



ORACLE[®]

Mysteries of the Binary Log

Mats Kindahl

Lead Replication Developer

Charles Bell

Lead Backup Developer

About the Speakers

·Mats Kindahl, PhD

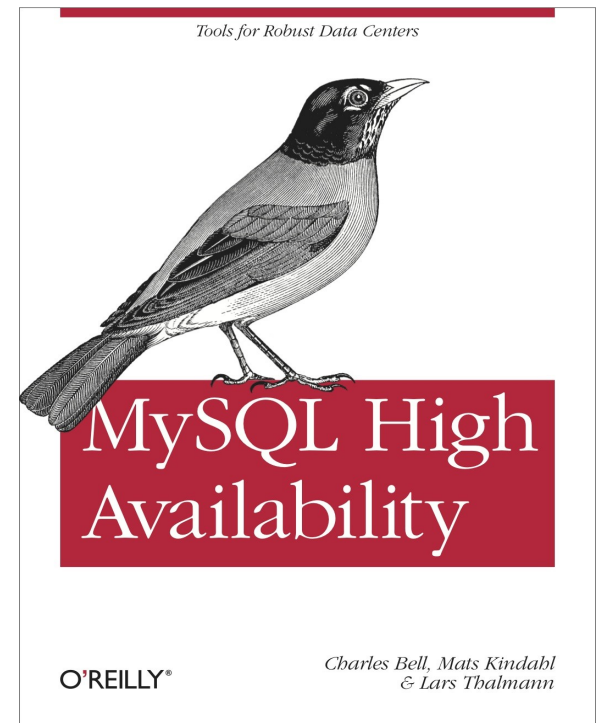
- Replication Expert and Lead Developer
- mats.kindahl@sun.com

·Chuck Bell, PhD

- Enterprise Backup and Replication
- chuck.bell@oracle.com

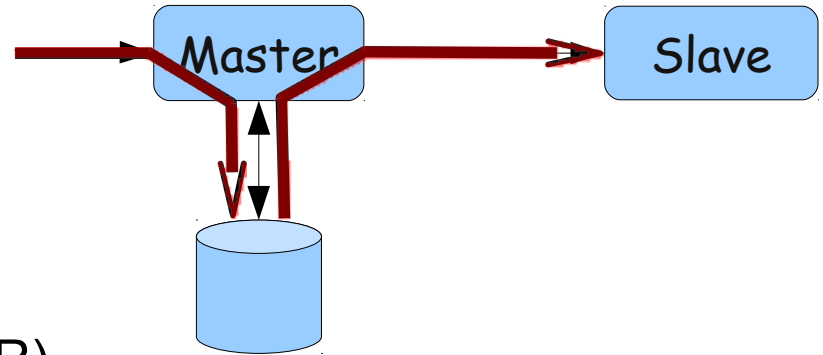
·Lars Thalmann, PhD

- Development Manager, Replication and Backup
- lars.thalmann@sun.com



What is the binary log?

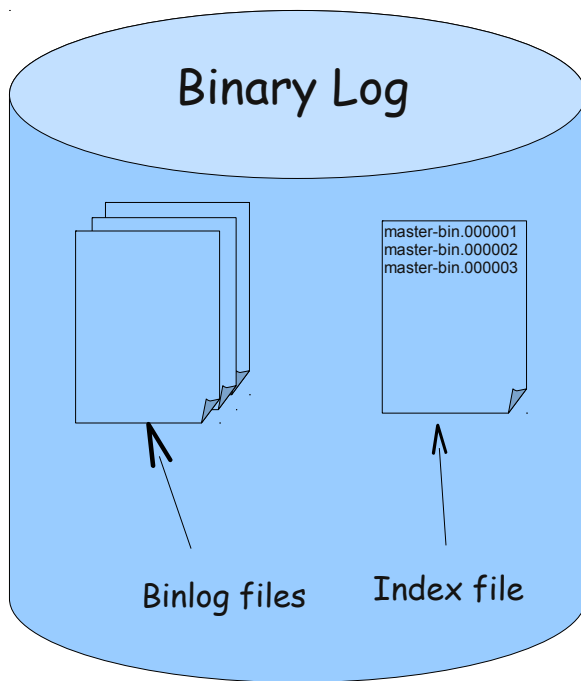
- Record of changes
- Used for
 - Replication
 - Auditing
 - Point-In-Time Recovery (PITR)
- Slave executes changes with privileges turned off
 - Security implications?
 - “If it is OK to execute on the master, it should be OK to execute on the slave”
 - Not always true (as you will see)



Best Practices

- Manage your log rotations
- Protect your logs
 - Store on secure location
 - Don't store on same disk as data
- Purge old logs
 - PURGE command
 - --expire-log-days
- Use log filters sparingly if point-in-time recovery a priority
- Protect the replication user account from tampering
- Avoid using sensitive data in statements (e.g. passwords)

Structure of the Binary Log



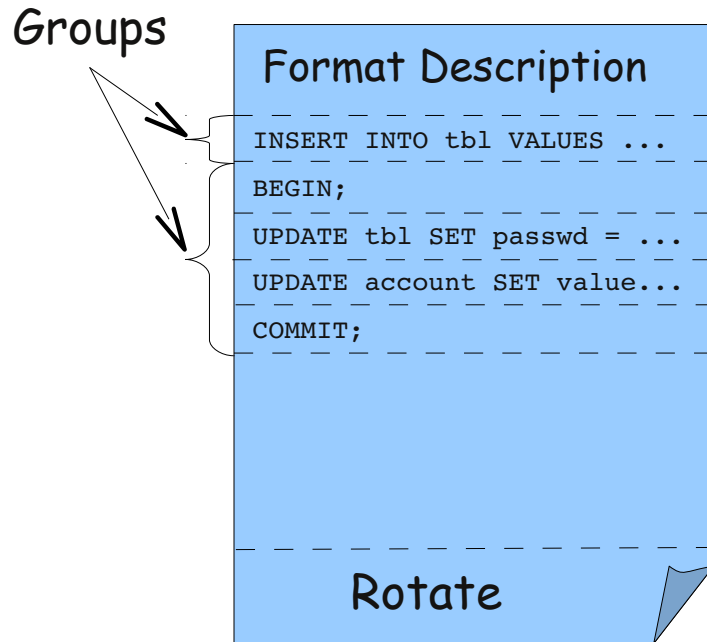
Binlog files

- Option `log-bin`
- Default `master-bin.000001`
- Content of binary log

Binlog index

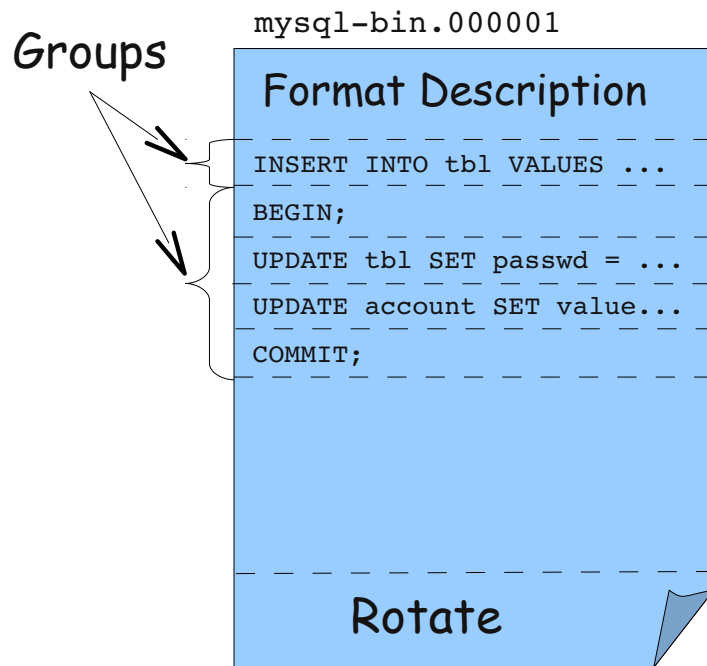
- Option `log-bin-index`
- Default `master-bin.index`
- Index over binlog files

Binlog file structure



- Format Description Event
 - File-specific data
 - Binlog Format Version
 - Server Version
- Rotate
 - Terminate binlog file
 - Next file in sequence
- Binlog Events
 - Organized in groups
 - MySQL 5.1 have 26 different event types

Binlog file structure



- Binary Log Coordinate
 - File name
 - File position

Investigating Binary Log

- SHOW BINLOG EVENTS
 - IN *file*
 - FROM *position*
 - LIMIT *events*
- Shows contents of first binlog file (!)
 - **Not** contents of last binlog file

```
mysql> SHOW BINLOG EVENTS;
```

Log_name	Pos	Event_type	Server_id	End_log_pos	Info
mysql-bin.000001	4	Format_desc	1	106	Server ver: 5.1.37-lubuntu5.1-log, Binlog ver: 4
mysql-bin.000001	106	Query	1	250	use `test`; CREATE TABLE book (id INT UNSIGNE...
mysql-bin.000001	250	Query	1	373	use `test`; CREATE TABLE author (name VARCHAR...

```
3 rows in set (0.00 sec)
```

Size = End_log_pos - Pos

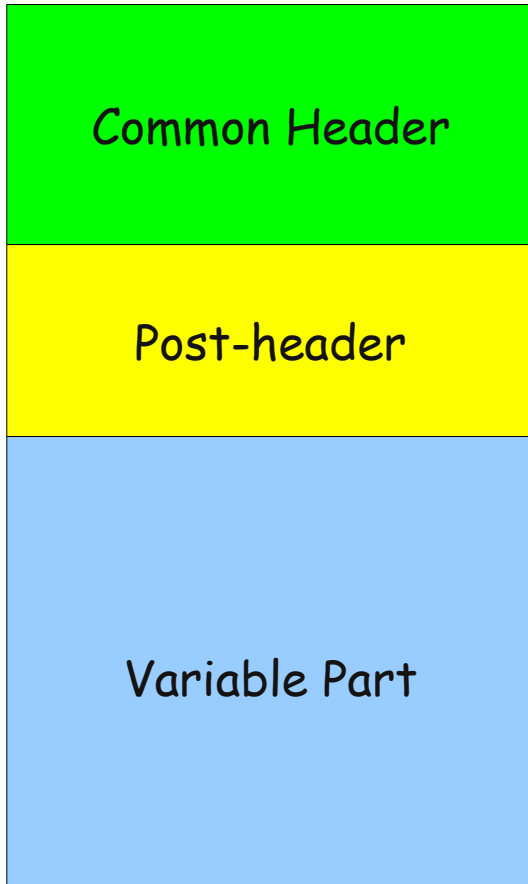
Purging Binlog Files

- PURGE BINARY LOG TO *filename*
Deletes all binary log files before the named file.
- PURGE BINARY LOG BEFORE *datetime*
Purge will always delete complete files. This means that if there is at least one event in the log file that has a time stamp after datetime, the file is not deleted.
- RESET MASTER
 - Deletes all binary log files listed in the index file, resets the index, and creates a new binlog file.

Purging Binlog Files

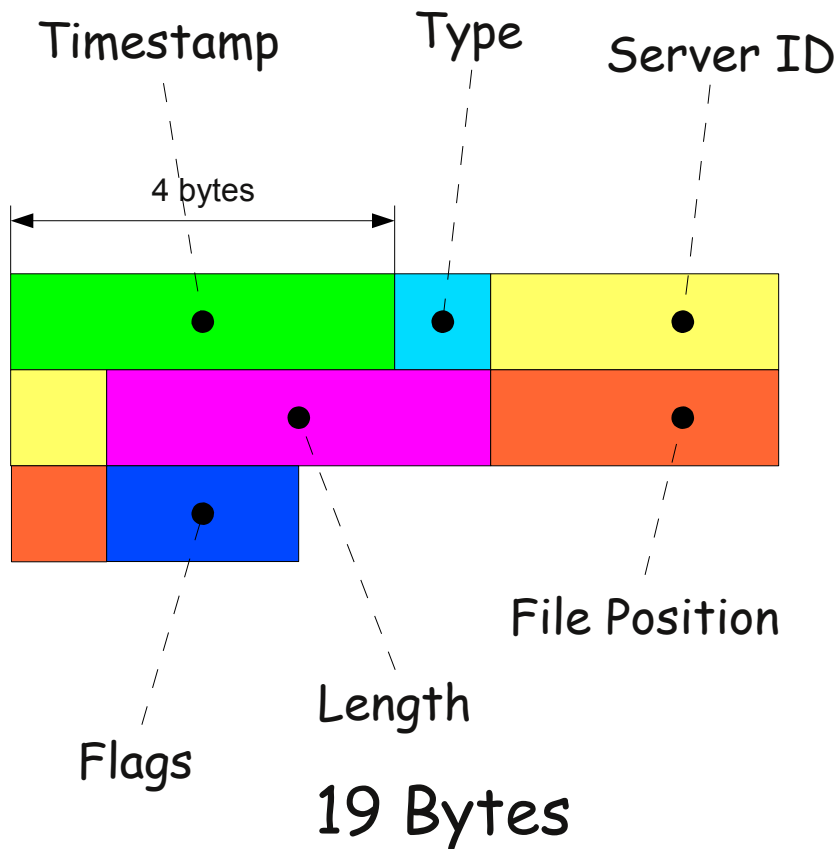
- Automatically purge logs
 - Server_variable: `expire_logs_days`
 - Removes the binlog files that are at least that old
 - The removal happens at server start or log flush

Binlog Event Structure



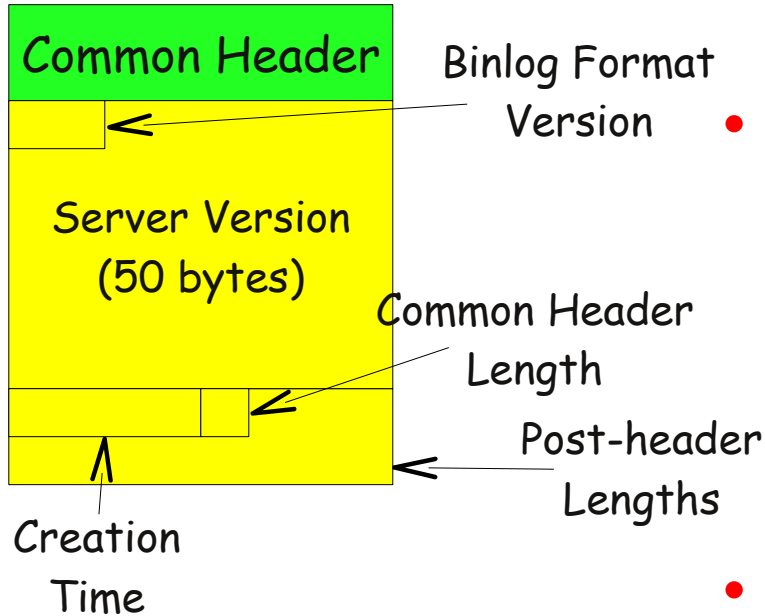
- Common header
 - Generic data
 - Fixed size
- Post-header
 - Event-specific data
 - Fixed size
- Variable part
 - Event-specific data
 - Variable size

Binlog Event Common Header



- Data common to all events
- File Position
 - *End* of event
- Timestamp
 - Statement *start* time
- Flags
 - Binlog-in-use
 - Thread-specific
 - Suppress “use”
 - Artificial
 - Relay-log event

Format Description Event



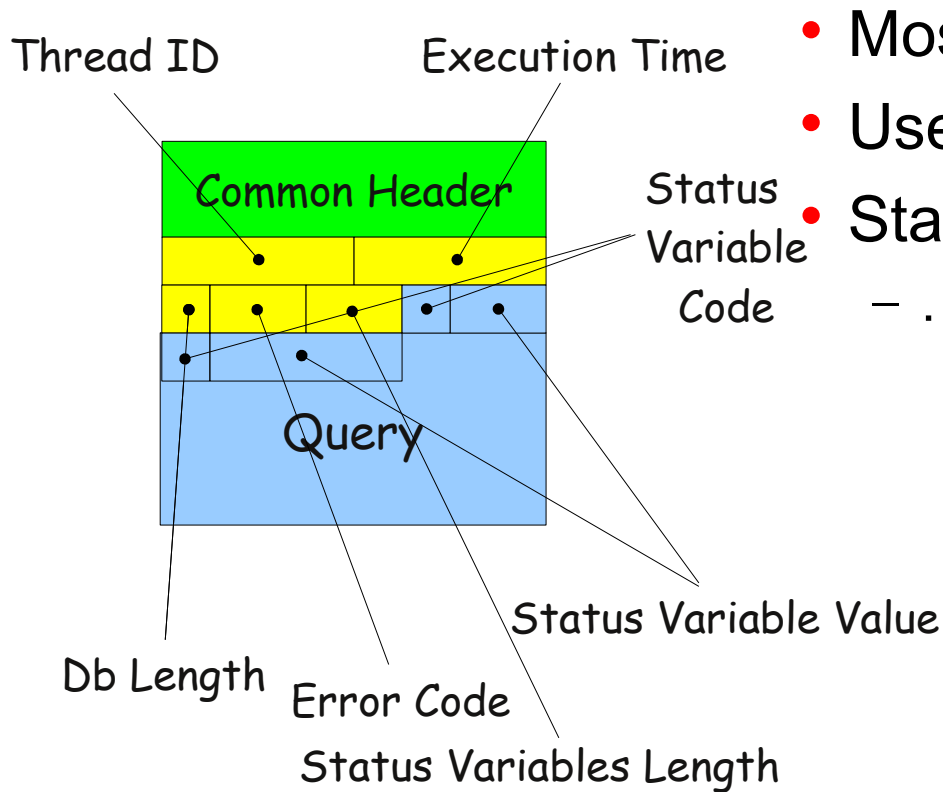
- Describes file information
 - Different files can have different information
 - Design for extensibility
 - Common header length
 - Post-header lengths
- Fixed size!

```
mysql> SHOW BINLOG EVENTS;
```

Log_name	Pos	Event_type	Server_id	End_log_pos	Info
mysql-bin.000001	4	Format_desc	1	106	Server ver: 5.1.37-lubuntu5.1-log, Binlog ver: 4
mysql-bin.000001	106	Query	1	250	use `test`; CREATE TABLE book (id INT UNSIGNED...
mysql-bin.000001	250	Query	1	373	use `test`; CREATE TABLE author (name VARCHAR...

```
3 rows in set (0.00 sec)
```

Query Event



- Most common event
- Used for statements
- Statement logged literally
 - ... in almost all cases

Handling Statement Context

- SQL_MODE
- Time functions
 - NOW(), CURDATE(), CURTIME(), UNIX_TIMESTAMP()
 - SYSDATE()
- Auto-increment handling
 - Insert into AUTO_INCREMENT column
 - Using LAST_INSERT_ID()
 - @@auto_increment_increment
 - @@auto_increment_offset
- User-defined variables
- Seed for RAND()
- Character set

Status Variables

- Added in query event
 - Only when needed
- SQL_MODE
- Catalog
- auto_increment_increment
- auto_increment_offset
- Character Set
- Time Zone

Time Functions

- Execution start time is saved for session
 - Recorded in binlog event
- Some functions use statement start time
 - NOW(), CURTIME(), CURDATE(), UNIX_TIMESTAMP() OK
- Some functions call *time(2)* directly
 - SYSDATE()

Warning!

Context Events

- Context events are used for:
 - User-defined variables
 - RAND() seeds
 - AUTO_INCREMENT
 - LAST_INSERT_ID()
- Context event(s) before Query event
 - There can be several context events before a Query
- Context event(s) + Query event = Binlog Group

Rand Event: RAND()

- For statements that use RAND () function
INSERT INTO tbl VALUE (RAND())
- RAND event precedes query
- Hold two seed values used by RAND () on slave

```
master> SHOW BINLOG EVENTS IN 'mysqld1-bin.000004' FROM 336;
```

Log_name	Pos	Event_type	Server_id	End_log_pos	Info
mysqld1...	336	RAND	1	371	rand_seed1=677022992,rand_seed2=870104260
mysqld1...	371	Query	1	465	use `test` ; INSERT INTO tbl VALUES (RAND())

```
2 rows in set (0.00 sec)
```

Intvar Event: AUTO_INCREMENT

- Inserting into an AUTO_INCREMENT column
INSERT INTO book(title)
VALUES("MySQL High Availability")
- Type = INSERT_ID
- Value = *integer*

```
mysql> SHOW BINLOG EVENTS FROM 373;
```

Log_name	Pos	Event_type	Server_id	End_log_pos	Info
mysql...	373	Intvar	1	401	INSERT_ID=1
mysql...	401	Query	1	522	use `test` ; INSERT INTO book...

```
2 rows in set (0.00 sec)
```

Intvar Event: AUTO_INCREMENT

- Using LAST_INSERT_ID()

```
INSERT INTO author(name, book_id) VALUES
('Charles Bell', LAST_INSERT_ID()),
('Mats Kindahl', LAST_INSERT_ID()),
('Lars Thalmann', LAST_INSERT_ID());
```

- Type = LAST_INSERT_ID
- Value = *integer*

```
mysql> SHOW BINLOG EVENTS FROM 522;
```

Log_name	Pos	Event_type	Server_id	End_log_pos	Info
mysql...	522	Intvar	1	550	LAST_INSERT_ID=1
mysql...	550	Query	1	746	use `test`; INSERT INTO auth...

```
2 rows in set (0.00 sec)
```

User_var Event: User-Defined Variables

- Using user variable

```
SET @LID = LAST_INSERT_ID();
SET @OReilly = "O'Reilly Media";
UPDATE book SET publisher = @OReilly
WHERE id = @LID;
```

- Typed: STRING, REAL, INT, DECIMAL

```
mysql> SHOW BINLOG EVENTS FROM 1021;
```

Log_name	Pos	Event_type	Serv...	End_log_pos	Info
mysql...	1021	User var	1	1048	@`LID`=1
mysql...	1048	User var	1	1102	@`Pub`= latin1 0x4F... COLLATE latin1_swedish
mysql...	1102	Query	1	1217	use `test` ; UPDATE book SET publisher = @P...

```
3 rows in set (0.00 sec)
```

Security and the Binary Log

- The replication user with the REPLICATION SLAVE privilege can read everything
- You must protect this account from exploitation
- Precautions
 - Prohibit login from outside the firewall
 - Audit the account and place log on a secure location
 - Use encrypted connection (e.g. SSL)
- But... even if you heed these precautions, it isn't enough if the data in the binary log is compromised
- Avoid sensitive data in the binary log like passwords

Securing Sensitive Data

- This is bad:

```
UPDATE employee SET pass = PASSWORD('foobar')  
WHERE email = 'mats@example.com';
```

- Rewrite the statement to use user-defined variables.
- This is good:

```
SET @password = PASSWORD('foobar');  
UPDATE employee SET pass = @password WHERE email =  
'mats@example.com';
```

- SET statement is not replicated
- No password written to the binary log

Stored Programs

- Stored Procedure
 - CREATE PROCEDURE
 - CALL
- Stored Function
 - CREATE FUNCTION
- Triggers
 - CREATE TRIGGER
- Events
 - CREATE EVENT
- Logging
 - Stored Program Definitions?
 - Stored Program Executions?

Stored Procedures Definition

- Always written as statement
- Requires one of
 - DETERMINISTIC
 - NO SQL
 - READS SQL DATA
- There is no check that you follow protocol!

```
CREATE PROCEDURE add_author(  
    book_id INT,  
    name VARCHAR(64)  
)  
    DETERMINISTIC  
    SQL SECURITY INVOKER  
BEGIN  
    INSERT INTO author  
        VALUES (book_id, name);  
END
```

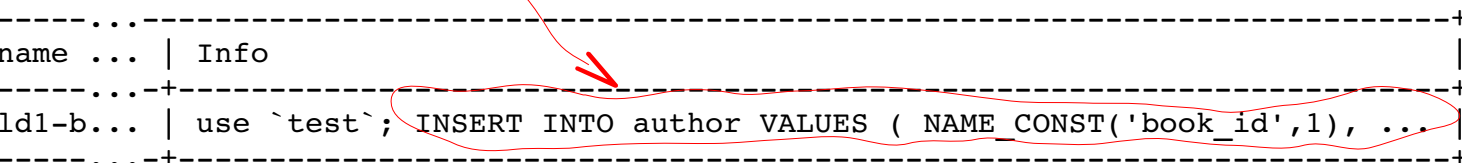
Stored Procedure Call

- Slave execute with privileges off
 - “CALL my_exploit(@@server_id = 1)”
 - Security Issue?
- Execution is unrolled
 - Actual statements executed are written to binary log
 - Procedure parameters replaced using NAME_CONST

```
mysql> CALL add_author(1, 'Mats Kindahl');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SHOW BINLOG EVENTS FROM 3457;
```

```
+-----+-----+  
| Log_name ... | Info |  
+-----+-----+  
| mysqld1-b... | use `test`; INSERT INTO author VALUES ( NAME_CONST('book_id',1), ... |  
+-----+-----+  
1 row in set (0.00 sec)
```



Stored Procedures Definition

- What about definer?
 - Statement executed by slave thread
- User on slave thread?
 - Not normally
 - Who's definer on slave?
- Rewritten using DEFINER

```
CREATE PROCEDURE add_author(  
    book_id INT,  
    name VARCHAR(64)  
)  
    DETERMINISTIC  
    SQL SECURITY DEFINER  
BEGIN  
    ...  
END
```

```
mysql> SHOW BINLOG EVENTS FROM 3672;
```

```
+-----+-----+  
| Log_n... | Info |  
+-----+-----+  
| mysql... | use `test`; CREATE DEFINER=`mats`@`localhost` PROCEDURE `add_author` (... END |  
+-----+-----+  
2 rows in set (0.00 sec)
```

Triggers and Events

Definitions:

- Similar to stored procedure definitions, trigger and event definitions use DEFINER clause
- Event *definitions* are replicated to the slave as slave-side-disabled
 - If you want the replicated events enabled on the slave, you need to enable them manually

Execution:

- Event *execution* effects are replicated

Trigger Execution

```
CREATE TABLE employee (name VARCHAR(64));
CREATE TABLE log (name VARCHAR(64));
CREATE TRIGGER tr_employee BEFORE UPDATE ON employee
FOR EACH ROW
  INSERT INTO log VALUES
    (CONCAT(OLD.name, " changed to ", NEW.name))
```

- What about this statement?

```
INSERT INTO employee VALUES ("Chuck")
```

- What is in the binary log with respect to the trigger?
 - Nothing
 - Trigger definitions can be different on master and slave
 - ... even non-existent on master or slave
 - Trigger effects do not have to be replicated
 - ... because trigger fires on the slave

Stored Function Execution

```
CREATE FUNCTION get_employee(who VARCHAR(64))
  RETURNS VARCHAR(64)
BEGIN
  DECLARE result VARCHAR(64);
  SELECT name INTO result FROM employee WHERE name = who;
  RETURN result;
END
```

- Consider this statement

```
INSERT INTO authors
  VALUES (get_employee("Chuck"));
```

- Statement is logged as entered by user
 - Similar to triggers
 - Definitions can be different on master and slave

Stored Function Execution

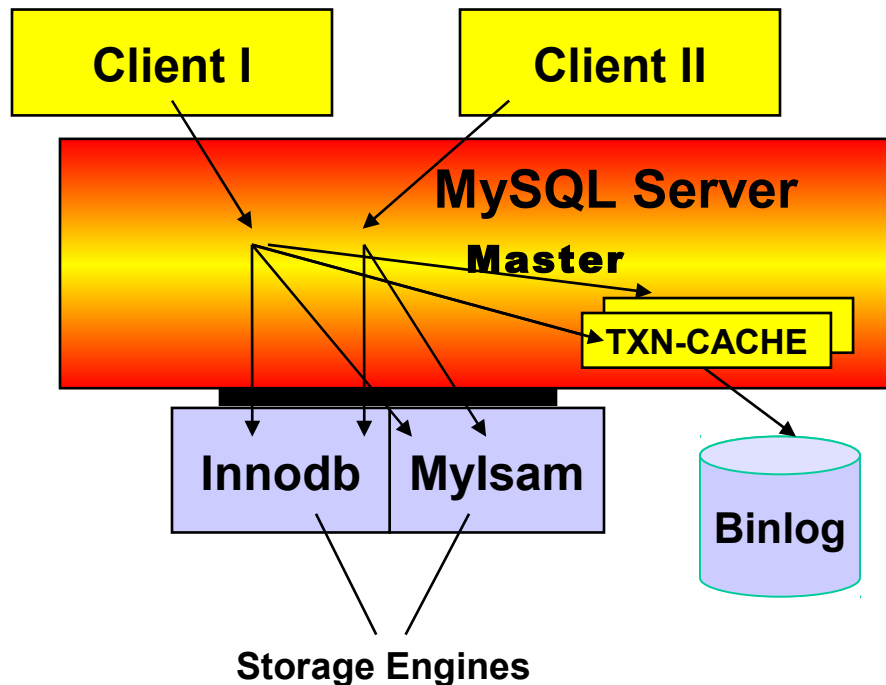
```
CREATE FUNCTION get_employee(who VARCHAR(64))
  RETURNS VARCHAR(64)
  SQL SECURITY INVOKER
BEGIN
  DECLARE result VARCHAR(64);
  IF @@server_id = 1 THEN
    SELECT name INTO result FROM employee WHERE name = who;
  ELSE
    SELECT name INTO result FROM secret_agents LIMIT 1;
  END IF
  RETURN result;
END
```

- Consider this statement:

```
INSERT INTO author
  VALUES (get_employee("Chuck"));
```

- Executed without privileges on slave!
 - ... and executes different code on master and slave!
- CREATE FUNCTION requires SUPER privileges
 - log-bin-trust-function-creators

Replication Architecture



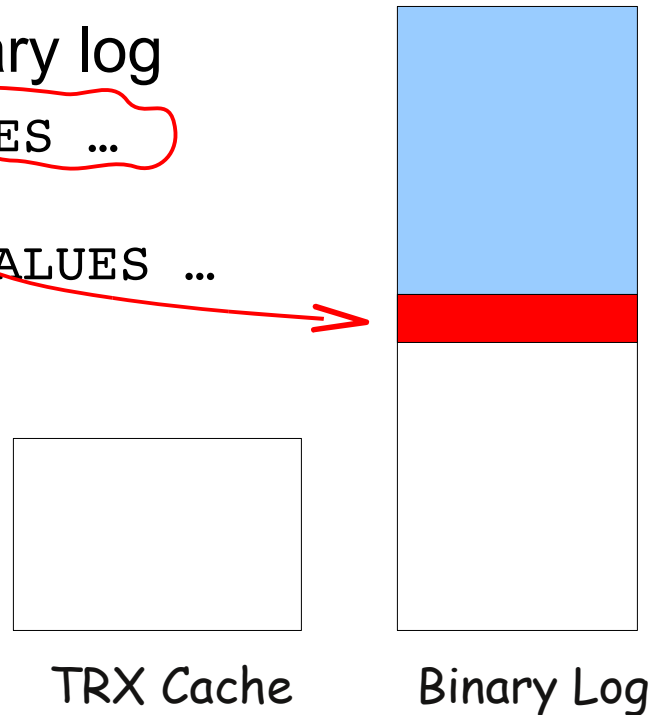
Mixing InnoDB and MyISAM changes in the same transaction can cause slave to be inconsistent with the master

- MyISAM changes are visible immediately when statement ends
- Transactional cache is flushed at commit time

Mixing Engines in Transactions #1

- Non-transactional change outside a transaction go directly to binary log

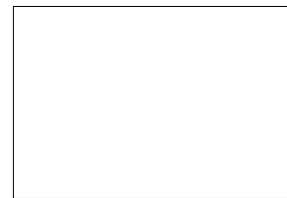
```
INSERT INTO myisam VALUES ...  
BEGIN;  
INSERT INTO my_innodb VALUES ...  
.  
.  
.  
COMMIT;
```



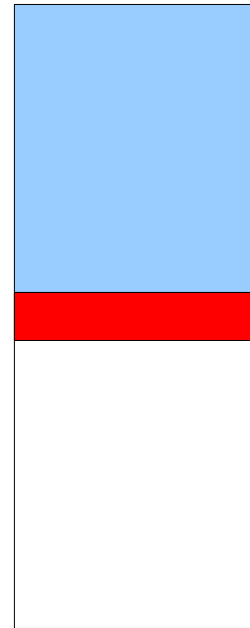
Mixing Engines in Transactions #2

- Non-transactional statement goes directly to binary log if transaction cache is empty

```
BEGIN;  
INSERT INTO myisam VALUES ...  
INSERT INTO my_innodb VALUES ...  
.  
.  
.  
COMMIT;
```



TRX Cache

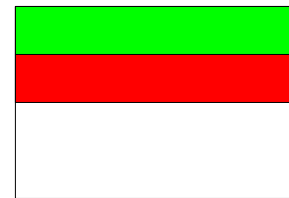


Binary Log

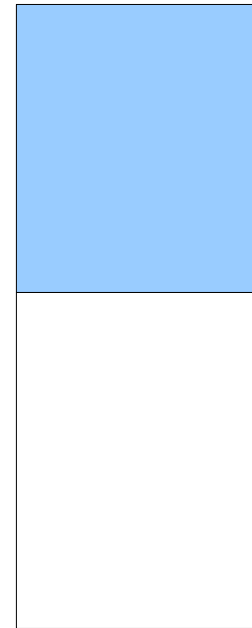
Mixing Engines in Transactions #3

- Non-transactional statement goes to transaction cache if not empty

```
BEGIN;  
INSERT INTO my_innodb VALUES ...  
INSERT INTO myisam VALUES ...  
.  
.  
.  
COMMIT;
```



TRX Cache



Binary Log

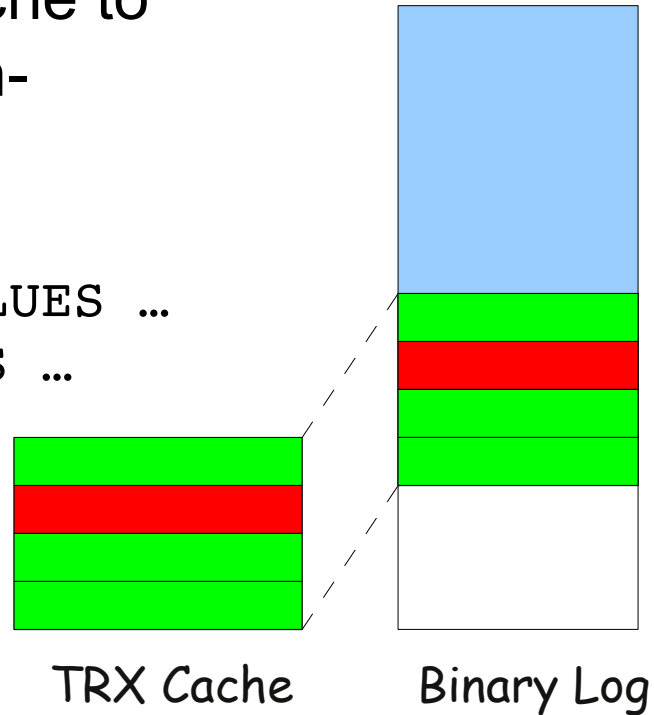
MYSQL 5.5

--binlog-direct-non-transactional-changes

Mixing Engines in Transactions #4

- Rollback writes transaction cache to binary log if it contains any non-transactional changes

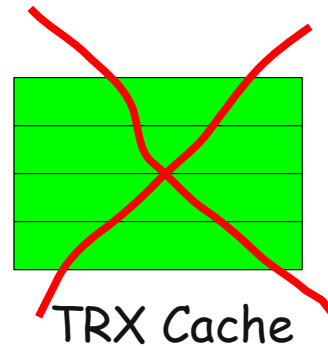
```
BEGIN;  
INSERT INTO my_innodb VALUES ...  
INSERT INTO myisam VALUES ...  
.  
.  
.  
ROLLBACK;
```



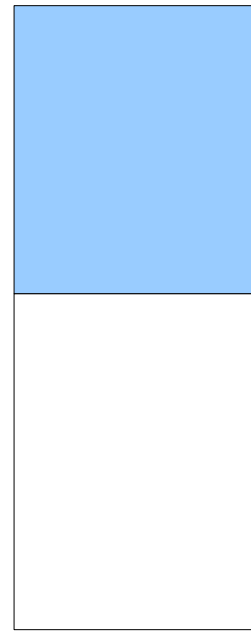
Mixing Engines in Transactions #5

- Rollback clears transaction cache if it contains only transactional changes

```
BEGIN;  
INSERT INTO my_innodb VALUES ...  
.  
.  
.  
ROLLBACK;
```



TRX Cache



Binary Log

Mixing Engines in Statements

```
CREATE TABLE employee (name VARCHAR(64)) ENGINE=InnoDB;  
CREATE TABLE log (name VARCHAR(64)) ENGINE=MyISAM;  
CREATE TRIGGER tr_employee BEFORE UPDATE ON employee  
FOR EACH ROW  
  INSERT INTO log VALUES  
    (CONCAT(OLD.name, " changed to ", NEW.name))
```

- What about this statement?

```
UPDATE employee SET name = "Charles"  
WHERE name = "Chuck";
```

- Transactional or non-transactional?
 - If a statement contain any non-transactional changes, it is considered non-transactional

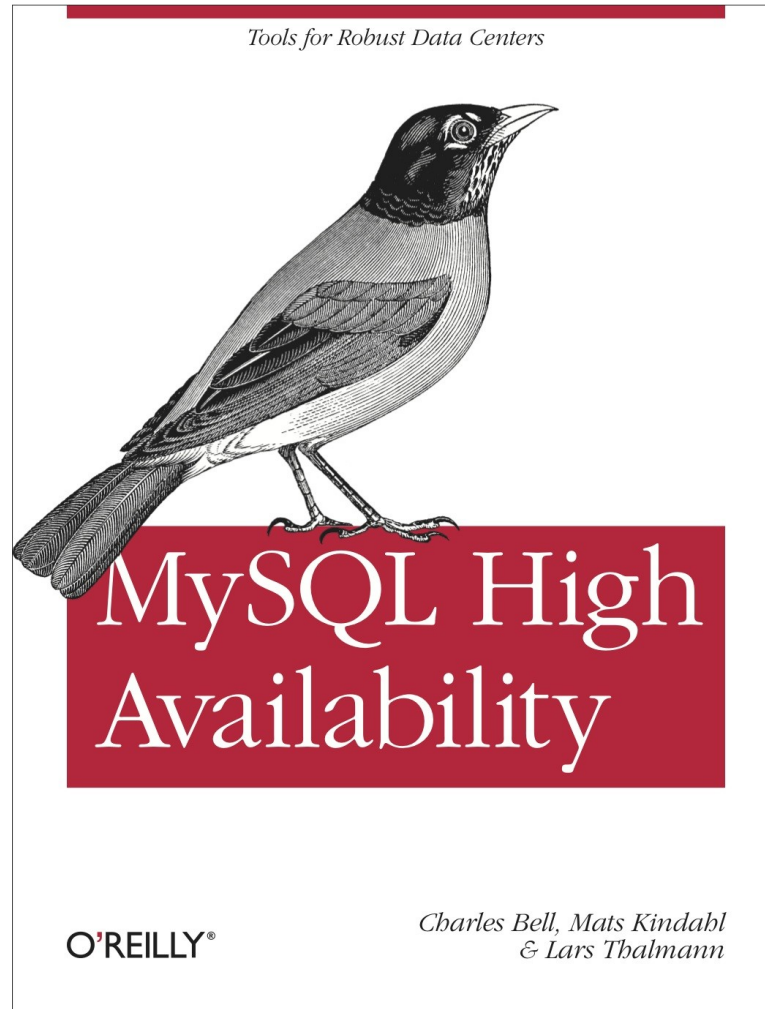
Mixing Transactional and Non-transactional statements

- You can do this, but it has consequences
- Considerations
 - Non-transactional statements inside the transaction are committed implicitly – and written to the binary log
 - If the transactional statements fail, they are not written to the binary log
- Implicit commits
 - CREATE, ALTER
 - Modifications to mysql database (post 5.1.3)
 - Pragmatic causes (e.g., LOAD DATA INFILE)
- Avoiding the problem
 - Place non-transactional statements first
 - If you need values from these statements, use temporary tables or variables

Questions?

Buy this book!

**Coming soon:
June 2010**



Buy the book!



ORACLE®