PALOMINODB

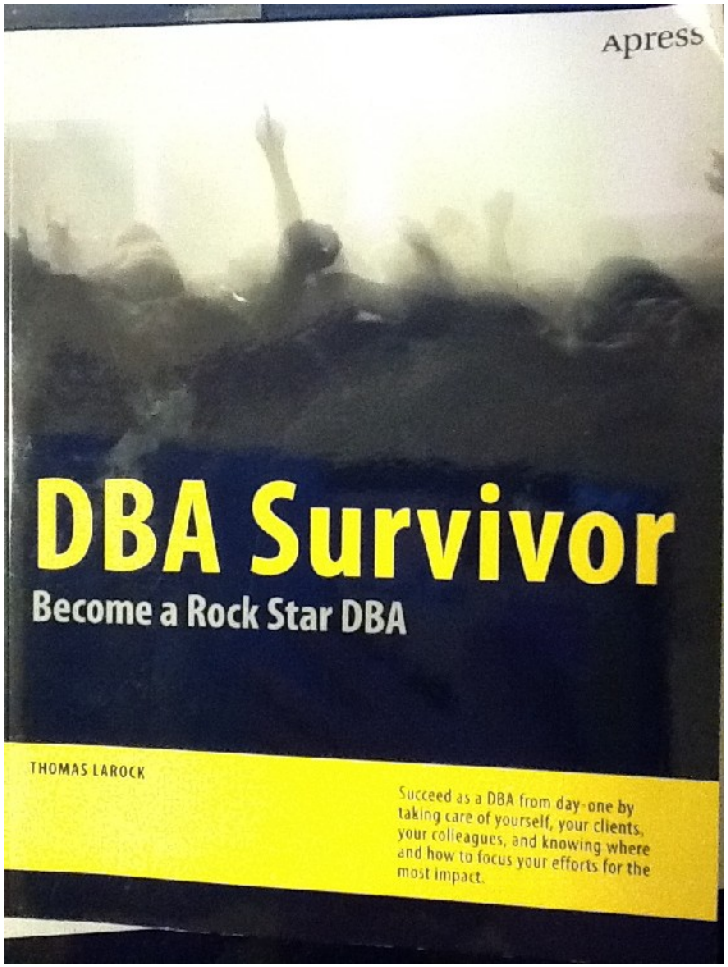
Becoming a Rock Star MySQL DBA

Presented by:

Sheeri K. Cabral - @sheeri

Senior DBA & Community Liasion

www.palominodb.com


apress
DBA Survivor
Become a Rock Star DBA
THOMAS LAROCK
Succeed as a DBA from day-one by taking care of yourself, your clients, your colleagues, and knowing where and how to focus your efforts for the most impact.

Most important...

“Many of the fundamental lessons for a new database administrator . . . are about how you interact with other people in your enterprise's IT environment.”

-Kevin Kline

Day 1

- Find your stuff
- Find your customer's stuff
- Start making an action plan

Your stuff

- Server list
  - Most important db's
- Backups
- Test restore(s)
- Upcoming projects / deliverables
- Baselines
- Recovery plan

"These tasks will all take longer than a day to perform, but do not delay in getting the checklist started."

Getting a Server List

- Existing docs / Asking
- Port scanning
- Slaves
- Not responsible for

Backups

- Which servers?
- What?
- How?
- When?
- How big?

Verify a Restore

- Verify for each backup
- Same-server restore: be careful!
- Repeat regularly
- mk-table-checksum
  - “continual replication sync”
  - http://www.pythian.com/news/4397/online-verification-that-master-and-slaves-are-in-sync/

Customer List

- Who uses what?
- For communication purposes
- Find out apps and time used

“A development server is a production server to a developer”

Upcoming Projects / Deliverables

- Surprises are bad
- Starting point
- Is anything going to be decomissioned?

You are responsible for your review

Baseline

- Document your starting point
- Server physical config
  - Disks/RAID/SAN, CPU, RAM
  - Where data/logs are stored
- Goal: Track changes over time
  - http://code.google.com/p/mysql-cacti-templates
  - http://www.pythian.com/news/mysql-plugin-for-oracle-grid-control

Recovery Plan

- What disasters?
- How far back?
  - Disks/RAID/SAN, CPU, RAM
  - Where data/logs are stored
- Goal: Track changes over time
  - http://code.google.com/p/mysql-cacti-templates
  - http://www.pythian.com/news/mysql-plugin-for-oracle-grid-control

Now What?

- Prioritize
  - Restores are most important
- “What are your roadblocks?”
- Know the SA team

Alerts

- Use role accounts
- Is it important?
  - If not, why are you getting it?
  - e.g. CPU load

Work Together

- Spend time with coworkers
  - “Grab a cup of coffee”
  - Take a walk
- Even if you are not an extrovert
  - Just another [soft] skill to work on

Take Care of Yourself

- Do not spend every second with coworkers
- Have a life outside work
- Take care of yourself physically
  - When you feel your best, you do your best

Nobody Gets an A for Effort

“Being able to fix things does not make you a senior DBA. You become a senior DBA by making certain that things do not break in the first place.”

The 2-sided DBA

Right

- Behind-the-scenes
- “What do you do?”
- Fewer problems

Right Now

- Very visible
- “Go see Sheeri”
- Very vocal

Solve Root Causes

- Explain optimizations to devs
- Query reviews (mk-query-digest)
  - Slides - http://bit.ly/my10qreview
  - youtube.com/watch?v=IGkBzQuRS5E
- Review server/db configs regularly

Know Underlying Technology

- Disks – RAID, SAN, volumes, partitions
  - Spinning disks vs. SSDs
- How MySQL uses CPU
- How different caches work
  - Filesystem
  - Innodb

Benchmark, Benchmark, Benchmark

HA options

- HA is not DR
  - Great uptime but no recovery plan
  - Great recovery plan but bad uptime
- HA is not DR

HA options

- Server-level clustering
- Replication architectures
- MySQL Cluster
- Know underlying technology!!

Where to learn?

- Books
- Blogs - http://planet.mysql.com
- Podcasts (OurSQL, Meet the MySQL Experts)
- Conferences / Training / User Groups
- http://kimtag.com/mysql

DR Planning

- Should be “disasters/recoveries planning”
- Get ridiculous
- Monitor to prevent disasters
- Test it!

DBA

- Database Administrator
- Default Blame Acceptor
- Don't Bother Asking

Getting Others to Embrace Change

- Organizational need
- Individual/end user need
- Desired outcome
- Timeline

Standards / Processes

- People will not like to be limited
- Try to be helpful
- Desired outcome
- Timeline

“We should put images in the database”

Communicate!

- Everybody's needs are important
  - Especially to them!
- Give estimates, update them regularly
  - Once daily or more
- Be honest
- Be transparent

Documenting what and how you do your job is more job security than not documenting it.

Have SLAs and follow them

- Especially times
- Manage expectations
- Proactive vs. reactive - “How much longer?”

Sometimes you need to say, “I can spend time answering your questions, or I can spend time fixing this stuff. How about I update you in 15 minutes?”

Being a DBA is like a buffet; sometimes you just have to say no or you'll be stuffed.

Documentation = Communication

- Not the only means
- But an important one
- Have meetings where you read the plan aloud and go through steps if need be
- Not everyone reads e-mail!

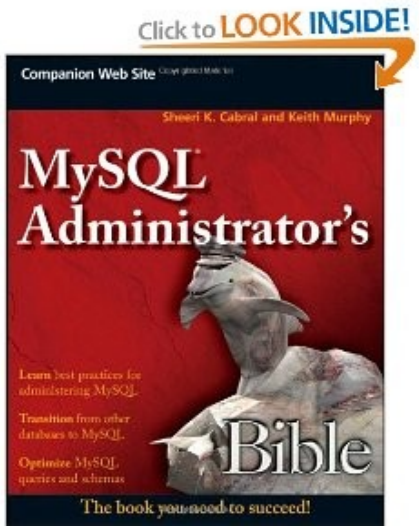
Outsourcing

- If you need an expert
- If you need another warm body
- If you need a 2nd set of eyes
- Documentation helps set metrics for outsourcing

Questions? Comments?

OurSQL podcast

MySQL Administrator's Bible



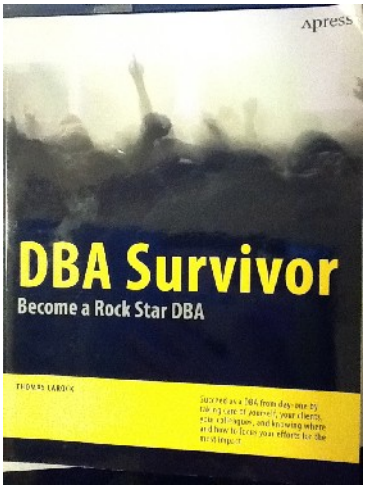
kimtag.com/mysql

planetmysql.com

PALOMINODB

Becoming a Rock Star MySQL DBA

Presented by:
Sheeri K. Cabral - @sheeri
Senior DBA & Community Liasion
www.palominodb.com



About Microsoft SQL Server, but concepts are easily transferred.

Most important...

“Many of the fundamental lessons for a new database administrator . . . are about how you interact with other people in your enterprise's IT environment.”

-Kevin Kline

Day 1

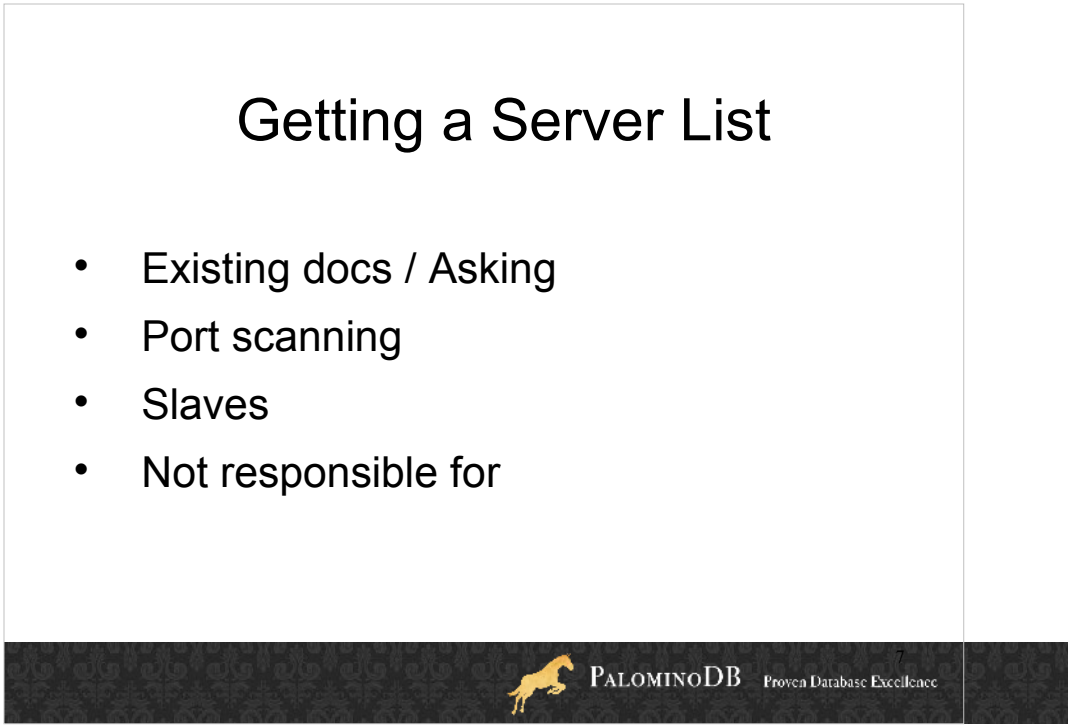
- Find your stuff
- Find your customer's stuff
- Start making an action plan

Your stuff

- Server list
  - Most important db's
- Backups
- Test restore(s)
- Upcoming projects / deliverables
- Baselines
- Recovery plan

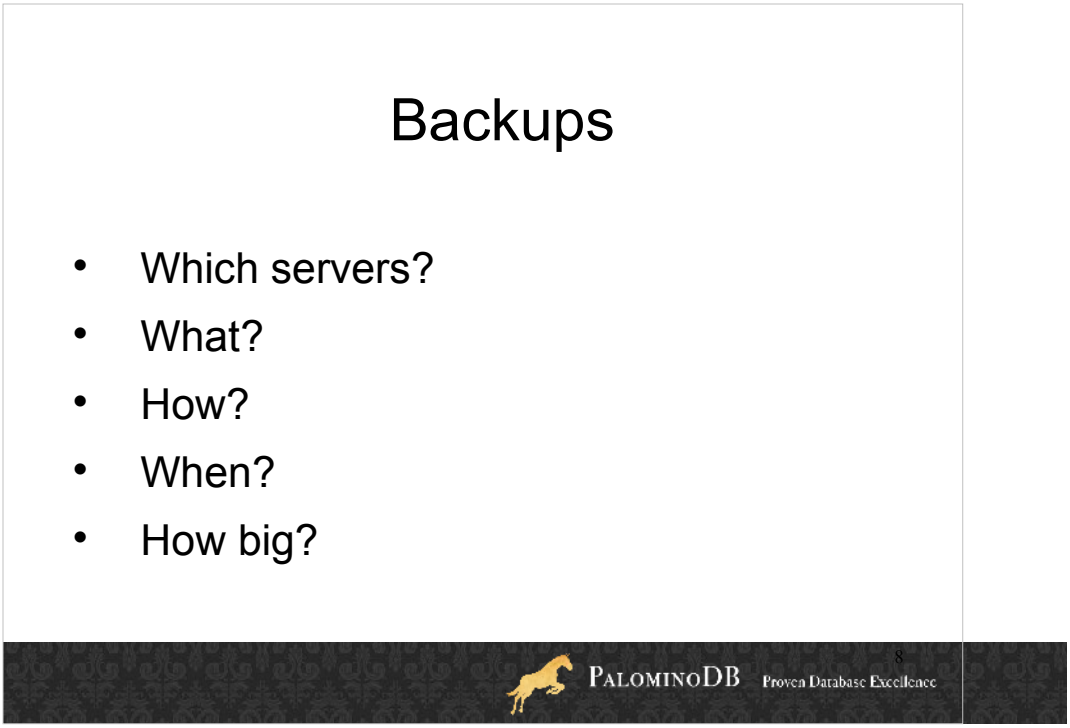
Recovery plan is not a backup/restore plan. It's a “what happens if disaster strikes” plan, including times to recover.

"These tasks will all take longer than a day to perform, but do not delay in getting the checklist started."



Look at the processlist to find slaves.

Are there db's you're not responsible for? Such as individual dev environments, not a shared staging one? Or something maintained by a vendor? Maybe a different type of DB, like Oracle or NoSQL? It's important to know about these, because if something dies, you want to know when it's NOT your responsibility too.



What is being backed up – and what needs to be backed up?

Ask yourself and your developers and sysadmins what would happen if each environment went away. Often you realize you need to backup the reporting environment, or you find out that the dev environment is being used for other purposes!

Are the backups logical or physical? Are they a complete backup? Consistent to a point-in-time? When are full backups done, and when are incremental backups done?

How frequently are the backups being done? Recently we restored a backup to a point in time for a client, but it took 4 hours because the disaster was on a Friday and they took full backups on Sundays only.

Verify a Restore

- Verify for each backup
- Same-server restore: be careful!
- Repeat regularly
- mk-table-checksum
  - "continual replication sync"
  - http://www.pythian.com/news/4397/online-verification-that-master-and-slaves-are-in-sync/

Verifying a restore does not have to be the only benefit of the exercise. Whenever you make a new slave, you are verifying a restore. When you restore a production environment to a staging system, you're testing the restore.

Verify not just that the db runs and is a similar size and replication doesn't break; verify by using checksums is great.

How long does it take? Copy files, decompress, apply logs with xtrabackup, import SQL, etc.

If you need to restart a db, you want to make sure the President isn't depending on the server to be up during that time!

Now you're starting to communicate with people. They now know that indeed you're in charge of those dbs and that you're there to help. This can be good and bad :D

You're also finding out what's important. If there's a non-production database that lots of folks depend on, then it's pretty important. e.g. ticketing system.

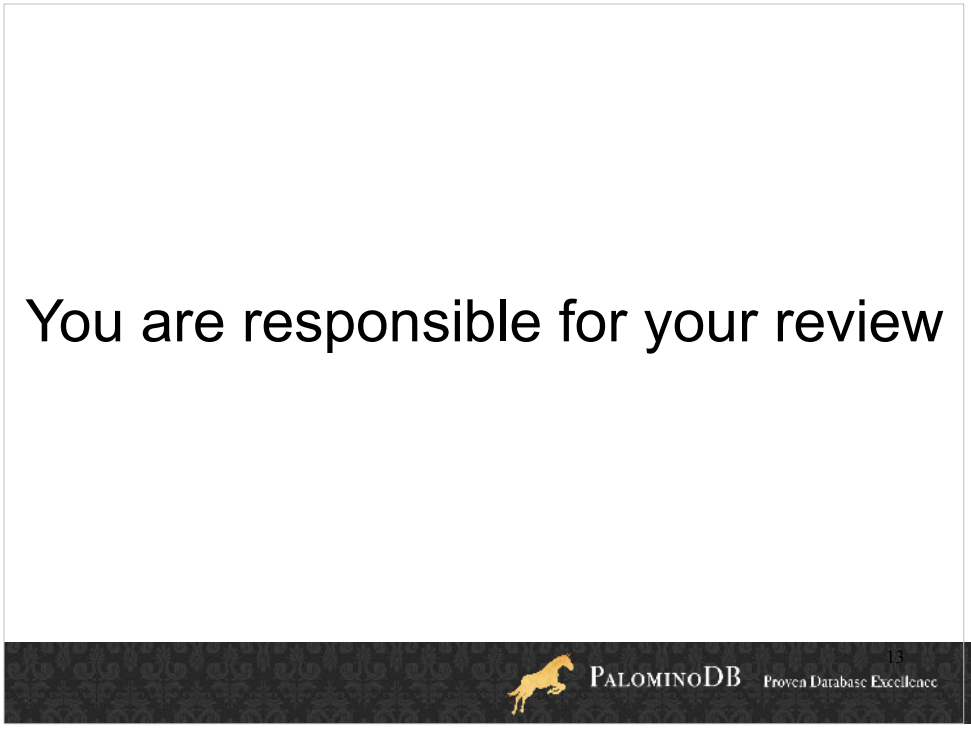
“A development server is a production server to a developer”

Upcoming Projects / Deliverables

- Surprises are bad
- Starting point
- Is anything going to be decomissioned?

Similar to finding out the important servers, find out the non-important servers so you don't spend too much time on those.

You have a starting point so you can track the tasks you do.



What I mean by this is:

Tracking things like how many databases and servers you are handling can give you some good metrics. Imagine being able to sit down at your first annual review and say, “I started out with 10 databases, and now I'm managing 15” or “There were 20 database servers when I started, but I was able to optimize and consolidate so that there are only 10 production databases, with plenty of capacity for the future, and we repurposed those other 10 servers into a full-fledged development environment.”

More is not necessarily better!

It's not your boss' job to remember that, it's yours. And by the way, that makes updating your resume fairly easy. Whenever you're working on a big project, write it down somewhere you'll remember it – for your annual review and for your resume.

Baseline

- Document your starting point
- Server physical config
  - Disks/RAID/SAN, CPU, RAM
  - Where data/logs are stored
- Goal: Track changes over time
  - http://code.google.com/p/mysql-cacti-templates
  - http://www.pythian.com/news/mysql-plugin-for-oracle-grid-control

Not just to have a good review, but also to troubleshoot issues. When I worked for an online dating site we had a huge spike in sales, and we had no idea why – there were no promotions going on.

You've already baseline'd your backups!

We looked at graphs of sales (a count(*) from the sales table in MySQL) and found that there was a similar spike 3 months ago – that's when Marketing said, “oh right, our summer promotion, where you bought 3 months at a reduced price!” This was a residual spike, where lots of those folks were renewing.

Recovery Plan

- What disasters?
- How far back?
  - Disks/RAID/SAN, CPU, RAM
  - Where data/logs are stored
- Goal: Track changes over time
  - http://code.google.com/p/mysql-cacti-templates
  - http://www.pythian.com/news/mysql-plugin-for-oracle-grid-control

If a customer's information is changed? (e.g. becomes free again, loses perks, then re-ups). If a table is dropped? If a server dies? If the 'net dies?

Amazon AWS is a great example!

What if you need to go back a few months, say for a subpoena, how long will that take? Do you have tapes offsite? How big are your backups, how long does it take to get them copied?

Also, consider in a disaster it might be best to be able to have the site up and running, but not necessarily have historical data (or maybe only data in the past 2 weeks or so, or only open orders). You still usually need a subset of db data, such as country lists, products, etc. It might be useful to do logical backups of that, or have that on a separate server if possible, so that in the event that aliens blow up your data center you can at least function going forward.

Now What?

- Prioritize
  - Restores are most important
- “What are your roadblocks?”
- Know the SA team

Sysadmins / network operations folks are like the secretaries of IT. They know everything. You're going to have to troubleshoot a problem with them eventually so get off on the right foot! They're also the ones to build out your machines. You'll also likely work with them for failovers.

Of course this is moot if you are also the SA.

Alerts

- Use role accounts
- Is it important?
  - If not, why are you getting it?
  - e.g. CPU load

Use a role account to receive pages, because at some point you're going to go on vacation.

CPU load, aka “uptime” is the avg number of waiting processes. It doesn't have to do with load at all! With MySQL it's normal to have a load of over 1, even as a baseline, so don't just look at that.

Don't just assume, though. Ask the developers, SA's, customers, customer service reps, etc.

And if you don't have alerting, put it in place. Don't accept “but our customers call us” as an excuse.

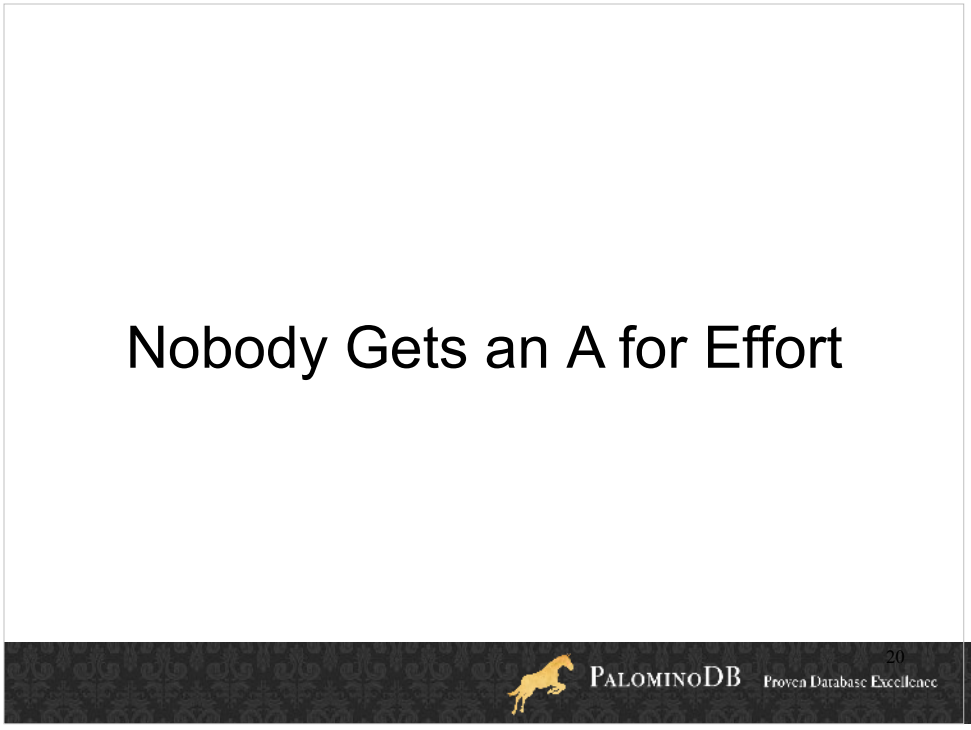
Work Together

- Spend time with coworkers
  - "Grab a cup of coffee"
  - Take a walk
- Even if you are not an extrovert
  - Just another [soft] skill to work on

Even if you don't buy anything, take that walk. Socialize but don't gossip or get involved in office politics.

Take Care of Yourself

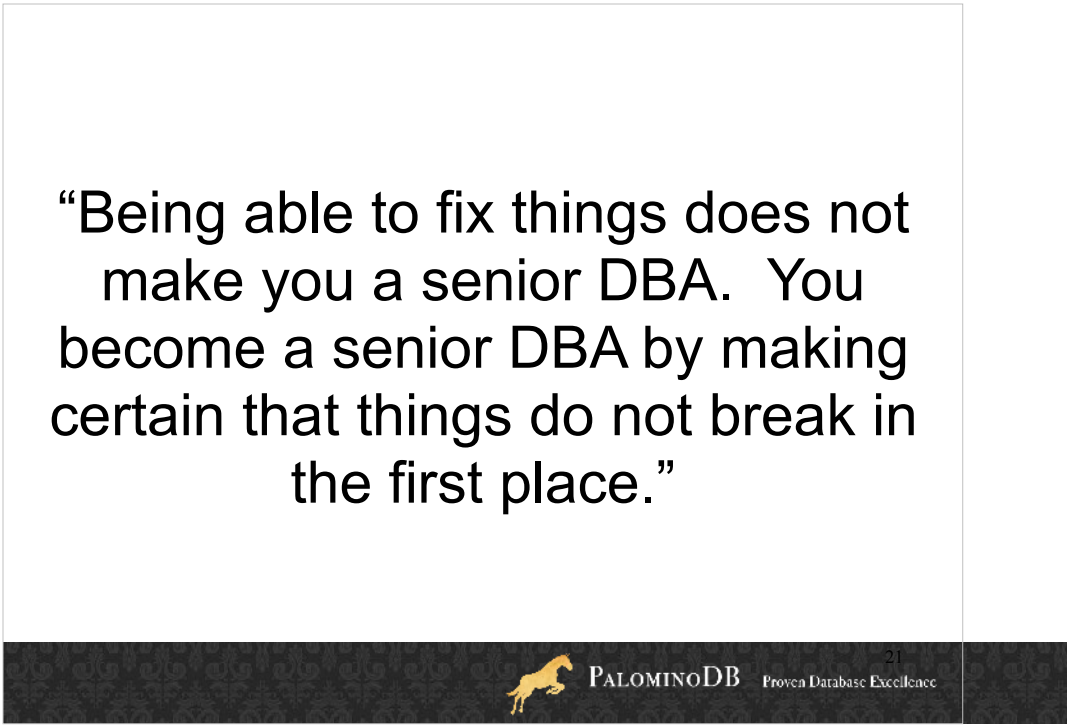
- Do not spend every second with coworkers
- Have a life outside work
- Take care of yourself physically
  - When you feel your best, you do your best



“you worked really hard but the data was lost”

Put the effort in but also be able to track your results. When you came, if you had 3 servers that had backups and now there are 2 more important servers with backups, and by the way you're testing the restores quarterly or monthly, that's progress!

That includes time spent. Be visible about it but not cranky about it. If you need to be at a release, go into the office, so people see you there.



Of course, restoring data in a complicated way is something a senior DBA can do – but hopefully there are checks in place to make sure that does not happen.

The 2-sided DBA

Right	Right Now
• Behind-the-scenes	• Very visible
• “What do you do?”	• “Go see Sheeri”
• Fewer problems	• Very vocal

You get morr praise as “Right Now” but you're a better DBA as “Right”

Solve Root Causes

- Explain optimizations to devs
- Query reviews (mk-query-digest)
  - Slides - http://bit.ly/my10qreview
  - youtube.com/watch?v=IGkBzQuRS5E
- Review server/db configs regularly

Know Underlying Technology

- Disks – RAID, SAN, volumes, partitions
  - Spinning disks vs. SSDs
- How MySQL uses CPU
- How different caches work
  - Filesystem
  - Innodb

Also learn tools that can help you see performance of technology. It will help you debug!

Learn the difference between striping and mirroring, and figure out what makes sense to you. It's all well and good to say “RAID 10 is really reliable” but it might be more expensive than you need, especially if you have failover systems anyway.

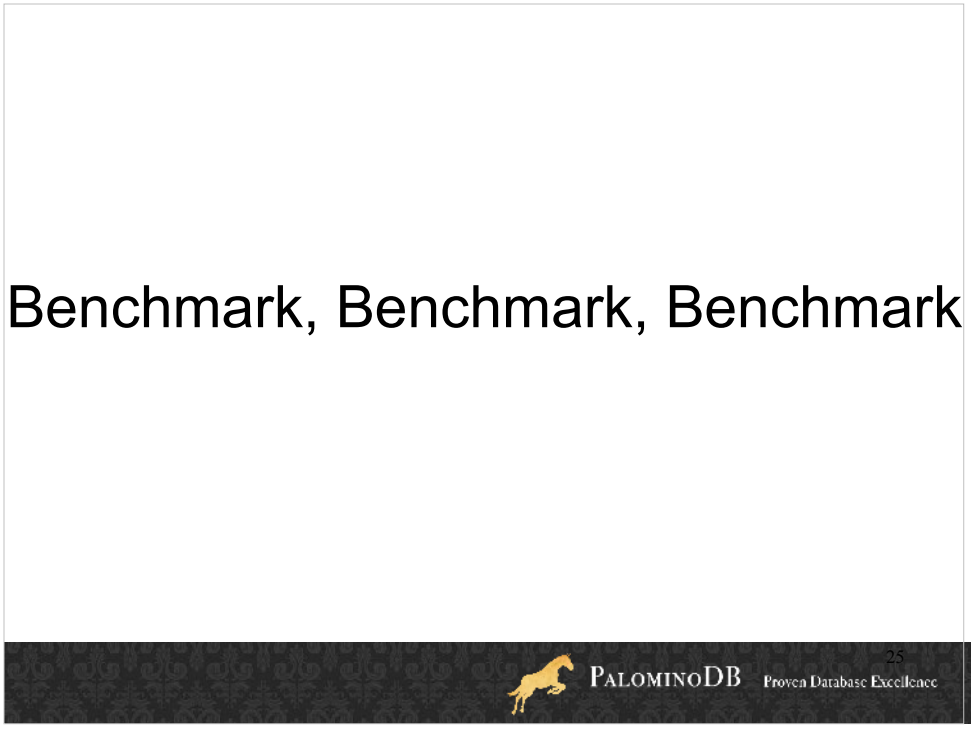
Beware of SANs and shared disk. Or shared anything for that matter. If you know underlying tech, you will be better prepared for confusing stuff like when your Amazon AWS instance works funny.

Spinning disks good for range searches and sequential logs, SSDs better for random.

MySQL uses CPU by filling one CPU and going to the next, so your DB may act as if there were full CPU when it's only 6.25% (e.g. 16 cpu system)

MyISAM uses the filesystem cache for data, and one or more MyISAM key caches for indexes. InnoDB caches everything in a global buffer pool. Write-mostly tables sucha s logging tables are better on MyISAM, at least on the write master.

Other things, like where the temporary db is stored, are also important. Also checkpoints to disk, when memory writes to disk, etc.

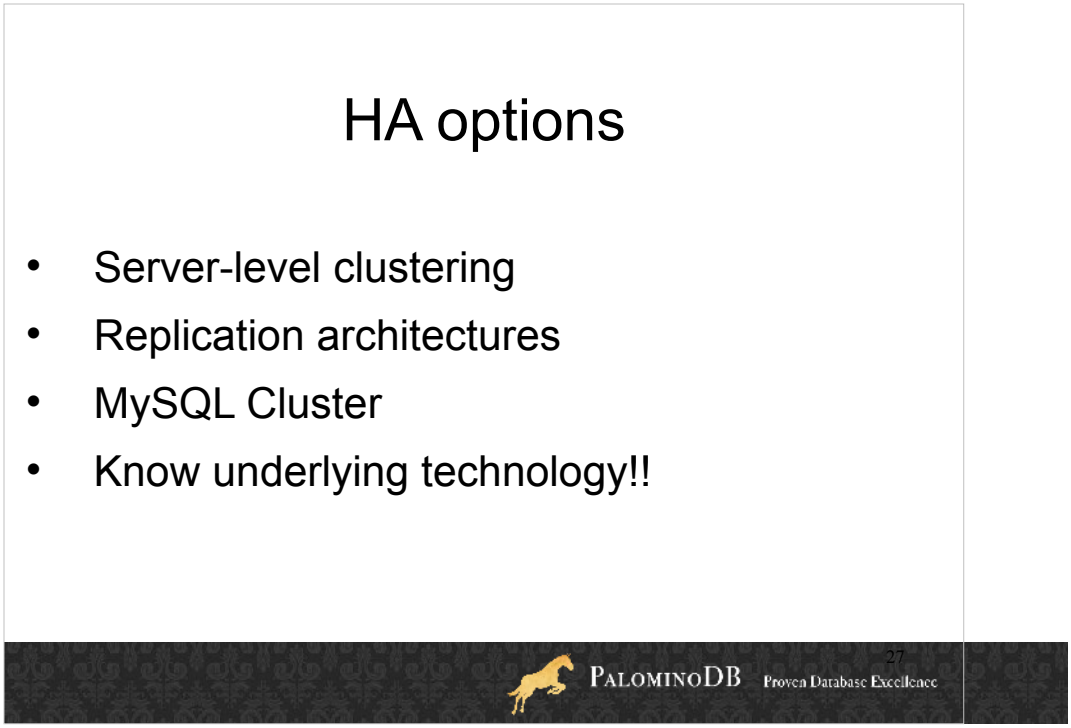


Usually I/O is the problem bottleneck, figure out how much before your server falls over, and make sure you don't get too close to that.

Benchmark as often as you can – preferably upon any change – but also check your production environment to make sure the load is similar. If you benchmark a pattern of usage different from production, it doesn't really matter.

HA options

- HA is not DR
  - Great uptime but no recovery plan
  - Great recovery plan but bad uptime
- HA is not DR



If you know the underlying technology, you can understand what is appropriate.

For example, what works for many shops is to have two servers in a master/master pair, with slaves hanging off one of the servers. However, it is a bad idea to write to more than one machine at a time, because then you do not have a valid source of truth. And you get no benefit because writes cannot be scaled with replication, only reads. So why have a master/master pair? For easy failover and HA.

Learn about row-based replication, statement-based replication and mixed replication. Learn why those are misnomers, because it's not replication that has those options, it's the binary log format. The OurSQL Podcast and the MySQL Administrator's Bible explain these well.

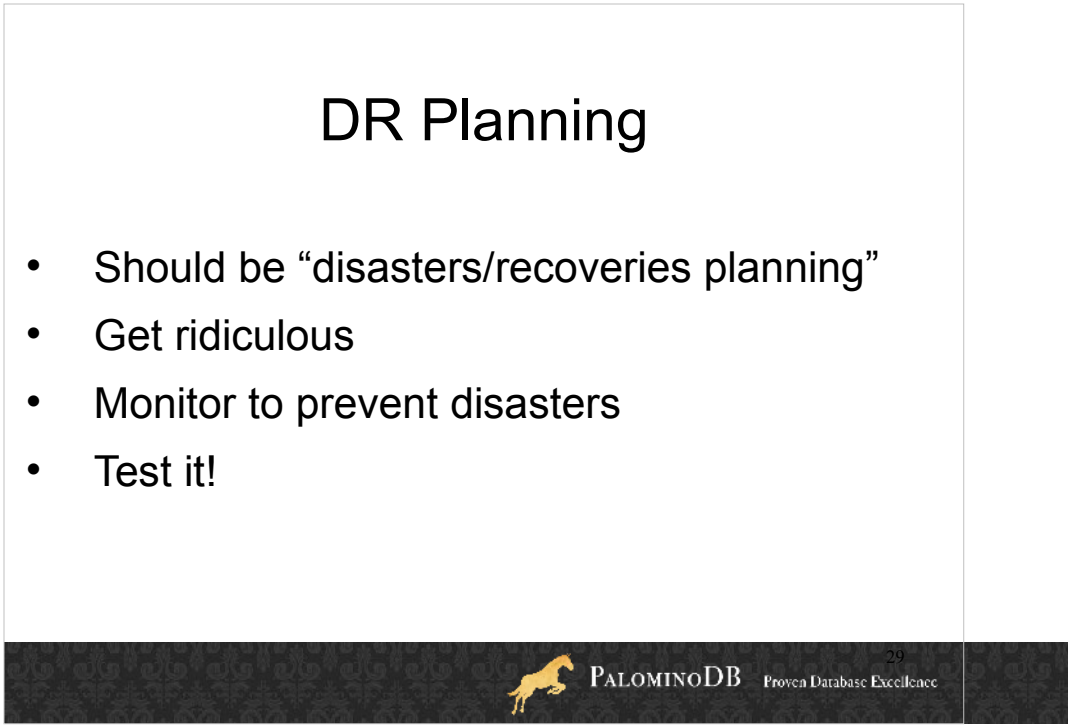
Beware of sites saying “you should set this variable that way” with little or no explanation. Something like “if you set innodb_flush_logs_at_trx_commit=2, there is less I/O” is OK, but understand what that really means – it's a risk, so be sure you want to take it. Are you risking ACID compliance or security to make something faster? Learn WHY, not just “what”.

Where to learn?

- Books
- Blogs - http://planet.mysql.com
- Podcasts (OurSQL, Meet the MySQL Experts)
- Conferences / Training / User Groups
- http://kimtag.com/mysql

Google can be a junior DBA, “trust but verify”. THINK about why that is, and think about the case where that wouldn't work. There's always the exception to prove the rule!

You know more than you think you do – join or start a user group and pick a topic to speak on. You'll hone your skills and be up here as an expert before you know it!



DR is great but most of the time people think about one server. Think about one row, one table, one schema, one server, the whole rack, the whole data center, more than one data center.

Get it in writing; at some point someone will say “that's ridiculous, that won't happen”. Write down “The scenario in which both the New York and Los Angeles data center are down is not covered in this DR plan.”

RAID is all well and good, but you have to monitor the disks. If 1 disk fails you're OK, but if 2 disks fail you're sunk, so make sure you're monitoring your disks!

Like testing restores, test your disaster recovery plan. How long does it take to get up and running? This is why we do fire drills, remember, so that when the time comes there isn't a mass of confusion of what to do.

DBA

- Database Administrator
- Default Blame Acceptor
- Don't Bother Asking

Often the db is blamed, but it's not always the problem. “The DB is down!” might mean “I can't access it” which might be a password or network problem. Be able to help debug those problems, know when it's not your job, but don't be the person who's always saying “that's not my job.” People blame what they don't understand, and you might be able to help them understand better.

Your job isn't to pass around blame. That's not useful. You are a person and there to help, so do a little bit of work before you pass the buck. It also helps if you're wrong, which you will be eventually.

Getting Others to Embrace Change

- Organizational need
- Individual/end user need
- Desired outcome
- Timeline

Monitoring system – organization did not see a need for it.

I had the desire to monitor my db systems, and I had a short timeline and very few resources. In the end, instead of insisting I get hardware, I used an older db server that was not needed, put monitoring on it, and 2 months later they wanted to monitor other stuff, and then I asked for hardware.

Standards / Processes

- People will not like to be limited
- Try to be helpful
- Desired outcome
- Timeline

Most devs should not be involved in a production release, but they might be now. Sit in with them on a few releases, try to get wind of what they're doing, and start to be in on the review process – are they changing tables? Large ones? Will they cause a disturbance to production or replication?

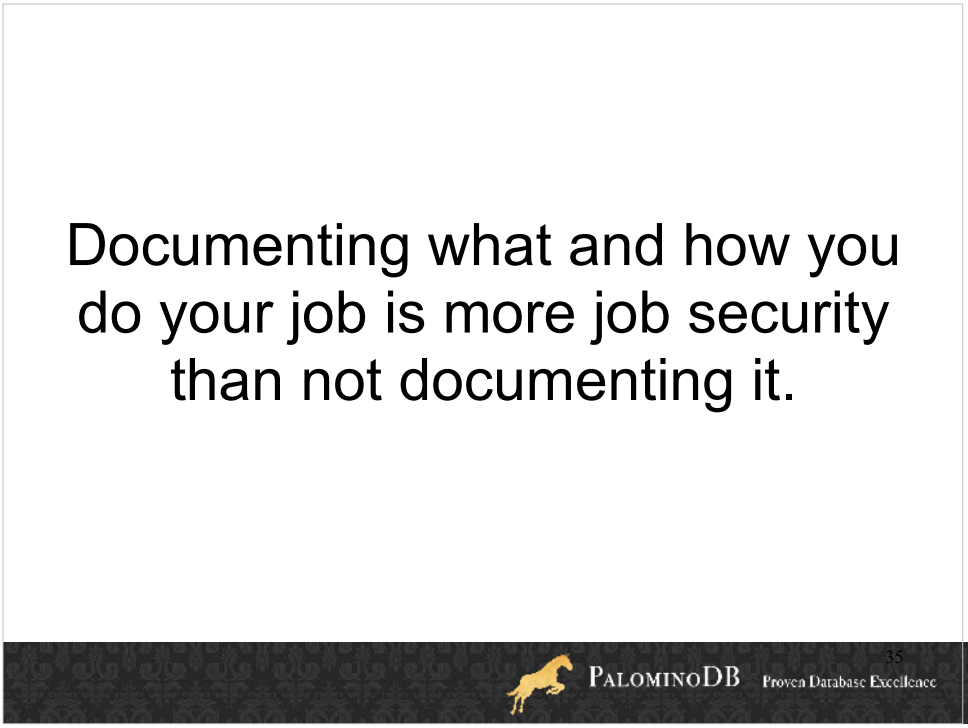
It's really for the benefit of everyone. If you do, say, limit who can create users in production, maybe you don't have to limit that in development. Even if you do, try to be speedy with those requests, so it's not as much of a hardship.

95% of the time, you're right. The 5% of the time you're wrong, though, you don't want to look like a jerk.

Assume the other person knows what they're talking about. That's called "respect". Ask questions like "is it worth the risk of making the backups so large? What's the benefit of storing them in the database? Is it worth the risks?"

Communicate!

- Everybody's needs are important
  - Especially to them!
- Give estimates, update them regularly
  - Once daily or more
- Be honest
- Be transparent

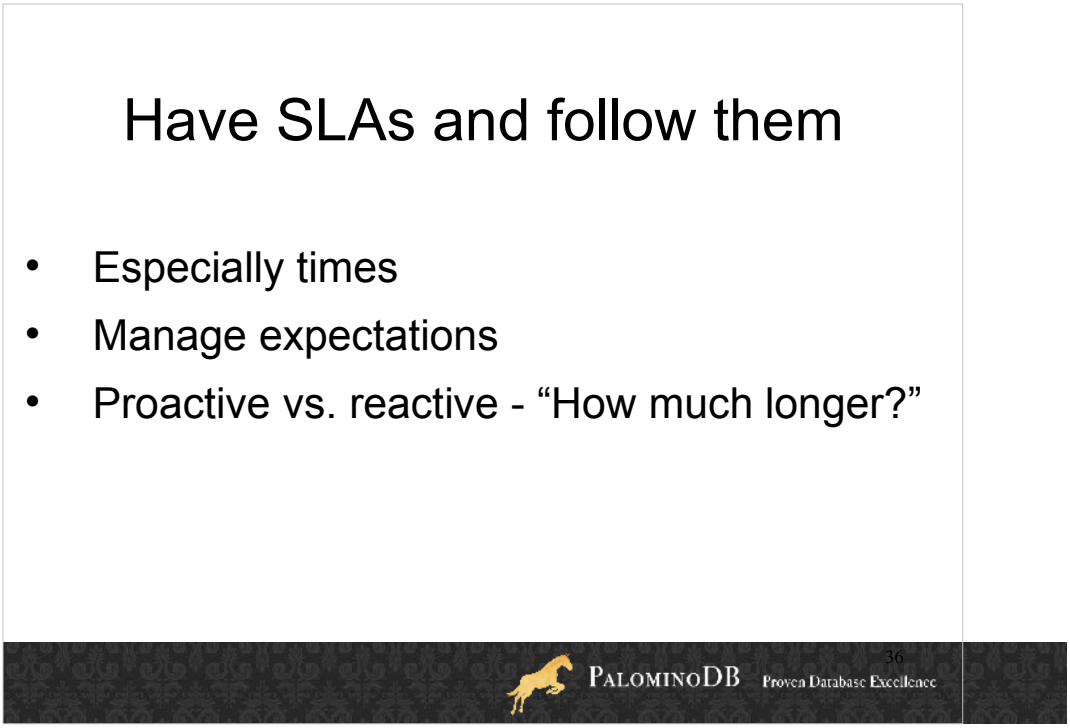


Firstly, people can see what it takes to do your job. Restoring a backup isn't just pressing a button, it's a series of steps to do, including verification.

If you ever run into problems, take your documentation/checklists to a 2nd set of eyes, and get approval. Then you share the responsibility.

You do want to go on vacation, right? Do you want your boss to see all that you do? Document so that you could do it at 3 am, tired and groggy.

If you do not document because you do not want people to see how easy your job is, then you don't actually deserve that raise, do you?



I'm not saying that you shouldn't immediately respond to a 5-minute request when you get back from lunch, when the SLA is 24 hours. But if you constantly do that you are pushing back other tasks, and you set up the expectation that you always do that.

I log out of IM about 30-60 minutes before leaving work. I do not usually check e-mail before leaving work. It's part of work-life balance, too – if they need you they should page you.

Don't try to hope that something will squeak by. If you're seeing that estimates change, communicate those sooner rather than later:

“Is the restore done yet?”

“looks like it's bigger, it should be about 15 minutes more.”

vs.

“Just a note – the restore is taking longer than estimated because the db is bigger than our last test, our new estimated completion is.....”

Sometimes you need to say, “I can spend time answering your questions, or I can spend time fixing this stuff. How about I update you in 15 minutes?”

But if you do that, FOLLOW THROUGH.

Being a DBA is like a buffet; sometimes you just have to say no or you'll be stuffed.

Documentation = Communication

- Not the only means
- But an important one
- Have meetings where you read the plan aloud and go through steps if need be
- Not everyone reads e-mail!

Outsourcing

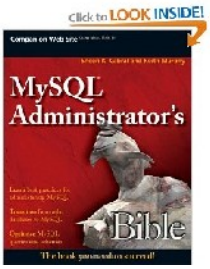
- If you need an expert
- If you need another warm body
- If you need a 2nd set of eyes
- Documentation helps set metrics for outsourcing

If you know how things are done, you can tell if your outsourcing is of good or poor quality.

Questions? Comments?

OurSQL podcast

MySQL Administrator's Bible



kimtag.com/mysql

planetmysql.com