



Data Recovery for MySQL

Aleksandr Kuzminsky & Istvan Podor
Percona Live London 2011

Agenda

1. InnoDB files format overview
2. InnoDB dictionary (SYS_INDEXES, SYS_TABLES)
3. InnoDB Primary and Secondary indexes
4. Typical failure scenarios
5. InnoDB recovery tool

1. InnoDB format overview

InnoDB table spaces

- Shared table space (ibdataX)
 - dictionary, rollback segment, undo space, insert buffer, double write buffer
 - Primary & Secondary indexes
 - External pages (BLOBs)
- A table space per table (ibdataX + *.ibd)
 - Primary & Secondary indexes
 - External pages (BLOBs)

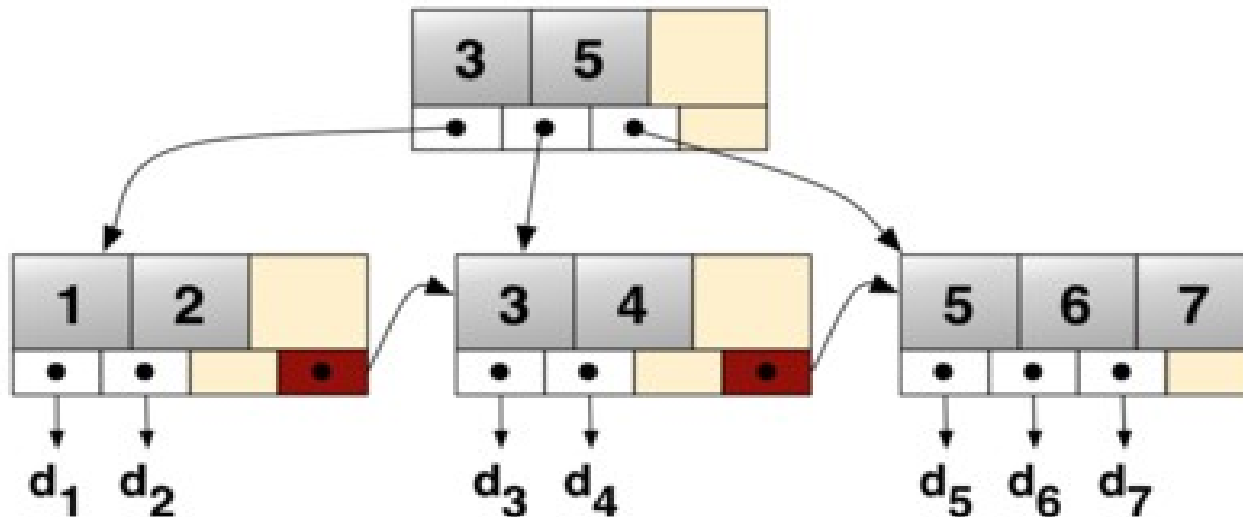
InnoDB logs

- a.k.a REDO logs, transactional logs
- ib_logfile[01]
- Contain changes to InnoDB pages:

Space ID	Page ID	Data
----------	---------	------

InnoDB Indexes. PRIMARY

- Table is a clustered index named PRIMARY
- B+ tree data structure, node is a page
- Key is Primary key or unique key or ROW_ID (6bytes)



InnoDB Indexes. Secondary

- Also B+ tree structure
- Key is indexed field(s), values are primary keys
 - If table (**id, first_name, last_name, birth_date**),
 - And index (**last_name**)
 - The index structure stores (**last_name, id**)

Index identifier (index_id)

- 8 bytes integer, often in two numbers notation:
0 254
- Table name → index_id correspondence is stored in InnoDB dictionary
- Visible in table monitor output(see next slide)
- In I_S tables if Percona Server

Table monitor output

```
mysql> CREATE TABLE innodb_table_monitor(x int)
engine=innodb
```

In Error log:

```
TABLE: name test/site_folders, id 0 119, columns 9, indexes 1, appr.rows 1
  COLUMNS: id: DATA_INT len 4 prec 0; name: type 12 len 765 prec 0;
  sites_count: DATA_INT len 4 prec 0;
              created_at: DATA_INT len 8 prec 0; updated_at:
DATA_INT len 8 prec 0;
              DB_ROW_ID: DATA_SYS prtype 256 len 6 prec 0; DB_TRX_ID:
DATA_SYS prtype 257 len 6 prec 0;
              DB_ROLL_PTR: DATA_SYS prtype 258 len 7 prec 0;
  INDEX: name PRIMARY, id 0 254, fields 1/7, type 3
  root page 271, appr.key vals 1, leaf pages 1, size pages 1
  FIELDS:  id DB_TRX_ID DB_ROLL_PTR name sites_count created_at
updated_at
```

InnoDB page format

InnoDB page is 16 kilobytes

	Size, bytes
FIL_HEADER	36
PAGE_HEADER	56
INFINUM+SUPREMUM RECORDS	varies
User records	varies
Free space	
Page directory	varies
FIL_TRAILER	fixed

Fil Header

- Common for all type of pages

Name	Size	Remarks
<i>FIL_PAGE_SPACE</i>	4	Space ID the page is in or checksum
<i>FIL_PAGE_OFFSET</i>	4	ordinal page number from start of space
<i>FIL_PAGE_PREV</i>	4	offset of previous page in key order
<i>FIL_PAGE_NEXT</i>	4	offset of next page in key order
<i>FIL_PAGE_LSN</i>	8	log serial number of page's latest log record
<i>FIL_PAGE_TYPE</i>	2	current defined types are: FIL_PAGE_INDEX , FIL_PAGE_UNDO_LOG , FIL_PAGE_INODE , FIL_PAGE_IBUF_FREE_LIST
<i>FIL_PAGE_FILE_FLUSH_LSN</i>	8	"the file has been flushed to disk at least up to this lsn" (log serial number), valid only on the first page of the file
<i>FIL_PAGE_ARCH_LOG_NO</i>	4	/* starting from 4.1.x this contains the space id of the page */

Page header

- Only for index pages

<i>Name</i>	<i>Size</i>	<i>Remarks</i>
PAGE_N_DIR_SLOTS	2	number of directory slots in the Page Directory part; initial value = 2
PAGE_HEAP_TOP	2	record pointer to first record in heap
PAGE_N_HEAP	2	number of heap records; initial value = 2
PAGE_FREE	2	record pointer to first free record
PAGE_GARBAGE	2	"number of bytes in deleted records"
PAGE_LAST_INSERT	2	record pointer to the last inserted record
PAGE_DIRECTION	2	either PAGE_LEFT, PAGE_RIGHT, or PAGE_NO_DIRECTION
PAGE_N_DIRECTION	2	number of consecutive inserts in the same direction, e.g. "last 5 were all to the left"
PAGE_N_RECS	2	number of user records
PAGE_MAX_TRX_ID	8	the highest ID of a transaction which might have changed a record on the page (only set for secondary indexes)
PAGE_LEVEL	2	level within the index (0 for a leaf page)
PAGE_INDEX_ID	8	identifier of the index the page belongs to
PAGE_BTR_SEG_LEAF	10	"file segment header for the leaf pages in a B-tree" (this is irrelevant here)
PAGE_BTR_SEG_TOP	10	"file segment header for the non-leaf pages in a B-tree" (this is irrelevant here)

How to check row format?

- Can be either COMPACT or REDUNDANT
- 0 stands for REDUNDANT, 1 - for COMACT
- The highest bit of the PAGE_N_HEAP from the page header
- ```
dc -e "2o `hexdump -C d pagefile |
grep 00000020 | awk '{ print $12}' `
p" | sed 's/./& /g' | awk '{ print
$1}'
```

# Rows in an InnoDB page

- Single linked list
- The first record – INFIMUM
- The last – SUPREMUM
- Ordered by PK

|      |    |          |
|------|----|----------|
| next |    | INFIMUM  |
| NULL |    | SUPREMUM |
| next | 10 | data     |
| next | 20 | data     |
| next | 30 | data     |
| next | 50 | data     |
| next | 40 | data     |

# Records are saved in insert order

```
insert into t1 values (10, 'aaa');
insert into t1 values (30, 'ccc');
insert into t1 values (20, 'bbb');
```

```
JG.....N<E.....
.....
.....2..
...infimum.....supremum.....6.
.....).....2..aaa.....
...*.....2..ccc.....+.
...2..bbb.....
.....
```

# Row format

```
CREATE TABLE `t1` (
 `ID` int(11) unsigned NOT NULL,
 `NAME` varchar(120),
 `N_FIELDS` int(10),
 PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT
CHARSET=latin1
```

| Name                 | Size                                               |
|----------------------|----------------------------------------------------|
| Offsets<br>(lengths) | 1 byte for small types<br>2 bytes for longer types |
| Extra bytes          | 5 bytes COMPACT<br>6 bytes REDUNDANT               |
| Field content        | varies                                             |

# Extra bytes(REDUNDANT)

| <i>Name</i>            | <i>Size</i> | <i>Description</i>                                                                       |
|------------------------|-------------|------------------------------------------------------------------------------------------|
| <i>record_status</i>   | 2 bits      | _ORDINARY, _NODE_PTR, _INFIMUM, _SUPREMUM                                                |
| <i>deleted_flag</i>    | 1 bit       | <b>1 if record is deleted</b>                                                            |
| <i>min_rec_flag</i>    | 1 bit       | 1 if record is predefined minimum record                                                 |
| <i>n_owned</i>         | 4 bits      | number of records owned by this record                                                   |
| <i>heap_no</i>         | 13 bits     | record's order number in heap of index page                                              |
| <i>n_fields</i>        | 10 bits     | <b>number of fields in this record, 1 to 1023</b>                                        |
| <i>1byte_offs_flag</i> | 1 bit       | 1 if each Field Start Offsets is 1 byte long (this item is also called the "short" flag) |
| <i>next 16 bits</i>    | 16 bits     | <b>pointer to next record in page</b>                                                    |

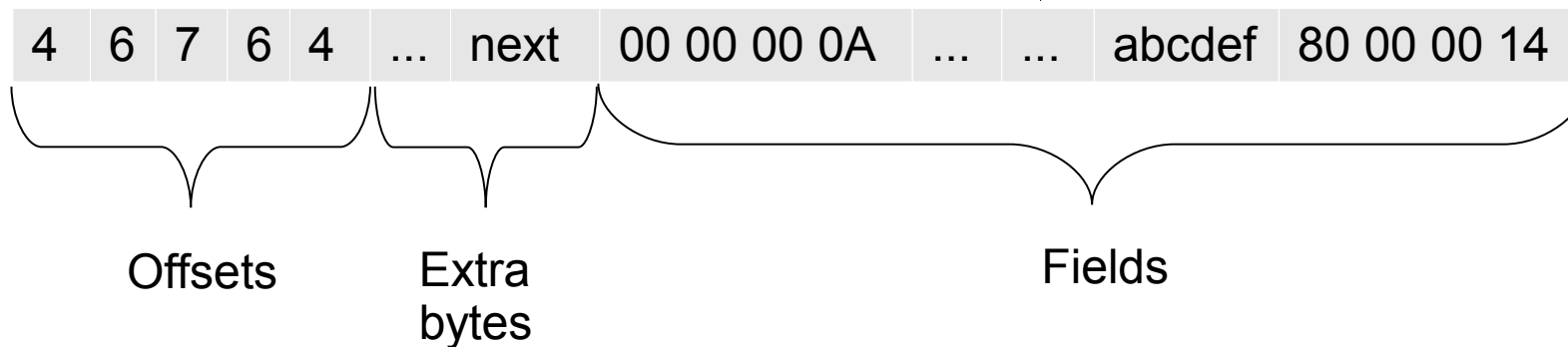
# Extra bytes(COMPACT)

| <i>Name</i>                                                        | <i>Size, bits</i> | <i>Description</i>                                                                              |
|--------------------------------------------------------------------|-------------------|-------------------------------------------------------------------------------------------------|
| <i>record_status</i><br><i>deleted_flag</i><br><i>min_rec_flag</i> | 4                 | 4 bits used to delete mark a record, and mark a predefined minimum record in alphabetical order |
| <i>n_owned</i>                                                     | 4                 | the number of records owned by this record<br>(this term is explained in page0page.h)           |
| <i>heap_no</i>                                                     | 13                | the order number of this record in the heap of the index page                                   |
| <i>record type</i>                                                 | 3                 | 000=conventional, 001=node pointer (inside B-tree), 010=infimum, 011=supremum, 1xx=reserved     |
| <i>next 16 bits</i>                                                | 16                | <b>a relative pointer to the next record in the page</b>                                        |

# Example: Redundant row

A row: (10, 'abcdef', 20)

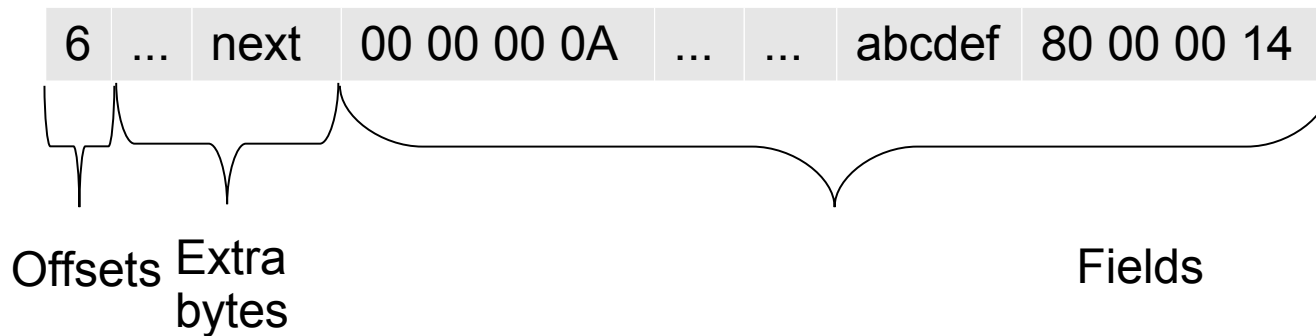
Actualy (10, TRX\_ID, PTR\_ID, 'abcdef', 20)



# The same row, but COMPACT

A row: (10, 'abcdef', 20)

Actually (10, TRX\_ID, PTR\_ID, 'abcdef', 20)



13 % smaller!

# Data types (highlights)

- Integer, float numbers are fixed size
- Strings:
  - VARCHAR(x) – variable
  - CHAR(x) – fixed if not UTF-8
- Date, time – fixed size
- DECIMAL
  - Stored as string before 5.0.3
  - Binary format afterward

# Data types (BLOBs)

- Field length (so called offset) is one or two bytes long
- If record size  $< (\text{UNIV\_PAGE\_SIZE}/2 - 200)$  == ~7k – the record is stored internally (in a PK page)
- Otherwise – 768 bytes in-page, the rest in an external page

---

# InnoDB dictionary (SYS\_INDEXES, SYS\_TABLES)

# Why are SYS\_\* tables needed?

---

- Correspondence “table name” → “index\_id”
- Storage for other internal information

# SYS\_\* structure

Always REDUNDANT format!

```
CREATE TABLE `SYS_INDEXES` (
 `TABLE_ID` bigint(20) unsigned NOT NULL
 default '0',
 `ID` bigint(20) unsigned NOT NULL default
 '0',
 `NAME` varchar(120) default NULL,
 `N_FIELDS` int(10) unsigned default NULL,
 `TYPE` int(10) unsigned default NULL,
 `SPACE` int(10) unsigned default NULL,
 `PAGE_NO` int(10) unsigned default NULL,
 PRIMARY KEY (`TABLE_ID`,`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```



**index\_id = 0-3**

```
CREATE TABLE `SYS_TABLES` (
 `NAME` varchar(255) NOT NULL default '',
 `ID` bigint(20) unsigned NOT NULL default
 '0',
 `N_COLS` int(10) unsigned default NULL,
 `TYPE` int(10) unsigned default NULL,
 `MIX_ID` bigint(20) unsigned default NULL,
 `MIX_LEN` int(10) unsigned default NULL,
 `CLUSTER_NAME` varchar(255) default NULL,
 `SPACE` int(10) unsigned default NULL,
 PRIMARY KEY (`NAME`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```



**index\_id = 0-1**

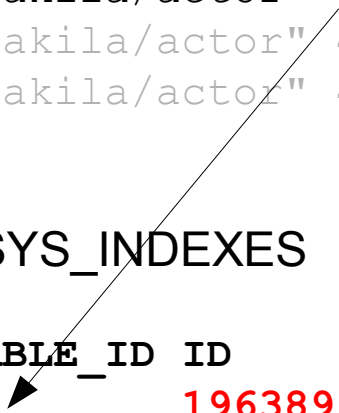
# Example: SYS\_\*

## SYS\_TABLES

| •NAME           | ID        | ...            |
|-----------------|-----------|----------------|
| •"sakila/actor" | <b>40</b> | 8 1 0 0 NULL 0 |
| •"sakila/actor" | 40        | 8 1 0 0 NULL 0 |
| •"sakila/actor" | 40        | 8 1 0 0 NULL 0 |

## SYS\_INDEXES

| •TABLE_ID   | ID            | NAME                  | ...            |
|-------------|---------------|-----------------------|----------------|
| • <b>40</b> | <b>196389</b> | "PRIMARY"             | 2 3 0 21031026 |
| •40         | 196390        | "idx_actor_last_name" | 1 0 0 21031028 |



---

# 3. InnoDB Primary and Secondary indexes

# PRIMARY Index

The table:

```
CREATE TABLE `t1` (
 `ID` int(11),
 `NAME` varchar(120),
 `N_FIELDS` int(10),
 PRIMARY KEY (`ID`),
 KEY `NAME` (`NAME`)
) ENGINE=InnoDB DEFAULT
 CHARSET=latin1
```

Fields in the PK:

1. **ID**
2. DB\_TRX\_ID
3. DB\_ROLL\_PTR
4. **NAME**
5. **N\_FIELDS**

# Secondary Index

The table:

```
CREATE TABLE `t1` (
 `ID` int(11),
 `NAME` varchar(120),
 `N_FIELDS` int(10),
 PRIMARY KEY (`ID`),
 KEY `NAME` (`NAME`)
) ENGINE=InnoDB DEFAULT
 CHARSET=latin1
```

Fields in the SK:

1. **NAME**
2. **ID** ← Primary key

---

# 4. Typical failure scenarios

# Wrong DELETE

- DELETE FROM `actor`;
- What happens?
  - Row(s) is marked as deleted
  - TRX\_ID and ROLL\_PTR are updated
  - Row remains in the page till UNDO log is purged
-

# Wrong DELETE

- What to do first?
  - Kill -9 mysqld\_safe ASAP!
  - Kill -9 mysqld ASAP
- Find pages which belong to the dropped table
- Fetch records from the pages(constraints\_parser from the recovery toolkit)

# Dropped Table/Database

- DROP TABLE actor;
- Very often DROP and then CREATE
- Bad because .frm files are removed
- Even worse when innodb\_per\_table
- What happens inside InnoDB:
  - Page is marked as free (or deleted tablespace)
  - A record is deleted from the dictionary

# Dropped Table/Database

- What to do?
  - Kill -9 safe\_mysql and mysqld
  - Remount a MySQL partition read-only
    - Or take an image
    - Or stop other services writing to the disk
- Fetch records from index pages  
(constraints\_parser from recovery toolkit)

# Truncate table

- What happens?
  - Equivalent to DROP/CREATE
  - Dictionary is updated, but index\_id may be reused
- What to do?
  - The same as after the DROP: kill mysqld, fetch records from the index

# Wrong UPDATE statement

- What happens?
  - If the new row is the same size, in-place update happens; otherwise – insert/delete
  - The old values go to UNDO space
  - roll\_ptr points to the old value in the UNDO space
- What to do?
  - Kill -9 safe\_mysqld, mysqld
  - However no tool available now

# Other wrongdoings

- Removed ibdata1 file
  - Take mysqldump ASAP before MySQL is stopped
  - Ibdconnect from recovery toolkit
- Wrong backups
  - innodb\_force\_recovery
  - Fetch records from index pages
- You name it

# Corrupt InnoDB tablespace

- Hardware failures
- OS or filesystem failures
- InnoDB bugs
- Corrupted InnoDB tablespace by other processes
- What to do?
  - `innodb_force_recovery`
  - `fetch_data.sh`
  - `constraints_parser`

---

# 5. InnoDB recovery tool

# Features

- A toolset to works with InnoDB at low level
  - `page_parser` – scans a bytes stream, finds InnoDB pages and sorts them by page type/index\_id
  - `constraints_parser` – fetches records from InnoDB page
  - `Ibdconnect` – a tool to “connect” an `.ibd` file to system tablespace.
  - `fetch_data.sh` – fetches data from partially corrupted tables choosing PK ranges.

# Recovery prerequisites

- Media
  - ibdata1
  - \*.ibd
  - HDD image
- Tables structure
  - SQL dump
  - \*.FRM files

# How to get CREATE info from .frm files

- Create table `actor` (id int) engine=InnoDB
- Stop MySQL and replace actor.frm
- Run MySQL with innodb\_force\_recovery=4
- SHOW CREATE TABLE actor;

# Percona Data Recovery Tool for InnoDB

<http://launchpad.net/percona-innodb-recovery-tool/>

**page\_parser** – splits InnoDB tablespace into 16k pages

**constraints\_parser** – scans a page and finds good records

# page\_parser

- Accepts a file

```
./page_parser -f /var/lib/mysql/ibdata1
```

- Produces:

- # ll pages-1319190009
- drwxr-xr-x 16 root root 4096 Oct 21 05:40 FIL\_PAGE\_INDEX/
- drwxr-xr-x 2 root root 12288 Oct 21 05:40 FIL\_PAGE\_TYPE\_BLOB/
- # ll pages-1319190009/FIL\_PAGE\_INDEX/
- drwxr-xr-x 2 root root 4096 Oct 21 05:40 0-1/
- drwxr-xr-x 2 root root 4096 Oct 21 05:40 0-3/
- drwxr-xr-x 2 root root 4096 Oct 21 05:40 0-18/
- drwxr-xr-x 2 root root 4096 Oct 21 05:40 0-19/

# constraints\_parser

- Accept a page or directory with pages

- # ./bin/constraints\_parser.SYS\_TABLES -4Uf pages-1319185222/FIL\_PAGE\_INDEX/0-1

|            |                    |    |             |   |   |   |
|------------|--------------------|----|-------------|---|---|---|
| SYS_TABLES | "SYS_FOREIGN"      | 11 | -2147483644 | 1 | 0 | 0 |
| SYS_TABLES | "SYS_FOREIGN_COLS" | 12 | -2147483644 | 1 | 0 | 0 |
| SYS_TABLES | "test/t1"          | 16 | 3           | 1 | 0 | 0 |

- Table structure is defined in "include/table\_defs.h"
- Prints LOAD DATA INFILE to stderr

# ibdconnect

- Create empty InnoDB tablespace
- Create the table:

```
mysql>CREATE TABLE actor (
```

- actor\_id SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,
- first\_name VARCHAR(45) NOT NULL,
- last\_name VARCHAR(45) NOT NULL,
- last\_update TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,
- PRIMARY KEY (actor\_id),
- KEY idx\_actor\_last\_name (last\_name)
- )ENGINE=InnoDB DEFAULT CHARSET=utf8;

# ibdconnect

- Update InnoDB dictionary (MySQL is down now)

```
ibdconnect -o /var/lib/mysql/ibdata1 -f
/var/lib/mysql/sakila/actor.ibd -d sakila -t actor
```

- Fix InnoDB checksums:

```
./innochecksum -f /var/lib/mysql/ibdata1
```

```
./innochecksum -f /var/lib/mysql/ibdata1
```

- Start MySQL and take mysqldump

# Thank You to Our Sponsors

## Platinum Sponsor



## Gold Sponsor



## Silver Sponsors



# Percona Live London Sponsors

## Exhibitor Sponsors



## Friends of Percona Sponsors

**Couchbase**



Monty Program



**Tokutek**



## Media Sponsors



**O'REILLY**

# Annual MySQL Users Conference

## *Presented by Percona Live*

The Hyatt Regency Hotel, Santa Clara, CA

April 10th-12th, 2012

### Featured Speakers

Mark Callaghan, Facebook

Jeremy Zawodny, Craigslist

Marten Mickos, Eucalyptus Systems

Sarah Novotny, Blue Gecko

Peter Zaitsev, Percona

Baron Schwartz, Percona

The Call for Papers is Now Open!

Visit [www.percona.com/live/mysql-conference-2012/](http://www.percona.com/live/mysql-conference-2012/)



**aleksandr.kuzminsky@percona.com**  
**istvan.podor@percona.com**

**We're Hiring! [www.percona.com/about-us/careers/](http://www.percona.com/about-us/careers/)**



PERCONA  
LIVE

[www.percona.com/live](http://www.percona.com/live)