

# ~~Best Practices~~ Ideas for DBAs



**Sheeri Cabral**

Senior DB Admin/Architect, Mozilla

@sheeri [www.sheeri.com](http://www.sheeri.com)

OurSQL Podcast [www.oursql.com](http://www.oursql.com)

# MIRE



- Make It Really Easy
- Automate
- Document
- As for your brain.....



**Use your brain for CPU, not storage!**

**(use a request tracking system!)**

# Monitoring Basics



- Graph (Cacti)
- Alert (Nagios)
- Oracle Grid Control
- Check your checks



# Tradeoffs



- Tradeoffs always exist
- Think about them

# Most Commonly Given Advice



- Backup
- Restore
- What do you use backups for?

# In Case of Failure.....



- Restore from backup
- Master/Slave
- Master/Master
- Cluster

# DR? HA!



- Test your DR/HA plans
- What scenarios do they cover?
- What scenarios do they NOT cover?



# Ounces of Prevention



- Configuration vs. reality
- Possible memory usage
- Disk space / tablespace size

# Pounds of Cure



- Error Logs
- Slow Query Logs
- Query Review

mysqldumpslow



Count: 1 Time=12772.00s (12772s) Lock=0.00s (0s)

Rows=59493.0 (59493), prod[prod]@[192.168.217.53]

```
select * from properties_new prop left join nodes node on prop.node_id
= node.id where node.class_type = 'S' and qname like 'S' and node_id
not in ( select ancestor from history_links ) and node.id >= N order
by node_id
```

Count: 7 Time=160.57s (1124s) Lock=0.00s (0s) Rows=5567304.1

(38971129), test[test]@localhost SELECT /\*!N SQL\_NO\_CACHE \*/ \* FROM  
`properties\_new`

Count: 2 Time=51.00s (102s) Lock=0.00s (0s) Rows=0.0 (0),

prod[prod]@[192.168.217.53]

```
update nodes set vers=N, version_id=N, guid='S', creator='S',
owner='S', lastModifier='S', createDate=N, modDate=N, accessDate=N,
is_root=N, store_new_id=null, acl_id=null where id=N and vers=N
```

Count: 7 Time=32.00s (224s) Lock=0.00s (0s) Rows=698414.6 (4888902),

test[test]@localhost

```
SELECT /*!N SQL NO CACHE */ * FROM `nodes`
```



```
Reading slow log '/opt/mysql/mysql/data/mysql-slow.log'.  
25851 total queries, 291 unique.  
Sorting by 'at'.
```

\_\_\_ 001 \_\_\_\_\_

---

```
Count          : 2 (0%)  
Time           : 19677 s total, 9838 s avg, 6905 s to 12772 s max  
95% of Time    : 6905 s total, 6905 s avg, 6905 s to 6905 s max  
Lock Time      : 0 s total, 0 s avg, 0 s to 0 s max  
Rows sent      : 29746 avg, 0 to 59493 max  
Rows examined  : 75011 avg, 0 to 150023 max  
User           : user1@/192.168.217.53 (96%)  
Database       : Unknown
```

```
SELECT * FROM properties_new prop LEFT JOIN nodes node ON prop.node_id =  
node.id WHERE node.class_type = 'S' AND qname LIKE 'S' AND node_id NOT IN  
(S0) AND node.id >= N ORDER BY node_id;
```



002

---



```
Count          : 13 (0%)
Time           : 24042 s total, 1849 s avg, 5 s to 19839 s max
95% of Time    : 4203 s total, 350 s avg, 5 s to 2055 s max
Lock Time      : 0 s total, 0 s avg, 0 s to 0 s max
Rows sent      : 7692 avg, 0 to 100000 max
Rows examined  : 32573 avg, 0 to 423454 max
User           : prod@/196.168.217.53 (96%)
Database       : prod
```

```
SELECT * FROM properties_new prop LEFT JOIN nodes node ON prop.node_id =
node.id WHERE node.class_type = 'S' AND qname LIKE 'S' AND node_id NOT IN
(S0) AND node.id >= N ORDER BY node_id LIMIT N;
```

# Query Profile



- `pt-query-profiler`

Press <ENTER> when the external program is finished

```
+-----+
|                1 (476.2167 sec)                |
+-----+
```



Overall stats	Value
Total elapsed time	476.217
Questions	5431
COMMIT	1723
DELETE	8
DELETE MULTI	0
INSERT	8
INSERT SELECT	0
REPLACE	0
REPLACE SELECT	0
SELECT	2833
UPDATE	0
UPDATE MULTI	0
Data into server	884177
Data out of server	1013231

Table and index accesses	Value
Table locks acquired	3959
Table scans	26
Join	0
Index range scans	0
Join without check	0
Join with check	0
Rows sorted	17489
Range sorts	203
Merge passes	0
Table scans	21
Potential filesorts	11

# Query Profile



- pt-query-profiler
- SHOW STATUS before and after
- mysqltuner



MySQL 5.0.45-log

uptime 52 21:53:13

Tue Apr 15 14:37:23 2008



Key

---

Buffer used 2.16M of 16.00M %Used: 13.51  
Current 3.32M %Usage: 20.73  
Write hit 0.00%  
Read hit 100.00%

Questions

---

Total 84.87M 18.6/s  
DMS 48.18M 10.5/s %Total: 56.77  
Com\_ 36.54M 8.0/s 43.06  
COM\_QUIT 143.59k 0.0/s 0.17  
+Unknown 581 0.0/s 0.00  
Slow (4) 799 0.0/s 0.00 %DMS: 0.00 Log: ON  
DMS 48.18M 10.5/s 56.77  
SELECT 47.88M 10.5/s 56.42 99.37  
INSERT 165.96k 0.0/s 0.20 0.34  
DELETE 79.91k 0.0/s 0.09 0.17  
UPDATE 55.47k 0.0/s 0.07 0.12  
REPLACE 23 0.0/s 0.00 0.00  
Com\_ 36.54M 8.0/s 43.06  
commit 24.27M 5.3/s 28.59  
set\_option 11.57M 2.5/s 13.63  
rollback 543.18k 0.1/s 0.64



---

\_\_ SELECT and Sort

Scan	472.67k	0.1/s	%SELECT:	0.99
Range	9.49k	0.0/s		0.02
Full join	4.94k	0.0/s		0.01
Range check	0	0/s		0.00
Full rng join	146.47k	0.0/s		0.31
Sort scan	194.32k	0.0/s		
Sort range	2.80M	0.6/s		
Sort mrg pass	10.12k	0.0/s		

---

\_\_ Table Locks

Waited	0	0/s	%Total:	0.00
Immediate	1.12k	0.0/s		

---

\_\_ Tables

Open	422 of 1000	%Cache:	42.20
Opened	7.16k	0.0/s	

---

\_\_ Connections

Max used	42 of 100	%Max:	42.00
Total	143.59k	0.0/s	

---

\_\_ Created Temp

Disk table	9.32k	0.0/s		
Table	238.23k	0.1/s	Size:	32.0M
File	32	0.0/s		



---

### Threads

Running	2 of 42		
Cached	0 of 0	%Hit:	0
Created	143.59k	0.0/s	
Slow	0	0/s	

---

### Aborted

Clients	0	0/s	
Connects	0	0/s	

---

### Bytes

Sent	2.83G	620.1/s	
Received	2.81G	614.6/s	

---

### InnoDB Buffer Pool

Usage	1.00G of 1.00G	%Used:	100.00
Read hit	99.84%		
Pages			
Free	1	%Total:	0.00
Data	63.90k	97.50 %Drty:	0.00
Misc	1638	2.50	
Latched	0	0.00	
Reads	326.29M	71.4/s	
From file	515.08k	0.1/s	0.16
Ahead Rnd	16422	0.0/s	
Ahead Sql	30699	0.0/s	
Writes	21.52M	4.7/s	
Flushes	631.57k	0.1/s	
Wait Free	0	0/s	



InnoDB Lock

Waits	0	0/s
Current	0	
Time acquiring		
Total	0 ms	
Average	0 ms	
Max	0 ms	

InnoDB Data, Pages, Rows

---

Data

Reads	787.19k	0.2/s
Writes	1.35M	0.3/s
fsync	1.16M	0.3/s
Pending		
Reads	0	
Writes	0	
fsync	0	

Pages

Created	21.68k	0.0/s
Read	2.19M	0.5/s
Written	631.57k	0.1/s

Rows

Deleted	4.64M	1.0/s
Inserted	4.95M	1.1/s
Read	1.81G	396.5/s
Updated	32.92k	0.0/s



# Query Profile



- pt-query-profiler
- SHOW STATUS before and after
- mysqlreport
- Is every index being used?

# What Does This Query Do?



- `SELECT amount, created FROM payments where user='sheeri'`

# What Does This Query Do?



- `SELECT amount, created FROM payments where user='sheeri'`

**VS**

- `SELECT /* find payments for customer */ amount, created FROM payments where user='sheeri'`

# Data Profile



- PROCEDURE ANALYSE()
- What does each column, table do?
- What's in a name?



# Does It Make Sense?



- Can you “read” the data?
- Question “best practices”
- Then, make some!

# Schema Profile



- Start normal
- Denormalize if necessary
  - Descriptive foreign keys can prevent denormalization
- Stored procedures for developers

# Replication



- Be careful of TRIGGERS
  - And any DML from SP or UDF
- Sync often with pt-table-sync
- Handle duplicates carefully

# Maintenance



- Partition
- Archive
- Purge
- Static data



# Manage User Expectations



- Constant report refreshes
- Aggregate data once every hour or 15 minutes
- Split off processing:
  - customer/non-customer
  - internal/external

# Creative Suggestions – Forums



- COUNT(\*) for paging
- LIMIT 0,n+1
- Cache each page of forum
  - First page can have up to n+5 entries

# Creative Suggestions – Calculations



- Limited set of calculations?
  - Distance between zip codes in the US
- Calculate the same thing more than once?
- Calculating easy constants?

# Foreign Keys



- Application still has to handle problems



# Summary



# When I Walk In the Door



- alerting
- request tracking
- graphing
- documentation

# My Toolbox



- EXPLAIN
- Percona Toolkit
- mysqltuner
- MySQL Manual

# Also Useful



- mytop
- innotop





# Testing, Testing...is this thing on?

- Functional
- Load
- Start now!

# Consider



- master/master vs. read\_only slave
- Learn the foundations
- Semi-dynamic data

# Redundancy



- Make everything replaceable
  - Kickstart
  - Automated config management
- Including yourself!
  - Don't you want a vacation?

# Be a Good DBA



- Prove your work to yourself
- Be clear



# Questions? Comments?



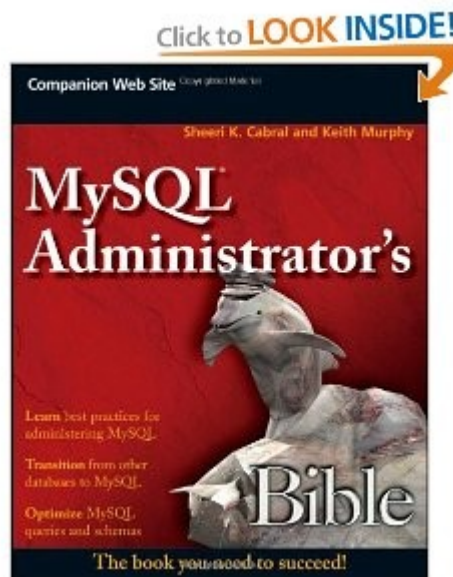
[scabral@mozilla.com](mailto:scabral@mozilla.com)

[@sheeri](mailto:@sheeri)

[www.oursqlcast.com](http://www.oursqlcast.com)

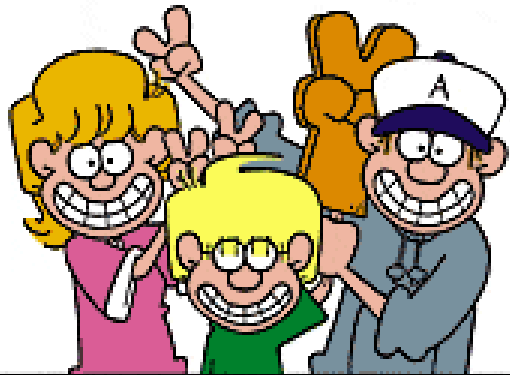
MySQL Administrator's Bible

- [tinyurl.com/mysqlbible](http://tinyurl.com/mysqlbible)



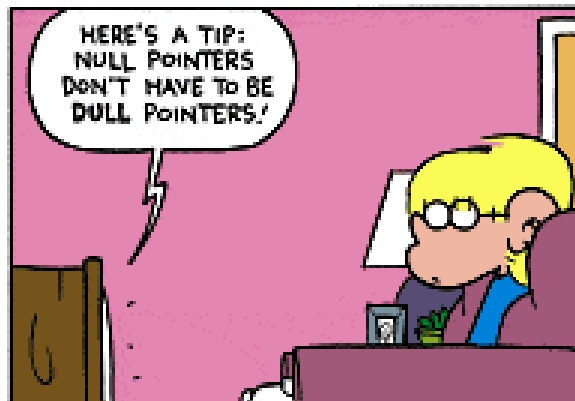
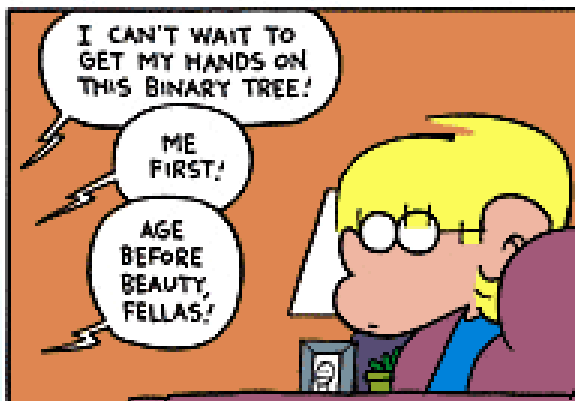
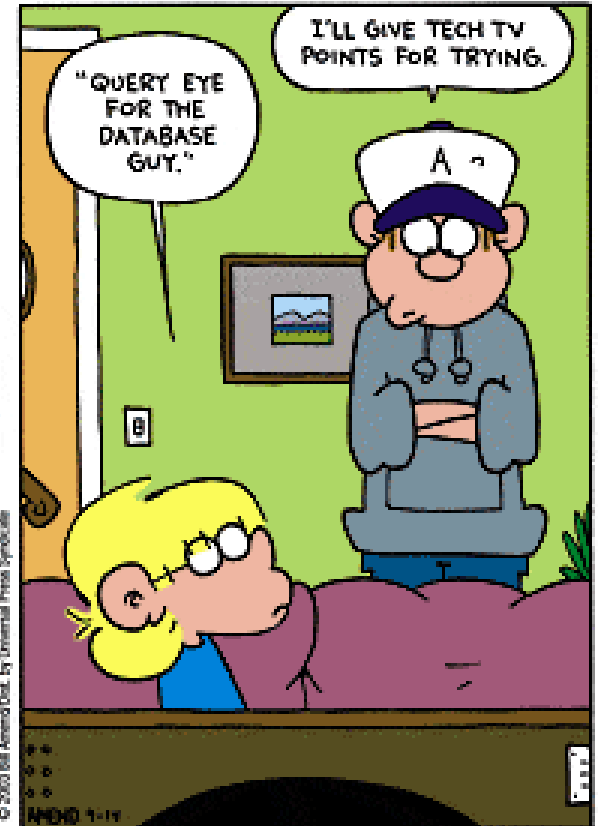
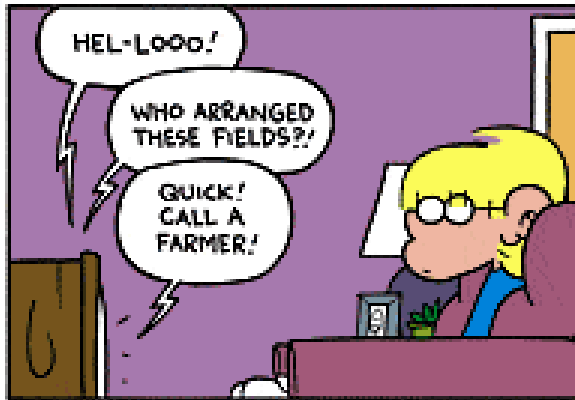
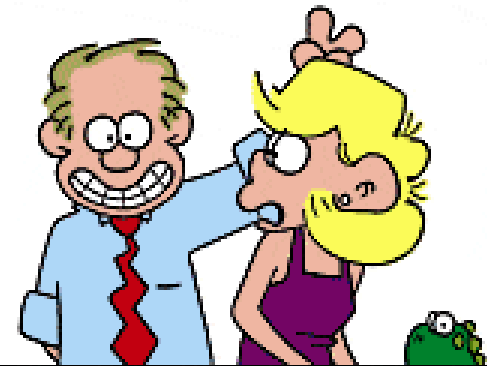
[kimtag.com/mysql](http://kimtag.com/mysql)

[planet.mysql.com](http://planet.mysql.com)



# FoxTrot

by Bill Amend



© 2003 Bill Amend/Dial, by Universal Press Syndicate

NDG 1-19

Panel

# Best Practices Ideas for DBAs



**Sheeri Cabral**

Senior DB Admin/Architect, Mozilla

@sheeri [www.sheeri.com](http://www.sheeri.com)

OurSQL Podcast [www.oursql.com](http://www.oursql.com)

## MIRE



- Make It Really Easy
- Automate
- Document
- As for your brain.....

### Checklists

documentation      yes it takes WORK to make it easy!  
automate  
graph!

force yourself to check status and alert you about errors. Don't rely on "get one e-mail per day per machine" -- what happens if one machine doesn't e-mail you, will you really see that?

Also, keep logs so you don't just log errors, you log good status, and can compare.

Documentation that's out of date is bad, but that's why you update it when it's wrong, or have others update it when it's wrong.



Use your brain for CPU, not storage!

(use a request tracking system!)

Especially one you can SEARCH and create documentation from.



# Monitoring Basics



- Graph (Cacti)
- Alert (Nagios)
- Oracle Grid Control
- Check your checks

Graph:

memory usage

cpu load (TALK ABOUT LOAD AVERAGE ON LINUX)

# of db threads

# of logged on users

TCP connections

disk space & usage

disk space & usage that = logs, data, etc.

replication lag?

other important metrics – maybe how long it took a page to load?

best to be able to correlate events, regular and otherwise – nightly spike = backup. That spike over there = weekly reports. That one over there = end of semester final crunch on a Friday at 5 pm.

Monitor:

all the above, and anything notable. Don't rely on customers/clients. Put reasonable thresholds. Do you really want to monitor every time there's a slow query? What about an entry into the error log?

Also, be notified if your check doesn't work, somehow. NULL=bad!

## Tradeoffs



- Tradeoffs always exist
- Think about them

Are you backing up for DR?

What about customer issue (ie, profile info)?

DROP TABLE recovery?

If you backup a slave are you keeping binary log positions of master?

Are you flushing your binary logs?

## Most Commonly Given Advice



- Backup
- Restore
- What do you use backups for?

Are you backing up for DR?

What about customer issue (ie, profile info)?

DROP TABLE recovery?

If you backup a slave are you keeping binary log positions of master?

Are you flushing your binary logs?

Again, think about the tradeoffs – you can mysqldump skipping the extended insert for easy grepping. But you have to lock tables.

Hot backup good, but costs \$\$ . mysqlhotcopy good but only MyISAM. Backups good but take time to restore.

## In Case of Failure.....



- Restore from backup
- Master/Slave
- Master/Master
- Cluster

Break glass!

Seriously though – you probably don't just have one disaster recovery scenario. If your development database dies, you restore from backup, and have downtime. If your production db dies, then what? What about testing, and all that test data?

Raise your hand if you work with databases

Raise your hand if you have a plan, and KNOW what it is for HA or DR or some kind of failure scenario.

Tell me what it is, if it's not on the slide

Raise your hand if you've practiced it somehow.

## DR? HA!



- Test your DR/HA plans
- What scenarios do they cover?
- What scenarios do they NOT cover?

Test your failover by propagating your slave to a master regularly – maybe when you have a schema change or otherwise would need a downtime.

Test your HA when you can – maybe in a downtime window, maybe not. But test, at least once every 6 months.

Nothing covers every scenario perfectly -  
TRADEOFFS!!!. RAID disks because of disks dying.  
2 machines because 1 machine could die. 2  
locations in case one loses power. What if someone  
drops the database? What if you have data centers  
in Boston and New York and power goes out for the  
whole Eastern Seaboard? That surely is OK to be  
without your service, yes?



## Ounces of Prevention



- Configuration vs. reality
- Possible memory usage
- Disk space / tablespace size

Check that the config matches what's in the db. You can write a script do to my\_print\_defaults and compare with SHOW VARIABLES and SHOW STATUS.

Possible memory usage: will you get a “sort aborted” or a crash due to possible total memory usage? Keep tabs on actual memory usage and compare.

How fast does your data grow?

Autoextend is great but do you really want a 100G innodb data file? Cap it with “max”. And of course monitor innodb free space. Make sure data and logs are sized appropriately! Also use innodb\_file\_per\_table if you're going to have large innodb tables.

## Pounds of Cure



- Error Logs
- Slow Query Logs
- Query Review

Look at these frequently.

Error logs – can show table corruption, network issues w/ dropped error connections, replication issues, etc.

slow query logs – mysqldump slow or mysqlsla!!!

Query review – like a code review. See if you can get a formal process, if not, you can always do it informally. “I noticed you have this query, if you're trying to do X then why not use Y? Or are you trying to do something else?”

Query review -- can be done any time, any where. Can use general log, application debug, or my personal favorite – mysql-proxy with adaptivity.

**THERE IS NO PERFECT QUERY** – this is a good place to think of tradeoff issues. What happens when the dataset gets very large? What indexes should you have now? How can you partition/purge?

```
mysqldumpslow

Count: 1  Time=12772.00s (12772s)  Lock=0.00s (0s)

  Rows=59493.0 (59493), prod[prod]@[192.168.217.53]

select * from properties_new prop left join nodes node on prop.node_id
= node.id where node.class_type = 'S' and qname like 'S' and node_id
not in ( select ancestor from history_links ) and node.id >= N order
by node_id

Count: 7  Time=160.57s (1124s)  Lock=0.00s (0s)  Rows=5567304.1
(38971129), test[test]@localhost  SELECT /*!N SQL_NO_CACHE */ * FROM
`properties_new`

Count: 2  Time=51.00s (102s)  Lock=0.00s (0s)  Rows=0.0 (0),
prod[prod]@[192.168.217.53]

update nodes set vers=N, version_id=N, guid='S', creator='S',
owner='S', lastModifier='S', createDate=N, modDate=N, accessDate=N,
is_root=N, store_new_id=null, acl_id=null where id=N and vers=N

Count: 7  Time=32.00s (224s)  Lock=0.00s (0s)  Rows=698414.6 (4888902),
test[test]@localhost

SELECT /*!N SQL NO CACHE */ * FROM `nodes`
```



Look at these frequently.


Error logs – can show table corruption, network issues w/ dropped error connections, replication issues, etc.

slow query logs – mysqldump slow or mysqlsla!!!

Query review – like a code review. See if you can get a formal process, if not, you can always do it informally. “I noticed you have this query, if you're trying to do X then why not use Y? Or are you trying to do something else?”

Query review -- can be done any time, any where. Can use general log, application debug, or my personal favorite – mysql-proxy with adaptivity.

THERE IS NO PERFECT QUERY – this is a good place to think of tradeoff issues. What happens when the dataset gets very large? What indexes should you have now? How can you partition/purge?



```
Reading slow log '/opt/mysql/mysql/data/mysql-slow.log'.
25851 total queries, 291 unique.
Sorting by 'at'.

__ 001 _____

Count          : 2 (0%)
Time           : 19677 s total, 9838 s avg, 6905 s to 12772 s max
95% of Time    : 6905 s total, 6905 s avg, 6905 s to 6905 s max
Lock Time      : 0 s total, 0 s avg, 0 s to 0 s max
Rows sent      : 29746 avg, 0 to 59493 max
Rows examined  : 75011 avg, 0 to 150023 max
User           : user1@/192.168.217.53 (96%)
Database       : Unknown

SELECT * FROM properties_new prop LEFT JOIN nodes node ON prop.node_id =
node.id WHERE node.class_type = 'S' AND qname LIKE 'S' AND node_id NOT IN
(S0) AND node.id >= N ORDER BY node_id;
```

Look at these frequently.

Error logs – can show table corruption, network issues w/ dropped error connections, replication issues, etc.

slow query logs – mysqldump slow or mysqlsla!!!


Query review – like a code review. See if you can get a formal process, if not, you can always do it informally. “I noticed you have this query, if you're trying to do X then why not use Y? Or are you trying to do something else?”

Query review -- can be done any time, any where. Can use general log, application debug, or my personal favorite – mysql-proxy with adaptivity.

THERE IS NO PERFECT QUERY – this is a good place to think of tradeoff issues. What happens when the dataset gets very large? What indexes should you have now? How can you partition/purge?

```
__ 002 _____
Count          : 13 (0%)
Time           : 24042 s total, 1849 s avg, 5 s to 19839 s max
95% of Time    : 4203 s total, 350 s avg, 5 s to 2055 s max
Lock Time      : 0 s total, 0 s avg, 0 s to 0 s max
Rows sent      : 7692 avg, 0 to 100000 max
Rows examined  : 32573 avg, 0 to 423454 max
User           : prod@196.168.217.53 (96%)
Database       : prod

SELECT * FROM properties_new prop LEFT JOIN nodes node ON prop.node_id =
node.id WHERE node.class_type = 'S' AND qname LIKE 'S' AND node_id NOT IN
(S0) AND node.id >= N ORDER BY node_id LIMIT N;
```



Look at these frequently.

Error logs – can show table corruption, network issues w/ dropped error connections, replication issues, etc.

slow query logs – mysqldump slow or mysqlsla!!!

Query review – like a code review. See if you can get a formal process, if not, you can always do it informally. “I noticed you have this query, if you're trying to do X then why not use Y? Or are you trying to do something else?”

Query review -- can be done any time, any where. Can use general log, application debug, or my personal favorite – mysql-proxy with adaptivity.

**THERE IS NO PERFECT QUERY** – this is a good place to think of tradeoff issues. What happens when the dataset gets very large? What indexes should you have now? How can you partition/purge?



## Query Profile



- pt-query-profiler

Again, tradeoffs. Think about rewriting queries to use existing indexes, or changing the indexes to cover more ground – not just adding one.



## Query Profile



- pt-query-profiler
- SHOW STATUS before and after
- mysqltuner

Again, tradeoffs. Think about rewriting queries to use existing indexes, or changing the indexes to cover more ground – not just adding one.

MySQL 5.0.45-log      uptime 52 21:53:13      Tue Apr 15 14:37:23 2008




---

Key  
Buffer used 2.16M of 16.00M %Used: 13.51  
Current 3.32M %Usage: 20.73  
Write hit 0.00%  
Read hit 100.00%

---

Questions  
Total 84.87M 18.6/s  
DMS 48.18M 10.5/s %Total: 56.77  
Com\_ 36.54M 8.0/s 43.06  
COM\_QUIT 143.59k 0.0/s 0.17  
+Unknown 581 0.0/s 0.00  
Slow (4) 799 0.0/s 0.00 %DMS: 0.00 Log: ON  
DMS 48.18M 10.5/s 56.77  
SELECT 47.88M 10.5/s 56.42 99.37  
INSERT 165.96k 0.0/s 0.20 0.34  
DELETE 79.91k 0.0/s 0.09 0.17  
UPDATE 55.47k 0.0/s 0.07 0.12  
REPLACE 23 0.0/s 0.00 0.00  
Com\_ 36.54M 8.0/s 43.06  
commit 24.27M 5.3/s 28.59  
set\_option 11.57M 2.5/s 13.63  
rollback 543.18k 0.1/s 0.64


Again, tradeoffs. Think about rewriting queries to use existing indexes, or changing the indexes to cover more ground – not just adding one.



__ SELECT and Sort			
Scan	472.67k	0.1/s	%SELECT: 0.99
Range	9.49k	0.0/s	0.02
Full join	4.94k	0.0/s	0.01
Range check	0	0/s	0.00
Full rng join	146.47k	0.0/s	0.31
Sort scan	194.32k	0.0/s	
Sort range	2.80M	0.6/s	
Sort mrg pass	10.12k	0.0/s	
__ Table Locks			
Waited	0	0/s	%Total: 0.00
Immediate	1.12k	0.0/s	
__ Tables			
Open	422 of 1000		%Cache: 42.20
Opened	7.16k	0.0/s	
__ Connections			
Max used	42 of 100		%Max: 42.00
Total	143.59k	0.0/s	
__ Created Temp			
Disk table	9.32k	0.0/s	
Table	238.23k	0.1/s	Size: 32.0M
File	32	0.0/s	


Again, tradeoffs. Think about rewriting queries to use existing indexes, or changing the indexes to cover more ground – not just adding one.





Threads			
Running	2 of 42		
Cached	0 of 0	%Hit:	0
Created	143.59k	0.0/s	
Slow	0	0/s	
Aborted			
Clients	0	0/s	
Connects	0	0/s	
Bytes			
Sent	2.83G	620.1/s	
Received	2.81G	614.6/s	
InnoDB Buffer Pool			
Usage	1.00G of 1.00G	%Used:	100.00
Read hit	99.84%		
Pages			
Free	1	%Total:	0.00
Data	63.90k	97.50 %Drty:	0.00
Misc	1638	2.50	
Latched	0	0.00	
Reads	326.29M	71.4/s	
From file	515.08k	0.1/s	0.16
Ahead Rnd	16422	0.0/s	
Ahead Sql	30699	0.0/s	
Writes	21.52M	4.7/s	
Flushes	631.57k	0.1/s	
Wait Free	0	0/s	

Again, tradeoffs. Think about rewriting queries to use existing indexes, or changing the indexes to cover more ground – not just adding one.



InnoDB Lock		
Waits	0	0/s
Current	0	
Time acquiring		
Total	0 ms	
Average	0 ms	
Max	0 ms	
InnoDB Data, Pages, Rows		
Data		
Reads	787.19k	0.2/s
Writes	1.35M	0.3/s
fsync	1.16M	0.3/s
Pending		
Reads	0	
Writes	0	
fsync	0	
Pages		
Created	21.68k	0.0/s
Read	2.19M	0.5/s
Written	631.57k	0.1/s
Rows		
Deleted	4.64M	1.0/s
Inserted	4.95M	1.1/s
Read	1.81G	396.5/s
Updated	32.92k	0.0/s

Again, tradeoffs. Think about rewriting queries to use existing indexes, or changing the indexes to cover more ground – not just adding one.

## Query Profile



- pt-query-profiler
- SHOW STATUS before and after
- mysqlreport
- Is every index being used?

Again, tradeoffs. Think about rewriting queries to use existing indexes, or changing the indexes to cover more ground – not just adding one.

## What Does This Query Do?



- `SELECT amount, created FROM payments where user='sheeri'`

## What Does This Query Do?



- `SELECT amount, created FROM payments where user='sheeri'`

**VS**

- `SELECT /* find payments for customer */ amount, created FROM payments where user='sheeri'`



## Data Profile



- PROCEDURE ANALYSE()
- What does each column, table do?
- What's in a name?

Could you be using better data types? When do you need BIGINT? INT instead of varchar? Look out for things like “email” which could use a “Reverse e-mail” column.

Is each column or table used? I worked at a company that had a user profile, and their user table was 80 columns long and growing. They had 2 columns for “user1” and “user2” for future growth, but whenever they actually needed a new column they'd ALTER the table to add a column, and have a downtime.

Also, does your field have a name like “Create date” when it's really a timestamp, not a date? A good rule – don't put a data type in the descriptor. Similarly, is it better to have different or the same names for columns? Dot notation vs. ease of “USING” clauses.

Does each table have a timestamp column and a created column and a row number? Why? With an 80 column table the timestamp is rather useless. But other times its useful.

Same with surrogate keys. They're useful but you don't ALWAYS need them. Ie, username :)

## Does It Make Sense?



- Can you “read” the data?
- Question “best practices”
- Then, make some!

Is the data readable? Descriptive foreign keys are good for 2 reasons – 1, less joining and 2, you don't have to worry about what “member\_type=1” means. Is it human readable or just machine readable? Consider things like unix timestamp versus real time, IP addresses. Duplicate the data if you want the best of both worlds!

Does each table have a timestamp column and a created column and a row number? Why? With an 80 column table the timestamp is rather useless. But other times its useful.

Same with surrogate keys. They're useful but you don't ALWAYS need them. Ie, username :)

Naming your columns, as we talked about before. CamelCase vs. under\_score may not matter, but do you really want to have to run SHOW CREATE TABLE because you forgot the difference?

There is no perfect solution, so again remember to think of the tradeoffs!

## Schema Profile



- Start normal
- Denormalize if necessary
  - Descriptive foreign keys can prevent denormalization
- Stored procedures for developers

A lot of times developers don't want to join tables, and say “why can't we just denormalize this right now?” Denormalization is a fact in data warehouses and reporting, but not in OLTP!

Often times a descriptive foreign key works well – ie, status of an account is “paid” rather than “1”. Be stingy with your tradeoffs here!

If there are some difficult queries and your developers are happier using an ORM or simple queries, make it easy for them – write a stored procedure, so you control the query!

# Replication



- Be careful of TRIGGERS
  - And any DML from SP or UDF
- Sync often with pt-table-sync
- Handle duplicates carefully

## Maintenance



- Partition
- Archive
- Purge
- Static data

At some point you will have a table that's really large. How could you partition it?

If you need to keep around legacy data for reporting, it doesn't need to be all on one server. For things like “orders in 2006” you can have a server that has the 2006 information. If you need to do some combined reporting you can have current information replicated to the reporting server. Or, aggregate what you need and put that on the reporting server.

Static data like lookup tables should always be in a different db so you can move the info if necessary, and for easier backup – you don't need to backup your list of what streets are in what city every single night. You can do it once a month.



## Manage User Expectations



- Constant report refreshes
- Aggregate data once every hour or 15 minutes
- Split off processing:
  - customer/non-customer
  - internal/external

Even though management is your customer, and even though you may have customers who use your db for their end-user customer, don't let reporting get in the way of end-user functionality!

I've done this by having caveats in the programming code that the reports can't be run during the busy time in the code. Could also use proxy to do that, or to send it to a particular server when it's a reporting query.

My bank isn't always up to the second and it has my MONEY in it! Can't users wait for the total # of their posts to be calculated every hour?

Or use TRIGGERS, but re-sync!

## Creative Suggestions – Forums



- COUNT(\*) for paging
- LIMIT 0,n+1
- Cache each page of forum
  - First page can have up to n+5 entries

Standard count(\*) for paging does it backwards. The oldest forum post will always be the oldest forum post.

For the initial page, limit 0,21 if you're paging by 20's. Then, you can have the correct arrow, and while the first 20 posts are being read you can do your select count(\*) and store that as a variable.

Another way to do it is that the last 20 posts are always the last page, posts 40-21 are always the 2<sup>nd</sup> to last page, etc. It would look odd to have only 1 or 2 posts on a page, so have the first page able to have up to 25 posts.

## Creative Suggestions – Calculations



- Limited set of calculations?
  - Distance between zip codes in the US
- Calculate the same thing more than once?
- Calculating easy constants?

Calculating easy constants – are you using “CURRENT\_DATE()” instead of just using your app to put the number in there? Especially if you're comparing things.....

# Foreign Keys



- Application still has to handle problems

# Summary





## When I Walk In the Door



- alerting
- request tracking
- graphing
- documentation

If it's not there, I put it in place. If there isn't a “company approved” solution I just say “This is for the db team only” 'cause we have jurisdiction....every single time it's been so useful others have taken to using it.

# My Toolbox



- EXPLAIN
- Percona Toolkit
- mysqltuner
- MySQL Manual

## Also Useful



- mytop
- innotop

## Testing, Testing...is this thing on?



- Functional
- Load
- Start now!

Know what you're looking for, too, because everything has a breaking point, and you need to know what's sensible.

How long does it take to make a user? 1,000 users in 1 minute? What is current site usage and the user perception? And of course what the graphs show.

## Consider



- master/master vs. read\_only slave
- Learn the foundations
- Semi-dynamic data

If you have a read\_only slave, failover is harder. Why not use master/master replication, and just point to one server? Helps with schema changes, etc.

Knowing the foundation – if you know what a B-tree is, and that that's how MySQL stores its data and non-memory indexes, you'll understand why queries work the way they do, and why range searches are good on B-trees.

Semi-dynamic data – when it comes to “store in a config file so the app can read it” or “store in the db so it's easy to change” why not do both? Have the db propagate to the file.



# Redundancy



- Make everything replaceable
  - Kickstart
  - Automated config management
- Including yourself!
  - Don't you want a vacation?

If you document what you do, and be transparent, you'll find that people appreciate it more. Whether or not they understand what you write, it's there, and if you need a 2<sup>nd</sup> set of eyes it becomes much easier.

Documentation isn't a project, a one-off thing. There might be a large effort to start. But there's constant updating needed, and you also need to train people to use it.

## Be a Good DBA



- Prove your work to yourself
- Be clear

le, confirm it as an underling would. Don't just GRANT permissions, try to login as that user!

Also, if you're asked to create a user, don't just say "done". Write back and say "I've created the user foo that has access to the database bar, you should be able to login from baz now."

# Questions? Comments?



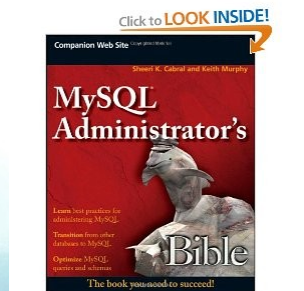
[scabral@mozilla.com](mailto:scabral@mozilla.com)

[@sheeri](#)

[www.oursqlcast.com](http://www.oursqlcast.com)

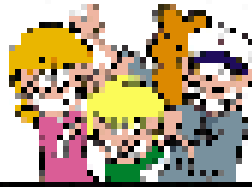
MySQL Administrator's Bible

- [tinyurl.com/mysqlbible](http://tinyurl.com/mysqlbible)



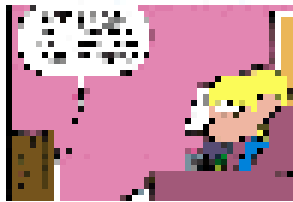
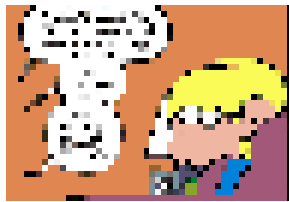
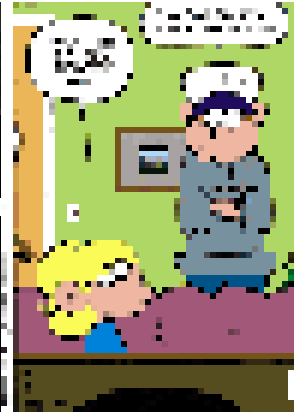
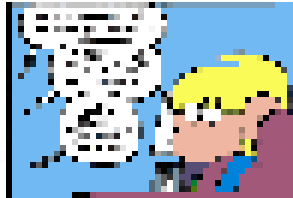
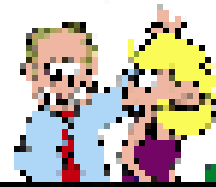
[kimtag.com/mysql](http://kimtag.com/mysql)

[planet.mysql.com](http://planet.mysql.com)



# FoxTrot

by Bill Amend



1