/en/replication-delayed.html

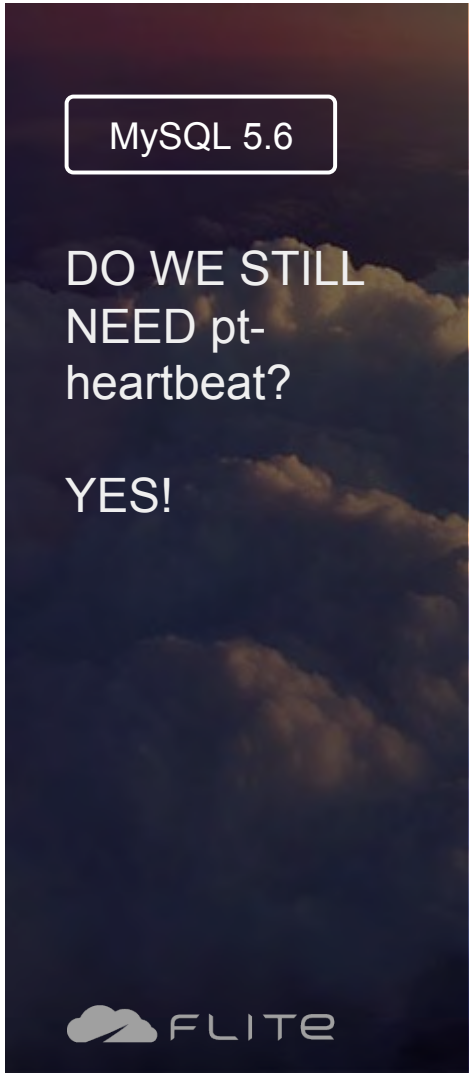
http://mechanics.flite.com/blog/2014/07/28/replacing-pt-slave-delay-with-master-delay-in-mysql-5-dot-6/



In MySQL 5.6 lots of DDL operations are no longer blocking, but it's complicated…

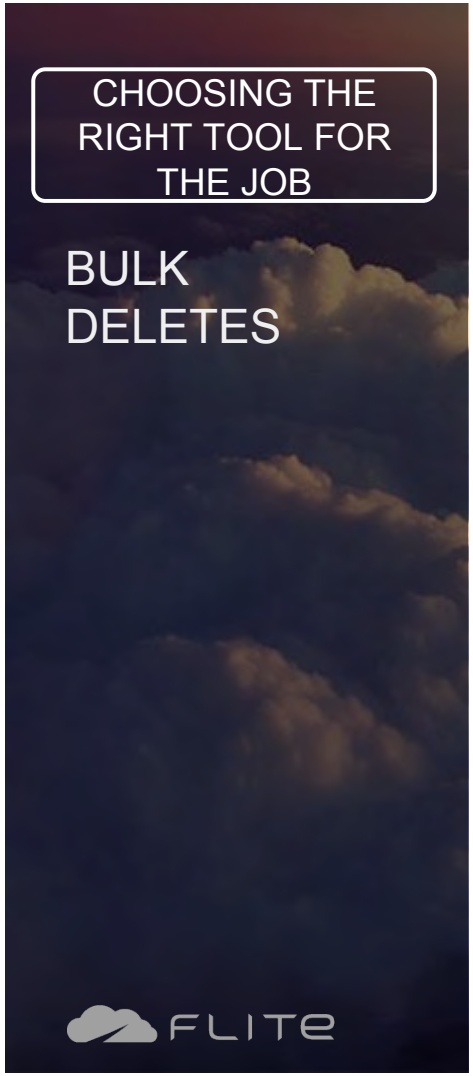
Lots of good details in this article:

http://www.tocker.ca/2013/11/05/a-closer-look-at-online-ddl-in-mysql-5-6.html

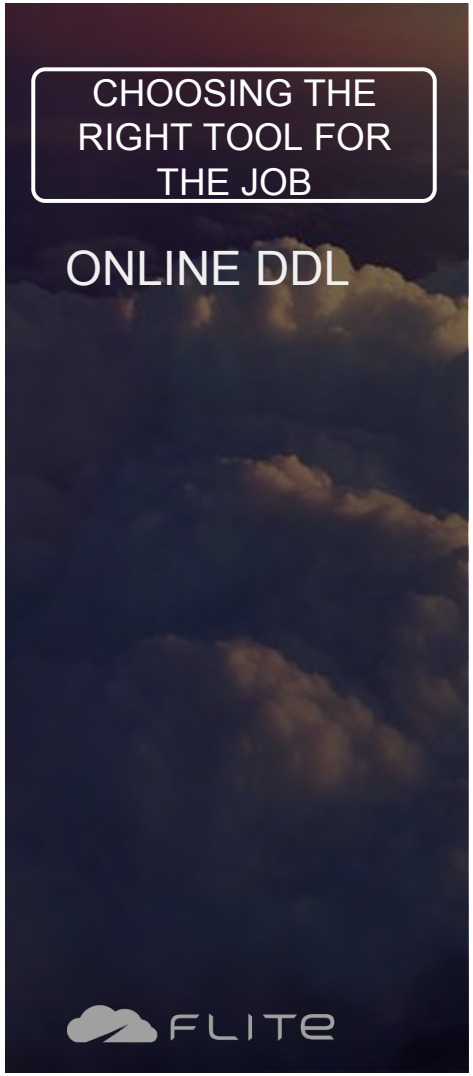


- It's still more accurate than Seconds_Behind_Master

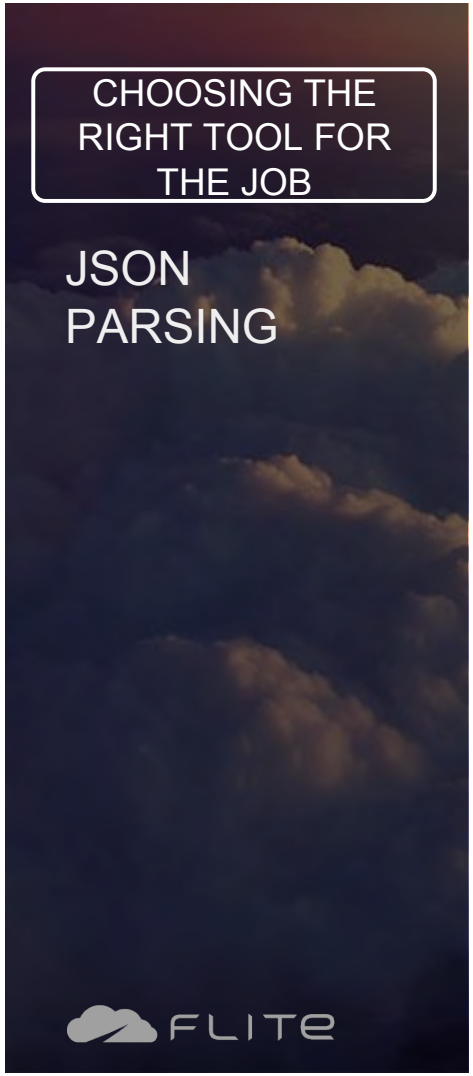
- Not the same thing as MASTER_HEARTBEAT_PERIOD



pt-archiver vs common_schema



pt-online-schema-change
vs
oak-online-alter-table
vs
ALTER TABLE in MySQL 5.6
vs
ALTER TABLE in MySQL 5.7



common_schema
vs
MySQL JSON UDFs
vs
rapidjson UDFs
vs
MySQL 5.7 built-in JSON functions

CONCLUSION

1. Be a **B.O.S.S.**
2. Always look for a **better way**
3. **Share** your experiences with the community

FLITE