



Your network is more powerful
than you think

MySQL migration from latin1 to UTF-8

Meetup Viadeo / LeMUG.fr, Paris 16-11-2011



Olivier DASINI - dasini.net/blog/

+ In this talk

- Why migrate in UTF-8
- Charset and collation ?
- Obstacles faced
- Solutions found, approved and tested
- Rolling upgrade
- Switchover
- Reminder... (abstract for exhausted people)



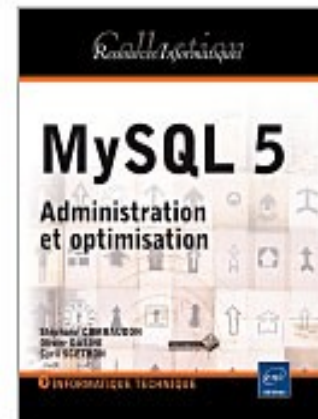
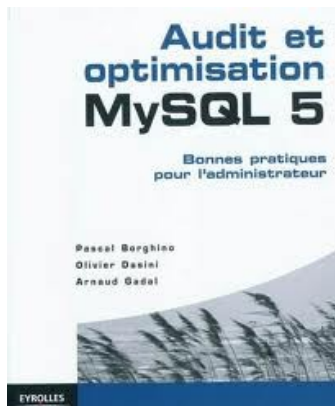
+ Me, myself & I

➤ Olivier DASINI

- MySQL Expert at Viadeo
 - <http://fr.viadeo.com/fr/profile/olivier.dasini>
- My technology watch blog on MySQL
 - <http://dasini.net/blog/>
- Co-founder of the french society: MySQL User Group Francophone (LeMug.fr)
 - <http://lemug.fr>

➤ Book author (in french)

- Audit et optimisation – MySQL 5, Bonnes pratiques pour l’administrateur
 - Eyrolles, ISBN-13: 978-2212126341
- MySQL 5 – Administration et optimisation
 - ENI, ISBN-13: 978-2-7460-5516-2



+ Migration UTF-8 mission

➤ Main mission

- Convert the data to UTF-8
 - Viadeo used in over 200 countries
 - Including China, India, Russia, Middle East, ...

➤ Secondaries missions

- Convert the tables to InnoDB
 - Performances, Hot Backup, Operability,...
- Server tuning
 - InnoDB tuning differ from MyISAM tuning



+ Charset & collation 1/4

あ A	か KA	さ SA	た TA	な NA
い I	き KI	し SI	ち TI	に NI
う U	く KU	す SU	つ TU	ぬ NU
え E	け KE	せ SE	て TE	ね NE
お O	こ KO	そ SO	と TO	の NO
だ DA	ぢ DJI	づ ZUE	で DE	ど DO

Charset

- Can be defined as the encoding of an alphabet
- 39 different charset in MySQL 5.5
- **Latin1** (ISO/CEI 8859-1) is the default charset in MySQL
 - « Almost » optimal for Western Europe
 - 1 character = 1 byte
- **utf8** is that we want (must) use
 - « universal » charset
 - 1 character = 1, 2 ou 3 byte(s)

```
SHOW CHARACTER SET WHERE Charset='latin1';
```

Charset	Description	Default collation	Maxlen
latin1	cp1252 West European	latin1_swedish_ci	1

```
SHOW CHARACTER SET WHERE Charset='utf8';
```

Charset	Description	Default collation	Maxlen
utf8	UTF-8 Unicode	utf8_general_ci	3

+ Charset & collation 2/4

➤ **Latin1 does not recognize all the characters**

```
CREATE TABLE t_latin1 (
  nom varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci
```

```
CREATE TABLE t_utf8 (
  nom varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_swedish_ci
```

```
SELECT * FROM t_latin1;
```

```
+-----+
| nom   |
+-----+
| ??   |
+-----+
```

 **Latin1 can't encode Mandarin**

À	Á	Â	Ã	Ä	Å	Æ	Ç	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
a	á	â	ã	ä	å	æ	ç	ð	ñ	ó	ô	õ	ö	ø	ù	ú	û	ü	ý	þ	ÿ	
í	j	k	l	ly	m	n	ny	o	ó	ô	õ	ö	ø	p								
Ñ	Λ	I	Ƴ	X	M	M	Q	N	M	Æ	Y	∫										
r	s	sz	t	ty	u	ú	ü	ú	v	z	zs	ak										

```
SELECT * FROM t_utf8;
```

```
+-----+
| nom   |
+-----+
| 谢谢 |
+-----+
```

+ Charset & collation 3/4

➤ Collation

- Can be defined as the set of rules that compare and order the symbols of an alphabet.
- Used mainly in the sorts
- Related to a charset
 - LATIN1 default collation is **latin1_swedish_ci**
 - UTF-8 default collation is **utf8_general_ci**
- Beware of the differences in behavior and performances...



+ Charset & collation 4/4

➤ Collation is used mainly in the sorts

```
SELECT * FROM table ORDER BY col COLLATE latin1_xxxxx_ci;
```

```
SELECT * FROM c1;
```

c
û
u
ü
ù

```
ORDER BY c COLLATE latin1_swedish_ci;
```

c
û
u
ù
ü

```
ORDER BY c COLLATE latin1_german1_ci;
```

c
û
u
ü
ù

```
ORDER BY c COLLATE latin1_german2_ci;
```

c
û
u
ù
ü

TO BE CONTINUED...

+ Method

➤ Convert the tables

- Charset : UTF-8
- Collation : utf8_swedish_ci
- Storage : InnoDB

➤ Very simple with MySQL

- A single command : ALTER TABLE
- Fingers in the nose !

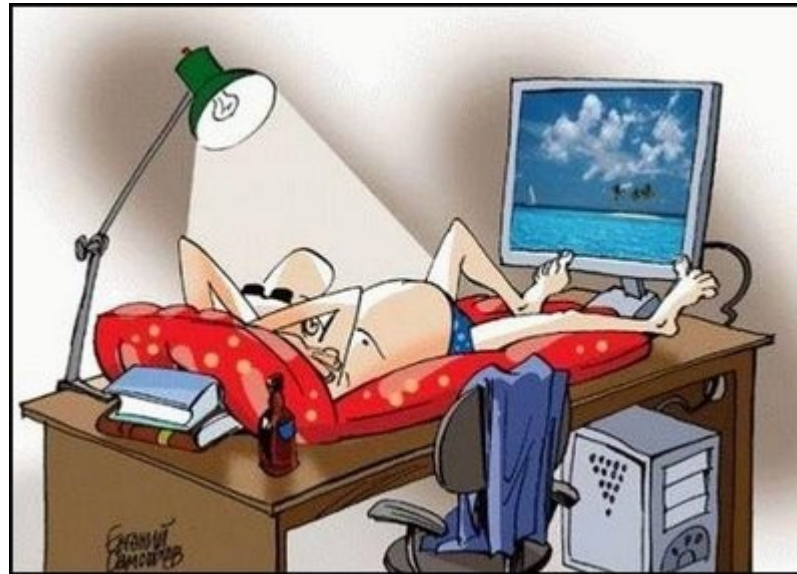
```
ALTER TABLE ma_table ENGINE = INNODB, ALTER TABLE ma_table CHARSET  
= utf8 COLLATE utf8_swedish_ci
```

Does it work ?



+ The end ?

Thank you for your attention !



+ Not so simple...

➤ Viadeo constraints

- Workload: a huge amount of data is managed (Nov. 2011)
 - 1 000 000 new members / month
 - 250 000 connections / day
 - 165 000 active discussions in forum
 - 80 000 forum member / month
 - 18 000 articles shared / day
 - 1 250 events organized / week
- Minimize the downtime
 - No connection = no income
- Nasty surprise
 - Dirty data
 - Legacy...

➤ MySQL constraints

- Maximum size of the index
- Characteristics related to the CHARSET & COLLATION
- => had suddenly become issues when migrating ...

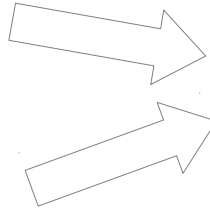
+ Mission: Impossible ?



+ Viadeo constraints

➤ MySQL @ Viadeo (prod OLTP) : huge volume of data

- About thirty instances distributed on 5 « clusters » ie Master / Slaves replication
- 2 TB of data (for MySQL)
- About 1000 tables
 - 70% MyISAM
- 5 billions rows



➤ 20% of the effort

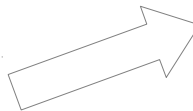
- Required good MySQL skills
- Scripting know-how

➤ Minimize the downtime with MySQL replication

- => *Rolling upgrade*
 - Slaves are migrated first
 - Allowing testing and validate the process
- => *Switchover*
 - A slave promoted to master

➤ 80% of the effort

- Required good business skills
- Large consumer of time and energy



➤ Data legacy

- for historical reasons, some data was not « clean »
 - Thank you to the elders.....



- The application migration was carried out several months before
- => mix of latin1 et utf8 data in latin1 tables
- => Tedious hand cleaning
- => Performed with a good knowledge of code
 - Thank you to the elders ! 😊



+ 80% of the effort

- **Required good business skills**
- **Large consumer of time and energy**
 - 3 weeks of labor
 - The dark side of the force...



+ Data legacy - Taille max des index

- Limited to 767 bytes for InnoDB (1000 bytes for MyISAM)
 - Specified key was too long; max key length is 767 bytes
 - Warning 1071 : for a not unique index
 - MySQL index the 255 first characters
 - KEY `idx_url` (`url`(255))
 - ERROR 1071 (42000) : for an unique index
 - Many ways to handle it.
 - Highly dependent on the business logic.
 - => Processed on a case by case basis.

```
ALTER TABLE _table CONVERT TO CHARACTER SET utf8;
ERROR 1071 (42000): Specified key was too long; max key length is 767 bytes
```

```
CREATE TABLE _table (
  info varchar(256) NOT NULL,      256 x 3 = 768 > 767 ... the count is not good !
  PRIMARY KEY (info)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

```
ALTER TABLE _table2 CONVERT TO CHARACTER SET utf8;
Query OK, 0 rows affected, 2 warnings (0.33 sec)
Records: 0 Duplicates: 0 Warnings: 2
```

```
CREATE TABLE _table2 (
  info varchar(256) NOT NULL,
  KEY info (info(255)) ← Add by MySQL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

+ Data legacy - Duplicate entry 1/3

Process

- Run tests on a sample data
- Understand the problem(s)
 - **ERROR 1062 (23000): Duplicate entry**
 - => Different charset = different behavior of the server
- Identify rows issues
 - The problematic characters :
 - Do not appear to cause problems or are sometimes invisible
 - With SQL queries : easy but sometimes long, often very long
- Deal with the problem
 - Understand the data
 - Understand MySQL
 - Difficult to automate => TEDIOUS

+ Data legacy - Duplicate entry 2/3

➤ Weird characters in the data

➤ A0 / A020 / 20A0 / C2A0 / ... at the end of some rows

Clean hand operation !



```
SELECT ID, hex(url) FROM _table WHERE LEFT(reverse(Url),2) LIKE  
CONCAT(UNHEX('A0'),'%') ;
```

```
for col in postID; do  
  for carac in A0 A020 20A0; do  
    mysql -ugrantless -uP4S5 -B -N -e"SELECT ID FROM _table  
WHERE LEFT(reverse($col),2) LIKE CONCAT(UNHEX('$carac'),'%');"  
  done;  
done;
```

+ Data legacy - Duplicate entry 2/3

- **ERROR 1062 (23000): Duplicate entry 'pykachu' for key 'surnom'**
 - While there are (apparently) no duplicates...

```
SELECT surnom... LIKE 'p_kachu';
```

surnom
pykachu
pÿkachu

← Unique index



?

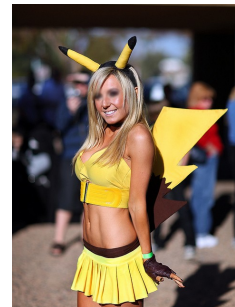
=



```
SELECT surnom... LIKE 'p_kachu';
```

surnom
pykachu
pykachu

← Unique index



?

=



➤ Similar but different characters...

- Depends on the collation

ÿ = ÿ
 ² = 2
 @ = a
 ß = ss

+ Charset & collation (again) 1/4

➤ 2 different letters can be similar...

```
SELECT 'u' = 'ü' COLLATE utf8_general_ci;
+-----+
| 'u' = 'ü' COLLATE utf8_general_ci |
+-----+
|                                1 |
+-----+
```

```
SELECT 'e' = 'ë' COLLATE utf8_general_ci;
+-----+
| 'e' = 'ë' COLLATE utf8_general_ci |
+-----+
|                                1 |
+-----+
```

```
SELECT 'u' = 'ü' COLLATE utf8_swedish_ci;
+-----+
| 'u' = 'ü' COLLATE utf8_swedish_ci |
+-----+
|                                0 |
+-----+
```

```
SELECT 'e' = 'ë' COLLATE utf8_swedish_ci;
+-----+
| 'e' = 'ë' COLLATE utf8_swedish_ci |
+-----+
|                                1 |
+-----+
```

```
SELECT 'u' = 'ü' COLLATE utf8_bin;
+-----+
| 'u' = 'ü' COLLATE utf8_bin |
+-----+
|                                0 |
+-----+
```

```
SELECT 'e' = 'ë' COLLATE utf8_bin;
+-----+
| 'e' = 'ë' COLLATE utf8_bin |
+-----+
|                                0 |
+-----+
```

+ Charset & collation (again) 2/4

➤ Behavior may differ between *latin1_swedish_ci* & *utf8_swedish_ci*

```
SELECT 'u' = 'ü' COLLATE latin1_swedish_ci;
+-----+
| 'u' = 'ü' COLLATE utf8_general_ci |
+-----+
|                                0 |
+-----+
```

```
SELECT 'u' = 'ü' COLLATE utf8_swedish_ci;
+-----+
| 'u' = 'ü' COLLATE utf8_swedish_ci |
+-----+
|                                0 |
+-----+
```

Identical behavior

- u is different from ü

```
SELECT 'e' = 'ë' COLLATE latin1_swedish_ci;
+-----+
| 'e' = 'ë' COLLATE utf8_general_ci |
+-----+
|                                0 |
+-----+
```

```
SELECT 'e' = 'ë' COLLATE utf8_swedish_ci;
+-----+
| 'e' = 'ë' COLLATE utf8_swedish_ci |
+-----+
|                                1 |
+-----+
```

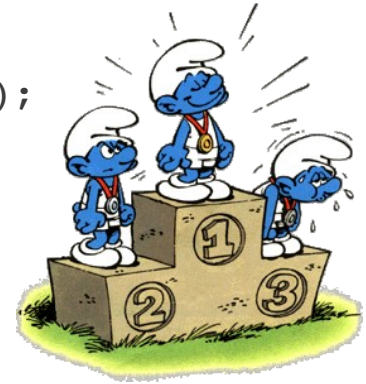
Different behavior

- e is different from ë in *latin1_swedish_ci*
- e is identical to ë in *utf8_swedish_ci*

+ Charset & collation (again) 3/4

➤ Collation : pay attention to differences in performance

```
SELECT BENCHMARK(1000000000, (select 'u'='ü' collate utf8_bin));
+-----+
| BENCHMARK(1000000000, (select 'u'='ü' collate utf8_bin)) |
+-----+
|                                                                 0 |
+-----+
1 row in set (20.62 sec)
```



```
SELECT BENCHMARK(1000000000, (select 'u'='ü' collate utf8_swedish_ci));
+-----+
| BENCHMARK(1000000000, (select 'u'='ü' collate utf8_swedish_ci)) |
+-----+
|                                                                 0 |
+-----+
1 row in set (57.53 sec)
```

```
SELECT BENCHMARK(1000000000, (select 'u'='ü' collate utf8_general_ci));
+-----+
| BENCHMARK(1000000000, (select 'u'='ü' collate utf8_general_ci)) |
+-----+
|                                                                 0 |
+-----+
1 row in set (27.71 sec)
```

+ Charset & collation (again) 4/4 - Illegal mix

➤ Collation : pay attention to collation mixes

```
CREATE TABLE City (  
...  
) DEFAULT CHARSET=utf8 COLLATE=utf8_swedish_ci
```

```
CREATE TABLE Country (  
...  
) DEFAULT CHARSET=utf8 ← collate=utf8_general_ci
```

```
SELECT City.name FROM City JOIN Country USING(name) ;
```

```
ERROR 1267 (HY000): Illegal mix of collations (utf8_swedish_ci,IMPLICIT)  
and (utf8_general_ci,IMPLICIT) for operation '='
```

```
SELECT City.name FROM City Ci JOIN Country Co ON Ci.name=Co.name COLLATE  
utf8_general_ci ;
```

Bypassing the problem

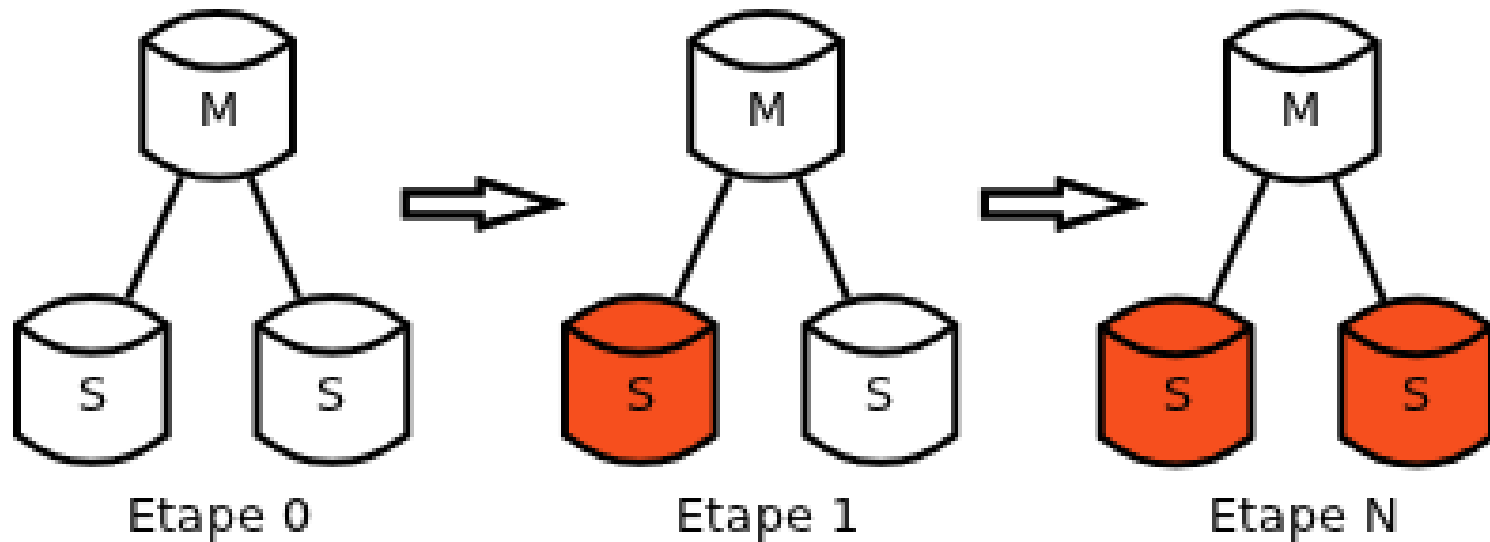
+ 20% of the effort

- Required good MySQL skills
- Scripting know-how
 - 3 days of labor
 - Light side of the force



+ Minimize the downtime - Rolling upgrade

➤ Updated tested and validated on slaves



White DB : latin-1
Orange DB : UTF-8

+ Rolling upgrade 1/4

➤ Optimize the duration of the migration

- Minimize disk I/O
- Maximize buffers utilization

➤ Extract, Transform and Load (data)

- mysqldump
- ALTER TABLE
- mysqlimport
- REPAIR TABLE (MyISAM)

➤ Tuning server

- Setting buffers o their proper value
- Various optimization

➤ MySQL replication

- Stop it before the migration process
- Start it after the migration process

Powered by bash !

+ Rolling upgrade 2/4

➤ Optimize the duration of the migration

- Minimize disk I/O
- Maximize buffers utilization

Disable logs

```
SET SESSION SQL_LOG_BIN=0;  
SET GLOBAL slow_query_log=0;  
SET GLOBAL general_log=0;
```

InnoDB tuning

Reduction disks I/O

```
SET GLOBAL innodb_flush_log_at_trx_commit = 0;  
SET GLOBAL innodb_support_xa = 0;  
SET GLOBAL unique_checks=0;  
SET GLOBAL foreign_key_checks=0;
```

MyISAM tuning

```
SET GLOBAL myisam_sort_buffer_size = N;  
  
SET GLOBAL bulk_insert_buffer_size = N;
```

Buffer for sorting MyISAM indexes:
REPAIR TABLE / CREATE INDEX / ALTER TABLE

Optimization : LOAD DATA INFILE

Server tuning

```
SET GLOBAL read_buffer_size = N;    Used for « full table scan »
```

Default DB charset & collation

```
ALTER DATABASE DEFAULT CHARACTER SET utf8 COLLATE utf8_swedish_ci;
```

+ Rolling upgrade 3/4

Extract, Transform and Load (data)

- mysqldump
- ALTER TABLE
- mysqlimport
- REPAIR TABLE (MyISAM)

Save the structure and the data in UTF8:

```
mysqldump --default-character-set=utf8 --hex-blob DB table1 table2 -T /path/to/bck/  
1m22.406s
```

Convert the tables in InnoDB, utf8, utf8_swedish_ci:

```
ALTER TABLE _table ENGINE=InnoDB, CONVERT TO CHARACTER SET utf8 COLLATE  
utf8_swedish_ci;
```

```
Query OK, 12193106 rows affected (31 min 14.77 sec)
```

```
Records: 12193106 Duplicates: 0 Warnings: 0
```

← To do without the data !!!

Load the data in UTF8:

```
time mysql --default-character-set=utf8 DB < data.sql  
real 26m4.613s
```

VS

```
time mysqlimport --default-character-set=utf8 DB _table.txt  
DB._table: Records: 12193106 Deleted: 0 Skipped: 0 Warnings: 0
```

```
real 13m40.607s
```

← mysqlimport (load data infile) is the fastest

+ Rolling upgrade 4/4

↳ Tuning server

- ↳ Setting buffers o their proper value
- ↳ Various optimization

Refactoring my.cnf + Tuning Server

```
default_storage_engine=InnoDB
```

```
# Charset & Collation
```

```
character_set_server=utf8
```

```
collation_server=utf8_swedish_ci
```

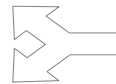
```
# Replication
```

```
Skip-slave-start
```

```
# Change hashing algorithm (REHASH passwords)
```

```
old_passwords=0
```

```
...
```



Limit bad surprises

The old passwords hashing algorithm is not secure (< 4.1)

```
$ time ./poc XXxxXxxXXXxXxXXX
```

```
mysql crack POC (c) 2006 Philippe Vigier & www.sqlhack.com
```

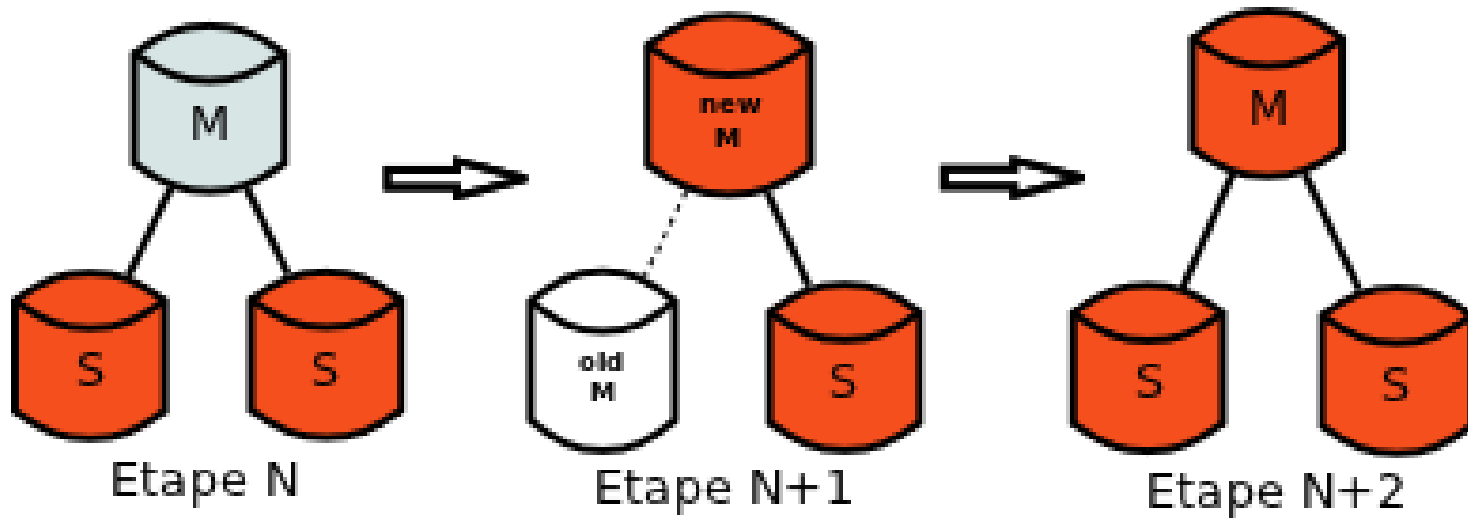
```
password for footprint XXxxXxxXXXxXxXXX = '_____'
```

```
real 38m47.400s
```

+ Minimize the downtime - Switchover

➤ Switchover for minimize the downtime

- A slave turn master
- The master become a slave



+ Switchover - Pseudo code 1/3

```
/* the master should be in READ ONLY mode*/  
$mA_cmd = "SET GLOBAL read_only='ON';";
```

Does not work with the
SUPER privilege

```
$mA_cmd = "FLUSH NO_WRITE_TO_BINLOG TABLES WITH READ LOCK;";
```

```
/*Retrieved old master binlog info*/
```

```
$mA_cmd = "SHOW MASTER STATUS;";
```

Allows the slave to stop at the exact position

```
$old_master_file = $row["File"];
```

```
$old_master_pos = $row["Position];
```

```
/* Kill the pending old master's connexions */
```

```
SELECT sleep(2);
```

```
SELECT ID, USER, HOST FROM information_schema.PROCESSLIST WHERE  
TIME > 2;
```

```
KILL CONNECTION ... Kill any remaining connections to the old master
```

+ Switchover - Pseudo code 2/3

```
/*For all the slaves*/
```

```
/*Wait for the slave up to date*/
```

```
$m[B|C]_cmd = "SELECT master_pos_wait('$old_master_file',  
$old_master_pos);";
```

 Gives back the hand when the slave is synchronized with the master

```
/*Stop the replication*/
```

```
$m[B|C]_cmd = "STOP SLAVE";
```

 Stop the replication

```
/*Retrive new master binlog info*/
```

```
$mB_cmd = "SHOW MASTER STATUS;";
```

On the promoted server.

```
$new_master_file = $row["File"];
```

Used to set the slaves on the new master

```
$new_master_pos = $row["Position"];
```

+ Switchover - Pseudo code 3/3

```
/*For all the new slaves*/
```

```
/*reset the old slave configuration*/
```

```
$mC_cmd = "RESET SLAVE;"; Reset replication informations on all the other slaves
```

```
/*Configure the new master*/
```

```
$mC_cmd = "CHANGE MASTER TO MASTER_HOST = '$new_master_host',  
MASTER_USER = '$new_master_user', MASTER_PASSWORD =  
'$new_master_pwd', MASTER_PORT=$new_master_port,  
MASTER_LOG_FILE='$new_master_file',  
MASTER_LOG_POS=$new_master_pos;"; Tell the slaves which is the new master
```

```
/*Start the replication*/
```

```
$mC_cmd = "START SLAVE"; Start replication on all the slaves
```

```
/*reset the replication info for the new master*/
```

```
$mB_cmd = "RESET SLAVE"; New master cleanup
```

+ Switchover - inspiration

➤ Audit et optimisation, MySQL 5 --- éditions Eyrolles (french book)

➤ Page 239

Soit A le serveur master, B le slave de A qui va être promu master et C un autre slave de A

1/ Interdire les écritures sur A:

```
SET GLOBAL read_only='ON';
```

 verrouille en lecture seule

```
FLUSH TABLES WITH READ LOCK;
```

 verrouille en lecture seule pour les autres comptes avec le droit SUPER

2/ Sauvegarder le numéro du journal binaire et la position de A :

```
SHOW MASTER STATUS;
```

paramètres : File, Position

3/ Laissez le serveur B & C rattraper leur retard au niveau de la réplication

```
SELECT master_pos_wait('mysql-bin.xxxxxx',N);
```

paramètres : File & Position de A (point 2).

La fonction rendra la main une fois l'esclave à jour

3/ Une fois B à jour,

assurez vous qu'il est configuré en master

log binaire

server-id unique

Utilisateur de réplication

exécutez `SHOW MASTER STATUS;`

File & Position serviront à reconfigurer A et C.

4/ Routez les clients sur B qui devient alors le serveur actif

5/ Reconfigurer les serveurs A et C pour qu'ils soient esclave de B et qu'il reparte sur le bon fichier

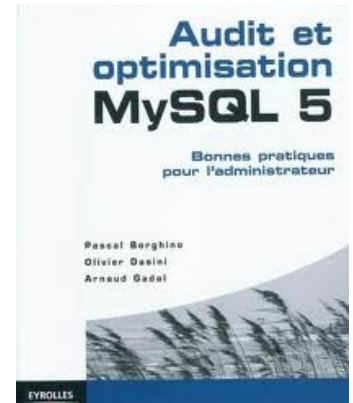
binaire et à la bonne position:

```
STOP SLAVE;
```

```
CHANGE MASTER TO MASTER_HOST="", MASTER_PORT=, MASTER_USER="", MASTER_PASSWORD="",  
MASTER_LOG_FILE = 'mysql-bin.xxxxxx', MASTER_LOG_POS = N;
```

A doit être configuré comme un slave

... **la suite dans le livre !)**



+ Recap

↳ Lessons learned from this ~~painful~~ experience

- ↳ Know you data
 - Know and understand the business logic
 - Respect the data
 - => Check them before you store them (in an ideal world)
 - => Otherwise you will pay sooner or later... (in the real world)
- ↳ Know MySQL
 - The charset that I need, the collation that I chose
 - Limits => RTFM
 - Rolling upgrade
 - Switchover
- ↳ Migrate the DB before migrating the application
 - See point 1...
- ↳ Automate everything that can
 - Scripting for ever

+ Merci

➤ The Viadeo team

- Anna
- Christophe
- Élodie
- Lionel
- Lourdes
- Marie Anne
- Nicolas
- Pierre(s)
- Sabri et son équipe
- Sandy
- Séverine
- Stéphane
- Sylvain
- Yorick
- ...

*Thank
You*

➤ The speakers

- Stéphane Combaudon
- Olivier Dasini
- Cédric Peintre
- Marc Thomas

➤ You...





+ Questions

- **Where to find the slides of the meetup ?**
 - <http://dasini.net/blog/presentations/>

