

5.1 In-Depth

Brian Aker

Director of Technology
(brian@mysql.com)

April 2007

MySQL AB





Business Intelligence/Data Warehousing

- Table/Index Partitioning
- Full Text Search Enhancements
- Better XML Handling - XPath
- Archive engine enhancements

High Availability

- Disk-based Cluster
- Row-based Replication
- Cluster replication

Easier Manageability

- Task Scheduler
- Transaction support for Federated Engine
- CREATE SERVER

Higher Performance

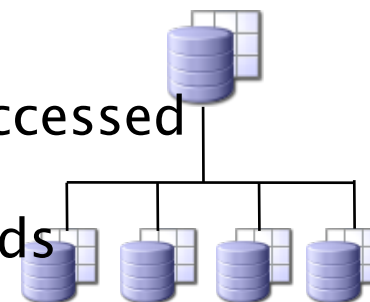
- Faster alter table
- Faster add/drop index for MySQL Cluster
- Faster data import operations
- Better problem user and SQL identification
- New Performance/Load Testing Utility
- Customized Engines



Business Intelligence/Data Warehousing

Table/Index Partitioning

- Perfect for data warehouses or other VLDB situations
- Increases performance – only needed partitions are accessed
- Eases space/data management burden on DBA
- Supports range, hash, key, list, and composite methods
- Supported by all storage engines in MySQL
- More options than Microsoft, DB2, or Sybase partitioning
- Partition key must be integer (or convertible to integer)
- Local indexes only – no global primary or unique keys
- Parallelism not supported in this version (except in NDB Cluster engine)



Partitioning Examples

- Range Partitioning – good for segregating historical data or other logical separations, such as retail store identifiers.

```
CREATE TABLE RANGE_BY_DATE
(
    CUSTOMER_NUMBER int NOT NULL,
    CUSTOMER_ORDER VARCHAR(50) NOT NULL,
    CUSTOMER_ORDER_DATE DATETIME NOT NULL)
PARTITION BY RANGE (YEAR (CUSTOMER_ORDER_DATE))
(
    PARTITION P1 VALUES LESS THAN (2000) ,
    PARTITION P2 VALUES LESS THAN (2003) ,
    PARTITION P3 VALUES LESS THAN (2005) ,
    PARTITION P4 VALUES LESS THAN MAXVALUE
)
```

Partitioning Examples

- Hash Partitioning – good for spreading out data across a variety of physical drives for storage and reduction in I/O contention. Must choose hash key carefully.

```
CREATE TABLE HASH_EXAMPLE (col1 INT, col2 CHAR(5),
    col3 DATE)
PARTITION BY HASH(col1)
(
PARTITION P1 DATA DIRECTORY = '/.../mysql51/data',
PARTITION P2 DATA DIRECTORY = '/.../mysql51/data2',
PARTITION P3 DATA DIRECTORY = '/.../mysql51/data3',
PARTITION P4 DATA DIRECTORY = '/.../mysql51/data4'
)
;
```

Partitioning Examples

- Key Partitioning – Like Hash, but MySQL guarantees even distribution of data. Must use all or part of a table's PK.

```
CREATE TABLE HASH EXAMPLE (col1 INT primary key,  
    col2 CHAR(5), col3 DATE)  
PARTITION BY KEY(col1)  
(  
PARTITION P1 DATA DIRECTORY = '/.../mysql51/data',  
PARTITION P2 DATA DIRECTORY = '/.../mysql51/data2',  
PARTITION P3 DATA DIRECTORY = '/.../mysql51/data3',  
PARTITION P4 DATA DIRECTORY = '/.../mysql51/data4'  
)  
;
```

Partitioning Examples

- List Partitioning – Helpful for distribution of data in need of logical separation and access – like geographic store regions.

```
CREATE TABLE LIST_BY_AREA
(
    STORE_NUMBER      int           NOT NULL,
    STORE_LOCATION    int           NOT NULL,
    ROLLUP_DATE       DATE          NOT NULL,
    STORE_RECEIPTS    DECIMAL(10,2) NOT NULL)
PARTITION BY LIST(STORE_LOCATION)
(
PARTITION P1 VALUES IN (1,2) ,
PARTITION P2 VALUES IN (3) ,
PARTITION P3 VALUES IN (4,5)
)
```

Partitioning Examples

- Sub or Composite Partitioning – Useful for further sub-dividing data into smaller subsets for faster access.

```
CREATE TABLE SUB_EXAMPLE
(
    CUSTOMER_NUMBER INT          NOT NULL,
    CUSTOMER_ORDER VARCHAR(50)   NOT NULL,
    CUSTOMER_ORDER_DATE DATETIME NOT NULL,
    CUSTOMER_SERVICE_REGION INT  NOT NULL)
PARTITION BY RANGE (YEAR (CUSTOMER_ORDER_DATE))
SUBPARTITION BY HASH (CUSTOMER_SERVICE_REGION)
(
    PARTITION P1 VALUES LESS THAN (2000) (SUBPARTITION S0, SUBPARTITION S1) ,
    PARTITION P2 VALUES LESS THAN (2003) (SUBPARTITION S2, SUBPARTITION S3) ,
    PARTITION P3 VALUES LESS THAN (2005) (SUBPARTITION S4, SUBPARTITION S5) ,
    PARTITION P4 VALUES LESS THAN MAXVALUE (SUBPARTITION S6, SUBPARTITION S7)
)
;
```

Partitioning Metadata

```
mysql> desc partitions;
```

Field	Type	Null	Key	Default	Extra
TABLE_CATALOG	varchar(512)	YES		NULL	
TABLE_SCHEMA	varchar(64)	NO			
TABLE_NAME	varchar(64)	NO			
PARTITION_NAME	varchar(64)	YES		NULL	
SUBPARTITION_NAME	varchar(64)	YES		NULL	
PARTITION_ORDINAL_POSITION	bigint(21)	YES		NULL	
SUBPARTITION_ORDINAL_POSITION	bigint(21)	YES		NULL	
PARTITION_METHOD	varchar(12)	YES		NULL	
SUBPARTITION_METHOD	varchar(5)	YES		NULL	
PARTITION_EXPRESSION	longtext	YES		NULL	
SUBPARTITION_EXPRESSION	longtext	YES		NULL	
PARTITION_DESCRIPTION	longtext	YES		NULL	
TABLE_ROWS	bigint(21)	NO		0	
AVG_ROW_LENGTH	bigint(21)	NO		0	
DATA_LENGTH	bigint(21)	NO		0	
MAX_DATA_LENGTH	bigint(21)	YES		NULL	
INDEX_LENGTH	bigint(21)	NO		0	
DATA_FREE	bigint(21)	NO		0	
CREATE_TIME	datetime	YES		NULL	
UPDATE_TIME	datetime	YES		NULL	
CHECK_TIME	datetime	YES		NULL	
CHECKSUM	bigint(21)	YES		NULL	
PARTITION_COMMENT	varchar(80)	NO			
NODEGROUP	bigint(21)	NO		0	
TABLESPACE_NAME	varchar(64)	NO			

Partitioning Metadata

```
mysql> insert into
test.RANGE_BY_DATE
VALUES (1, 'TEST', NOW());
```

```
mysql> insert into
test.RANGE BY DATE
VALUES (2, 'TEST', NOW());
```

```
mysql> insert into
test.RANGE BY DATE
VALUES (3, 'TEST', NOW());
```

```
mysql> select * from
partitions where
table_name =
'RANGE_BY_DATE'\G
```

```
***** 4. row *****
TABLE_CATALOG: NULL
TABLE_SCHEMA: test
TABLE_NAME: RANGE_BY_DATE
PARTITION_NAME: P4
SUBPARTITION_NAME: NULL
PARTITION_ORDINAL_POSITION: 4
SUBPARTITION_ORDINAL_POSITION: NULL
PARTITION_METHOD: RANGE
SUBPARTITION_METHOD: NULL
PARTITION_EXPRESSION: YEAR(CUSTOMER_ORDER_DATE)
SUBPARTITION_EXPRESSION: NULL
PARTITION_DESCRIPTION: MAXVALUE
TABLE_ROWS: 3
AVG_ROW_LENGTH: 24
DATA_LENGTH: 72
MAX_DATA_LENGTH: 281474976710655
INDEX_LENGTH: 1024
DATA_FREE: 0
CREATE_TIME: 2006-02-06 13:12:10
UPDATE_TIME: 2006-02-06 13:23:36
CHECK_TIME: NULL
CHECKSUM: NULL
PARTITION_COMMENT: default
NODEGROUP: 0
TABLESPACE_NAME: default
```

Partitioning and Performance

```
mysql> CREATE TABLE part_tab
-> ( c1 int ,c2 varchar(30) ,c3 date )
-> PARTITION BY RANGE (year(c3)) (PARTITION p0 VALUES LESS THAN (1995),
-> PARTITION p1 VALUES LESS THAN (1996) , PARTITION p2 VALUES LESS THAN (1997) ,
-> PARTITION p3 VALUES LESS THAN (1998) , PARTITION p4 VALUES LESS THAN (1999) ,
-> PARTITION p5 VALUES LESS THAN (2000) , PARTITION p6 VALUES LESS THAN (2001) ,
-> PARTITION p7 VALUES LESS THAN (2002) , PARTITION p8 VALUES LESS THAN (2003) ,
-> PARTITION p9 VALUES LESS THAN (2004) , PARTITION p10 VALUES LESS THAN (2010),
-> PARTITION p11 VALUES LESS THAN MAXVALUE );

mysql> create table no_part_tab (c1 int,c2 varchar(30),c3 date);
*** Load 8 million rows of data into each table ***

mysql> select count(*) from no_part_tab where c3 > date '1995-01-01' and c3 < date '1995-12-31';
+-----+
| count(*) |
+-----+
| 795181 |
+-----+
1 row in set (38.30 sec)

mysql> select count(*) from part_tab where c3 > date '1995-01-01' and c3 < date '1995-12-31';
+-----+
| count(*) |
+-----+
| 795181 |
+-----+
1 row in set (3.88 sec)
```



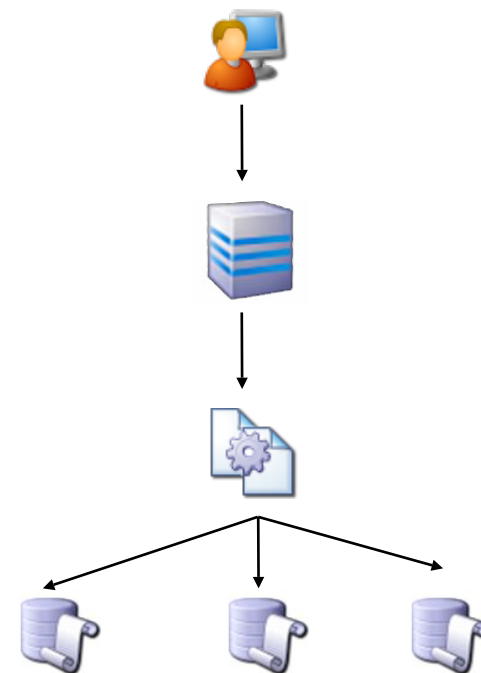
90% Response Time Reduction!



Business Intelligence/Data Warehousing

Full Text/Plug-in Enhancements

- Plug-In Parser with Examples
 - Common Word Handling
 - Case Selectivity
- Weighting Plug-In with Examples
 - Word Frequency
 - Term Proximity
- Boolean now the Default Mode
- Language Specific Plug-in Abilities
- SHOW PLUGIN
- UNINSTALL PLUGIN
- INSTALL PLUGIN
- --plugin_dir=path
- plugin.h
- See Chapter 26.2 of the 5.1 manual





Business Intelligence/Data Warehousing

XML Xpath Support

- Address parts in XML document
- Allows manipulation of
 - Strings
 - Booleans
 - Numbers
- Models XML document as tree of nodes



XPath Example

```
mysql> SELECT EXTRACTVALUE(doc, '/book/author/initial') FROM x;
```

```
+-----+
| EXTRACTVALUE(doc, '/book/author/initial') |
+-----+
| CJ |
| J |
+-----+
```

```
2 rows in set (0.01 sec)
```

```
mysql> SELECT extractValue(doc, '/book/child::*') FROM x;
```

```
+-----+
| extractValue(doc, '/book/child::*') |
+-----+
| A guide to the SQL standard |
| SQL:1999 |
+-----+
```

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

XML RSS Example: Extracting Titles

```
mysql> select ExtractValue(raw_xml, "/*/channel/title") from sites_log LIMIT 7;
```

```
+-----+
| ExtractValue(raw_xml, "/*/channel/ title") |
+-----+
| The Motley Fool |
| Nanodot: Nanodot: Nanotechnology News and Discussion of Emerging Technologies
|
| Slashdot: Science |
| Slashdot |
| Slashdot: Book Reviews |
| Slashdot: Features |
| Slashdot: Interviews |
+-----+
```

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

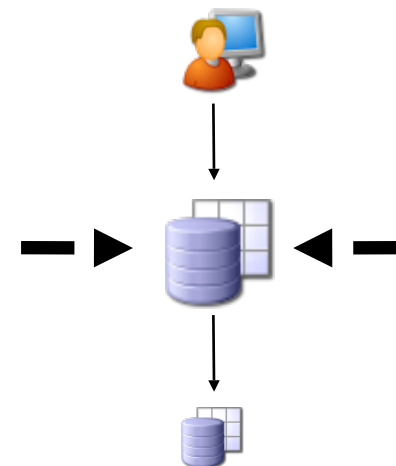
```
mysql> INSERT into titles select ExtractValue(raw_xml, "/*/channel/title") from
sites_log;
```



Business Intelligence/Data Warehousing

Archive Engine Enhancements

- Faster I/O operations
- Lower Memory requirements
- Autoincrement column support
 - Unique key support
 - Non-unique key support

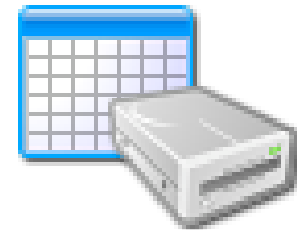




High Availability

MySQL Cluster Disk-Based Data

- Tables now can be designated to be disk-based
- Tablespaces used to store table disk data
- Tables can be either disk or main memory
- Indexes still main memory only



MySQL Cluster Disk-Based Data

```
CREATE TABLESPACE ts1
ADD DATAFILE 'datafile.dat'
USE LOGFILE GROUP lg1
INITIAL_SIZE 12M
ENGINE NDB;
```

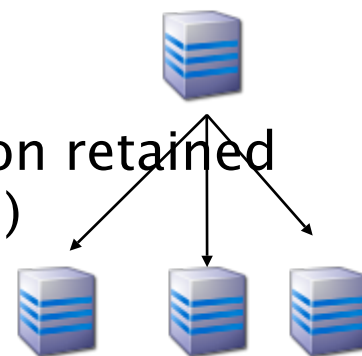
```
CREATE TABLE t1
(pk1 INT NOT NULL PRIMARY KEY,
 b INT NOT NULL,
 c INT NOT NULL
)
TABLESPACE ts1 STORAGE DISK
ENGINE NDB;
```



High Availability

Row-Based Replication

- New replication option – statement-based replication retained
- Handles all replication scenarios (deterministic, etc.)
- Safest form of replication
- Common to most other RDBMS's
- Statement-based approach still available
- Mixed mode available that does statement and row-based replication

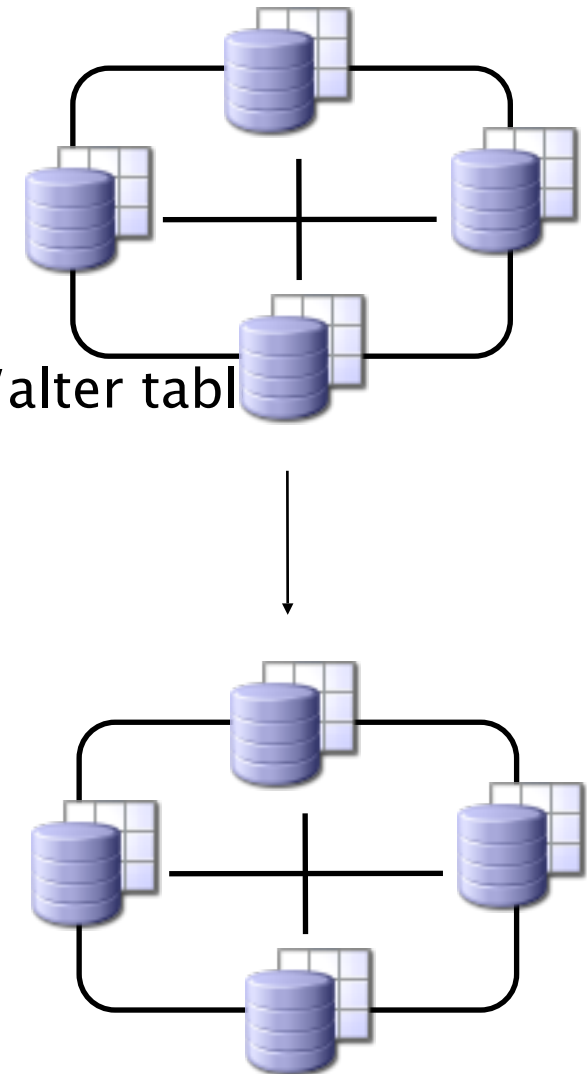




High Availability

MySQL Cluster Replication

- Replicate data from one MySQL Cluster to another
- Limitation - no cross-server logging of add/drop/alter table

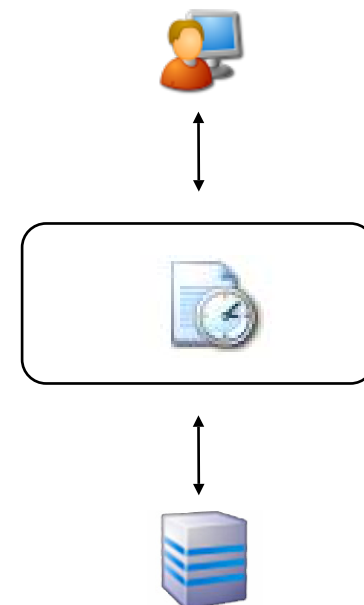




Easier Manageability

Task Scheduler

- New object – “Event”
- Create one-time or recurring tasks
- Execute SQL, block of SQL, or stored procedure
- Multiple threads used for task execution
- Can kill running task
- Only supported via row-based replication



Event Example

- DBA would like to schedule a reorg of InnoDB tables each Sunday night at 1AM.

```
DELIMITER //
CREATE EVENT OPTIMIZE_TABLES
ON SCHEDULE EVERY 1 WEEK
STARTS '2006-03-05 1:00:00'
ON COMPLETION PRESERVE
DO
BEGIN
OPTIMIZE TABLE test.table1;
OPTIMIZE TABLE test.table2;
END
//
```

Event Example

- DBA would like to schedule the creation of a view every day.

```
DELIMITER //
CREATE EVENT OPTIMIZE_TABLES
ON SCHEDULE EVERY 1 DAY
STARTS '2006-03-05 1:00:00'
ON COMPLETION PRESERVE
DO
BEGIN
OPTIMIZE TABLE test.table1;
OPTIMIZE TABLE test.table2;
END
//
```

Event Example

- DBA decides that as of 2007 the table is no longer needed.

```
DELIMITER //
CREATE EVENT OPTIMIZE_TABLES
ON SCHEDULE
AT TIMESTAMP '2007-01-01 0:01:00'
ON COMPLETION NOT PRESERVE
DO
BEGIN
DROP TABLE test.table2;
END
//
```

FAQ on Events!

- Can an event be missed?
- Can two events occur at the same time?
- And if daylight saving's time occurs?
- Is it possible to overload my system?
- Can I limit the number of events for a user?



Easier Manageability Transaction Support for Federated

- Handles transactions for InnoDB
- Does not work with XA
- Includes small performance improvements





Easier Manageability

CREATE SERVER

- Allows for the creation of connect types
- Creates one location for managing server information
- Available for memcache, http, AWS, Federated, ODBC-E

```
CREATE SERVER s
FOREIGN DATA WRAPPER mysql
OPTIONS (USER 'Remote', HOST '192.168.1.106',
DATABASE 'test');
```





Higher Performance

Faster Alter Table/Faster Add-Drop

Index

- Optimizations for MyISAM, InnoDB, and MEMORY
- Faster Add-Drop Index for MySQL Cluster only

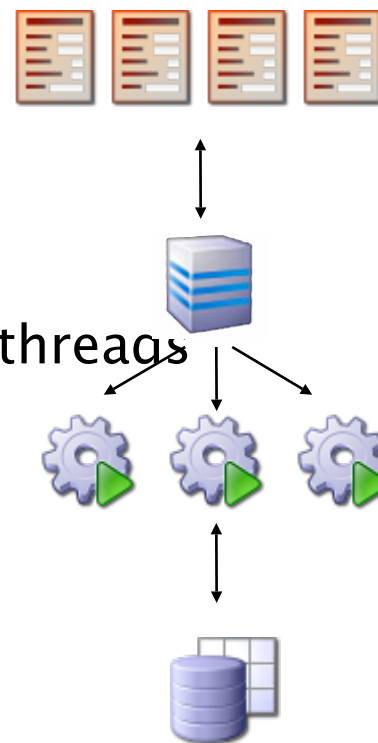




Higher Performance

Faster Data Import

- New option for mysqlimport utility
- --use-threads option allows DBA to specify multiple threads
- Particularly well-used by MySQL Cluster





Higher Performance

Better Problem User/SQL

Identification

- SHOW PROCESSLIST command now system table
- General and Slow Query Log now system tables
- Can easily query tables to find all or just inefficient SQL
- General/Slow Query Log use CSV engine and can be read by MS Excel



Processlist Table Metadata

```
mysql> desc processlist;
```

Field	Type	Null	Key	Default	Extra
ID	bigint(4)	NO		0	
USER	varchar(16)	NO			
HOST	varchar(64)	NO			
DB	varchar(64)	YES		NULL	
COMMAND	varchar(16)	NO			
TIME	bigint(4)	NO		0	
STATE	varchar(30)	YES		NULL	
INFO	varchar(100)	YES		NULL	

SQL Log Metadata

```
mysql> desc slow_log;
```

Field	Type	Null	Key	Default	Extra
start_time	timestamp	YES		CURRENT_TIMESTAMP	
user_host	mediumtext	NO			
query_time	time	NO			
lock_time	time	NO			
rows_sent	int(11)	NO			
rows_examined	int(11)	NO			
db	varchar(512)	YES		NULL	
last_insert_id	int(11)	YES		NULL	
insert_id	int(11)	YES		NULL	
server_id	int(11)	YES		NULL	
sql_text	mediumtext	NO			

```
mysql> desc general_log;
```

Field	Type	Null	Key	Default	Extra
event_time	timestamp	YES		CURRENT_TIMESTAMP	
user_host	mediumtext	YES		NULL	
thread_id	int(11)	YES		NULL	
server_id	int(11)	YES		NULL	
command_type	varchar(64)	YES		NULL	
argument	mediumtext	YES		NULL	

SQL Log Examples

```
mysql> select query_time, rows_examined, sql_text
-> from slow_log
-> order by query_time desc
-> limit 1\G

***** 1. row *****
query_time: 00:00:49
rows_examined: 9935
sql_text: SELECT c_custkey, c_name, SUM(l_extendedprice * (1 - l_discount)) AS
REVENUE,      c_acctbal, n_name, c_address, c_phone, c_comment FROM
dss_customer, dss_order, dss_lineitem, dss_nation WHERE c_custkey =
o_custkey      AND l_or derkey = o_orderkey      AND o_orderdate >=
'1993-10-01'      AND o_orderdate < '1994-1-01'      AND l_returnflag = 'R'
AND c_nationkey = n_nationkey GROUP BY c_custkey, c_name, c_acctbal,
c_phone,      n_name, c_address, c_comment ORDER BY REVENUE DESC

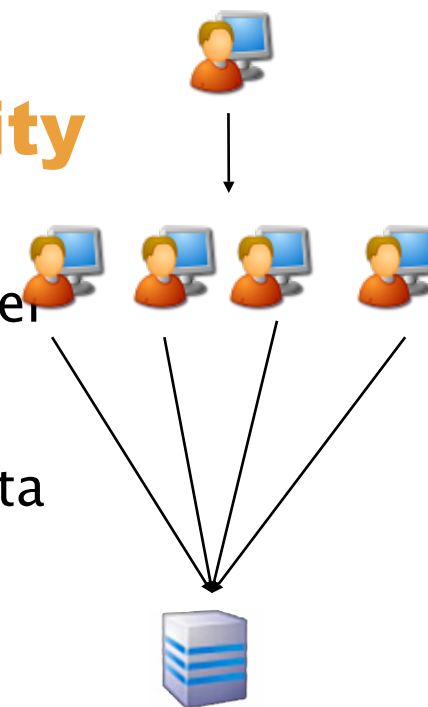
1 row in set (0.00 sec)
```



Higher Performance

New Performance/Load Testing Utility

- Named “mysqlslap”
- Simulates many concurrent connections to MySQL server
- Repetitively runs designated SQL load
- Can test multiple engines
- Performs creation of test schema and population of data



MySQLSlap Examples

```
mysqlslap --create="CREATE TABLE A (a int);INSERT INTO  
A (23)" --query="SELECT * FROM A" --concurrency=50 --  
iterations=200
```

```
mysqlslap --concurrency=5 --iterations=5 --  
query=query.sql  
--create=create.sql --delimiter=";"
```

SolidDB

- Transactional, ACID compliant Engine
- Optimistic Concurrency Control
- Configurable Durability

PBXT

- Transactional
- Designed for heavy INSERT usage.

memcache engine

- Designed to make caching architecture easier
- Can use as many memcached servers as are made available
- Beta today
- Multiple MySQL Servers can share the same data
- Information schema for monitoring memcache servers

AWS Engine

- Alpha Today
- Allows you to select and store information into Amazon's S3 service

HTTP Engine

- Generic web service engine.
- SELECT/INSERT/UPDATE/DELETE all mapped to generic web service architecture.
- Combined with new XML features you can parse RSS feeds
- Sister project, mod_methods, makes it easy to begin integration and testing with sites using Apache.

```
CREATE TABLE `d` (`a` varchar(125), b text, primary key(a)) ENGINE=HTTP  
DEFAULT CHARSET=latin1 CONNECTION="http://en.wikipedia.org/wiki/";
```