



PALOMINO DB

Advanced MySQL Scaling Strategies for Developers

Presented by:

Jonathan Levin

Operational DBA, PalominoDB

www.palominodb.com

Who is the tutorial for?



What won't it cover?



What will it cover?



Fundamentals + Advanced Techniques



20% that makes
80% of the difference



What is wrong with
relational databases?

or

Why are we here?



Slow



PALOMINODB

Proven Database Excellence

Transactions

Lock/Stall/Deadlock out



Forces you into a
relational schema



SQL is annoying

(terrible)



SELECT

```
CONTACT.CON_ID as contact_id,
contact.CON_EMAIL_LOWER as email,
contact.CON_CREATION_DATE as creation_date,
lastname.CONATTR_VALUE as givenname,
firstname.CONATTR_VALUE as familyname,
languages.CONATTR_VALUE as "LANGUAGE",
gender2.SELVAL_VALUE as gender,
mobile.CONATTR_VALUE as mobilephone,
office.CONATTR_VALUE as officephone,
home.CONATTR_VALUE as home,
address.CONATTR_VALUE as address,
zipcode.CONATTR_VALUE as zipcode,
province2.SELVAL_VALUE as province,
city2.SELVAL_VALUE as city,
intendedpurchase2.SELVAL_VALUE as intendedpurchase,
ownedcarbrand2.SELVAL_VALUE as ownedcarbrand,
minimodel3.SELVAL_VALUE as minimodel,
dealer3.SELVAL_VALUE as dealer,
subject3.SELVAL_VALUE as subject,
SUBSCRIPTION.sub_id as subscription_id,
brochure13.SELVAL_VALUE as brochure1,
brochure23.SELVAL_VALUE as brochure2,
brochure33.SELVAL_VALUE as brochure3,
message2.CAMATTR_VALUE as message
from CONTACT contact
left join CONTACT_ATTRIBUTE lastname on lastname.CONATTR_CON_ID = CONTACT.CON_ID and lastname.CONATTR_ATTR_ID = 1010
left join CONTACT_ATTRIBUTE firstname on firstname.CONATTR_CON_ID = CONTACT.CON_ID and firstname.CONATTR_ATTR_ID = 1020
left join CONTACT_ATTRIBUTE languages on languages.CONATTR_CON_ID = CONTACT.CON_ID and languages.CONATTR_ATTR_ID = 1000
left join
(SELECTION_VALUE gender2 INNER JOIN CONTACT_ATTRIBUTE gender on gender.CONATTR_SEL_ID=gender2.SELVAL_SEL_ID and
gender2.SELVAL_LANGUAGE = 'zh')
on gender.CONATTR_CON_ID = CONTACT.CON_ID and gender.CONATTR_ATTR_ID = 1030
left join CONTACT_ATTRIBUTE mobile on mobile.CONATTR_CON_ID = CONTACT.CON_ID and mobile.CONATTR_ATTR_ID = 1100
left join CONTACT_ATTRIBUTE office on office.CONATTR_CON_ID = CONTACT.CON_ID and office.CONATTR_ATTR_ID = 1110
left join CONTACT_ATTRIBUTE home on home.CONATTR_CON_ID = CONTACT.CON_ID and home.CONATTR_ATTR_ID = 1120
left join CONTACT_ATTRIBUTE address on address.CONATTR_CON_ID = CONTACT.CON_ID and address.CONATTR_ATTR_ID = 1200
left join CONTACT_ATTRIBUTE zipcode on zipcode.CONATTR_CON_ID = CONTACT.CON_ID and zipcode.CONATTR_ATTR_ID = 1210
left join
(SELECTION_VALUE province2 INNER JOIN CONTACT_ATTRIBUTE province on province.CONATTR_SEL_ID=province2.SELVAL_SEL_ID
and province2.SELVAL_LANGUAGE = 'zh')
on province.CONATTR_CON_ID = CONTACT.CON_ID and province.CONATTR_ATTR_ID = 510
left join (SELECTION_VALUE city2 INNER JOIN CONTACT_ATTRIBUTE city on city.CONATTR_SEL_ID=city2.SELVAL_SEL_ID and
city2.SELVAL_LANGUAGE = 'zh')
on city.CONATTR_CON_ID = CONTACT.CON_ID and city.CONATTR_ATTR_ID = 520
left join (SELECTION_VALUE intendedpurchase2 INNER JOIN CONTACT_ATTRIBUTE intendedpurchase
on intendedpurchase.CONATTR_SEL_ID=intendedpurchase2.SELVAL_SEL_ID and intendedpurchase2.SELVAL_LANGUAGE = 'zh')
on intendedpurchase.CONATTR_CON_ID = CONTACT.CON_ID and intendedpurchase.CONATTR_ATTR_ID = 1300
left join (SELECTION_VALUE ownedcarbrand2 INNER JOIN CONTACT_ATTRIBUTE ownedcarbrand
on ownedcarbrand.CONATTR_SEL_ID=ownedcarbrand2.SELVAL_SEL_ID and ownedcarbrand2.SELVAL_LANGUAGE = 'zh')
on ownedcarbrand.CONATTR_CON_ID = CONTACT.CON_ID and ownedcarbrand.CONATTR_ATTR_ID = 1400
left join SUBSCRIPTION on CONTACT.CON_ID=SUBSCRIPTION.SUB_CON_ID
```

```
left join
(SELECTION_VALUE minimodel3 INNER JOIN
(CAMPAIGN_ATTRIBUTE minimodel2 INNER JOIN SUBSCRIPTION minimodel on minimodel2.CAMATTR_SUB_ID=minimodel.SUB_ID)
on minimodel2.CAMATTR_ATTR_ID=52001)
on minimodel2.CAMATTR_SEL_ID = minimodel3.SELVAL_SEL_ID and minimodel3.SELVAL_LANGUAGE = 'zh' and minimodel.SUB_CON_ID=CONTACT.CON_ID
and SUBSCRIPTION.SUB_ID in (select distinct(CAMATTR_SUB_ID) from CAMPAIGN_ATTRIBUTE group by CAMATTR_SUB_ID)
left join
(SELECTION_VALUE dealer3 INNER JOIN
(CAMPAIGN_ATTRIBUTE dealer2 INNER JOIN SUBSCRIPTION dealer on dealer2.CAMATTR_SUB_ID=dealer.SUB_ID)
on dealer2.CAMATTR_ATTR_ID=52002)
on dealer2.CAMATTR_SEL_ID = dealer3.SELVAL_SEL_ID and dealer3.SELVAL_LANGUAGE = 'zh' and dealer.SUB_CON_ID=CONTACT.CON_ID
and SUBSCRIPTION.SUB_ID in (select distinct(CAMATTR_SUB_ID) from CAMPAIGN_ATTRIBUTE group by CAMATTR_SUB_ID)
left join
(SELECTION_VALUE subject3 INNER JOIN
(CAMPAIGN_ATTRIBUTE subject2 INNER JOIN SUBSCRIPTION subject on subject2.CAMATTR_SUB_ID=subject.SUB_ID)
on subject2.CAMATTR_ATTR_ID=50001)
on subject2.CAMATTR_SEL_ID = subject3.SELVAL_SEL_ID and subject3.SELVAL_LANGUAGE = 'zh' and subject.SUB_CON_ID=CONTACT.CON_ID
and SUBSCRIPTION.SUB_ID in (select distinct(CAMATTR_SUB_ID) from CAMPAIGN_ATTRIBUTE group by CAMATTR_SUB_ID)
left join
(SELECTION_VALUE brochure13 INNER JOIN
(CAMPAIGN_ATTRIBUTE brochure12 INNER JOIN SUBSCRIPTION brochure1 on brochure12.CAMATTR_SUB_ID=brochure1.SUB_ID)
on brochure12.CAMATTR_ATTR_ID=51001 and brochure12.CAMATTR_SEL_ID=5100101)
on brochure12.CAMATTR_SEL_ID = brochure13.SELVAL_SEL_ID and brochure13.SELVAL_LANGUAGE = 'zh' and brochure1.SUB_CON_ID=CONTACT.CON_ID
and SUBSCRIPTION.SUB_ID in (select distinct(CAMATTR_SUB_ID) from CAMPAIGN_ATTRIBUTE group by CAMATTR_SUB_ID)
left join
(SELECTION_VALUE brochure23 INNER JOIN
(CAMPAIGN_ATTRIBUTE brochure22 INNER JOIN SUBSCRIPTION brochure2 on brochure22.CAMATTR_SUB_ID=brochure2.SUB_ID)
on brochure22.CAMATTR_ATTR_ID=51001 and brochure22.CAMATTR_SEL_ID=5100102)
on brochure22.CAMATTR_SEL_ID = brochure23.SELVAL_SEL_ID and brochure23.SELVAL_LANGUAGE = 'zh' and brochure2.SUB_CON_ID=CONTACT.CON_ID
and SUBSCRIPTION.SUB_ID in (select distinct(CAMATTR_SUB_ID) from CAMPAIGN_ATTRIBUTE group by CAMATTR_SUB_ID)
left join
(SELECTION_VALUE brochure33 INNER JOIN
(CAMPAIGN_ATTRIBUTE brochure32 INNER JOIN SUBSCRIPTION brochure3 on brochure32.CAMATTR_SUB_ID=brochure3.SUB_ID)
on brochure32.CAMATTR_ATTR_ID=51001 and brochure32.CAMATTR_SEL_ID=5100103)
on brochure32.CAMATTR_SEL_ID = brochure33.SELVAL_SEL_ID and brochure33.SELVAL_LANGUAGE = 'zh' and brochure3.SUB_CON_ID=CONTACT.CON_ID
and SUBSCRIPTION.SUB_ID in (select distinct(CAMATTR_SUB_ID) from CAMPAIGN_ATTRIBUTE group by CAMATTR_SUB_ID)
left join
(CAMPAIGN_ATTRIBUTE message2 INNER JOIN SUBSCRIPTION message on message2.CAMATTR_SUB_ID=message.SUB_ID and
message2.CAMATTR_ATTR_ID=50002)
on message.SUB_CON_ID=CONTACT.CON_ID
and SUBSCRIPTION.SUB_ID in (select distinct(CAMATTR_SUB_ID) from CAMPAIGN_ATTRIBUTE group by CAMATTR_SUB_ID)
```



What is right with
relational databases?



How important is
your data to you?



What if the transaction
lost your money?



How do I add, change and
retrieve my data?



What are the alternatives to SQL?

(for Relational DBs)



Who does like SQL?





Can relational databases be fast?

/discuss





Relational Database

SQL Parser

Optimizer

Storage Engine



Using Concepts



The Database as a Lean Factory



Agile/Scrum

VS

Lean (Six Sigma)



Revolution

VS

Evolution
Theory of Constraints

Velocity and
Estimation of Effort

Intervals, defects and
variations



Value and Waste



Instrumentation

(Monitoring, SLAs)



New Features

(Simplification)



Organic Growth

(Strain/Bottlenecks)



Library Metaphor

(Understanding Indexes)





(image taken from <http://www.flickr.com/photos/reedinglessons/2239767394/>)



Typical Usage

(pretend you're doing it)



Full table scan



Index range scan



Covering Index



Inserting a new book



Updating or removing a book



Storage Engines in the Library



Students doing homework



Students and Casual readers



Students withholding books



Students doing a project



Sorting with index cards

(Group by, Order by)



Indexes vs Map Reduce



Working desks

(Temporary tables)



Book donations



Books, Indexes and Library space

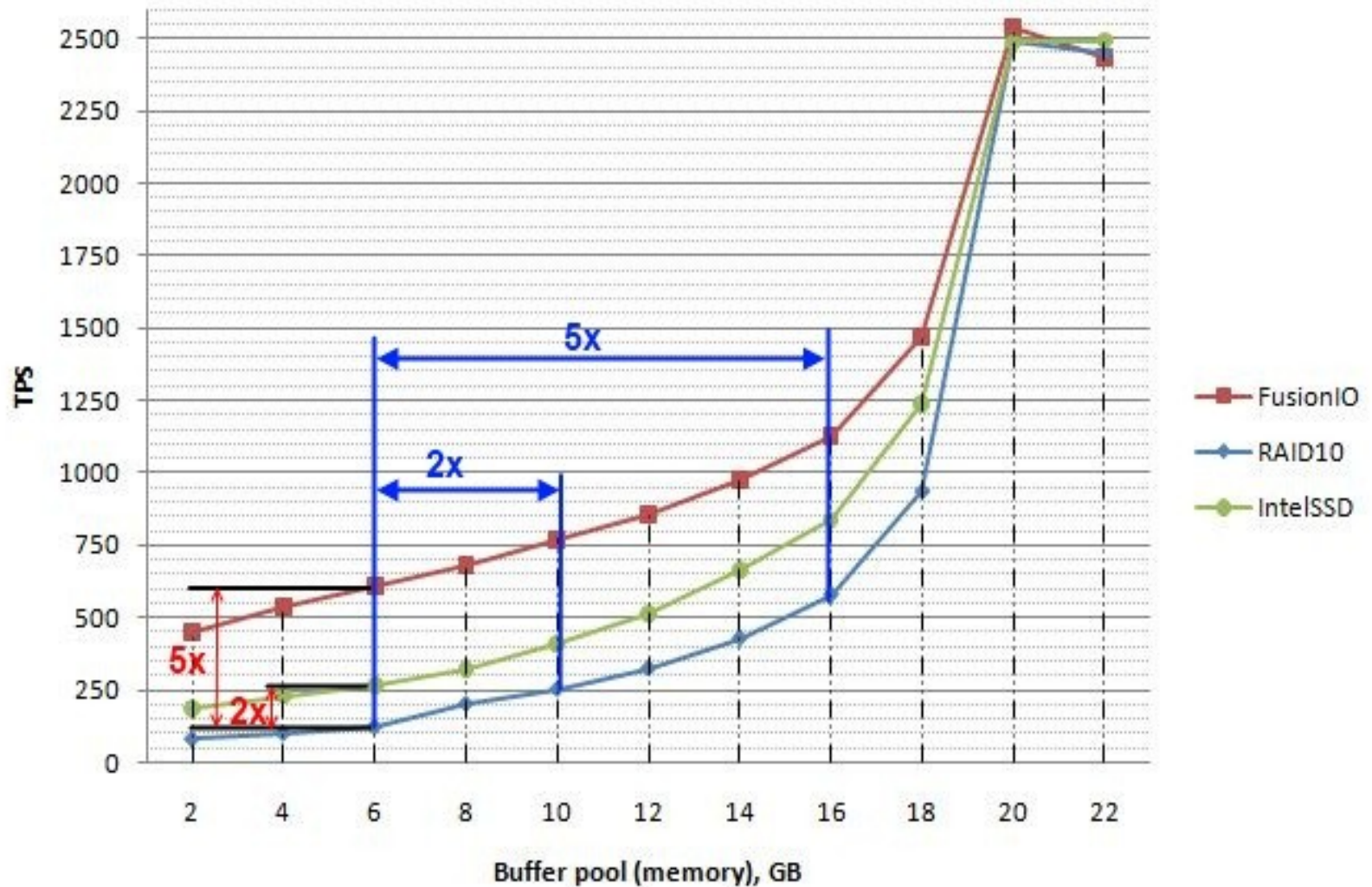


Book Cart

(memory)



sysbench oltp, 80mln rows (18GB data)



<http://www.mysqlperformanceblog.com/2010/04/08/fast-ssd-or-more-memory/>



PALOMINO DB

Proven Database Excellence

Back to the factory



Making a New Widget



Queries

(Think indexes)



Regular Usage

```
SELECT ▲, ■ FROM T̄  
WHERE Ω = 121;
```

(PRIMARY KEY Ω)



Range Scan

```
SELECT ▲, ■ FROM T̄  
WHERE Ω = 121  
AND ♪ BETWEEN 1 AND 100;
```

KEY (Ω, ♪)



Range Scan

```
SELECT ▲, ■ FROM T̄  
WHERE Ω = 121  
AND ♪ IN (1,100,30,7);
```

KEY (Ω, ♪)



Covering Index

```
SELECT ▲, ■ FROM T̄  
WHERE Ω = 121;
```

```
KEY (Ω, ▲, ■)
```



Not Optimal

```
SELECT ▲, ■ FROM T̄  
WHERE Ω = 121  
AND ♪ IN (1,100,30,7);
```

KEY (Ω, ▲, ■)



Broken Range

```
SELECT ▲, ■ FROM T̄  
WHERE Ω = 121  
AND ♪ IN (1,100,30,7);
```

KEY (♪,Ω)



Sub Queries

```
SELECT ▲, ■ FROM T̄  
WHERE ♪ IN (SELECT ♪ FROM ♠);
```

KEY (♪)



Indexes for Sorting

```
SELECT ☀, 😊 FROM T̄  
WHERE Ω = 121  
GROUP BY ▲  
ORDER BY ■;
```

KEY (Ω)



Indexes for Sorting

```
SELECT ☀, 😊 FROM T̄  
WHERE Ω = 121  
GROUP BY ▲  
ORDER BY ■;
```

KEY (Ω, ▲, ■)



Indexes for Joins

```
SELECT T.☀, M.😊 FROM T  
INNER JOIN M ON T.id = M.user_id  
WHERE T.Ω = 232;
```

```
T KEY (Ω)  
M KEY(user_id)
```



Join vs Sub-query



This guy in #MySQL chat asked..

```
SELECT ▲, (SELECT .. FROM WHERE )  
FROM ¯, (SELECT .. FROM WHERE) as £  
WHERE ♪ IN  
(SELECT ♪ FROM ♠ WHERE..);
```

I need to optimize my db config file..



Clustered PK and Secondary Indexes

KEY (▲,■,id)

PRIMARY KEY (id)
KEY (▲,■)



Index Merge

```
SELECT ☺ FROM T  
WHERE ▲ =1 OR ■ =2;
```

```
KEY (■)  
KEY (▲)
```



Index Merge

```
SELECT ☺ FROM T̄ WHERE ▲ =1  
UNION (SELECT ☺ FROM T̄  
WHERE ■ =2);
```

KEY (■)
KEY (▲)



Poor Librarians :(

(Appreciate the Librarians)



EXPLAIN Yourself!

Mr. Optimizer



Cardinality and the Optimizer

SHOW INDEX FROM Table;



Where did the resource
spend time on?

SHOW PROFILE;



How to find “bad” queries?



New Query



Slow web page

(instrumentation)



Slow/General log + query-digest

(microtime patch)



```
wget percona.com/get/pt-query-digest
mysql>set global long_query_time = 0;
mysql>set log_slow_verbosity = 'microtime'
(or 'full');
mysql>flush logs;
..Wait a while..
./pt-query-digest      -limit=100%      slow.log
>slow.txt
```



Continuous Improvement



Caching

(Layers)



Static Data

(Browser Cache, Reverse Proxy)



Caching Objects

(Large and Small. Memcached)



Database

(Buffer Pool, Key Buffer, Query Buffer..)



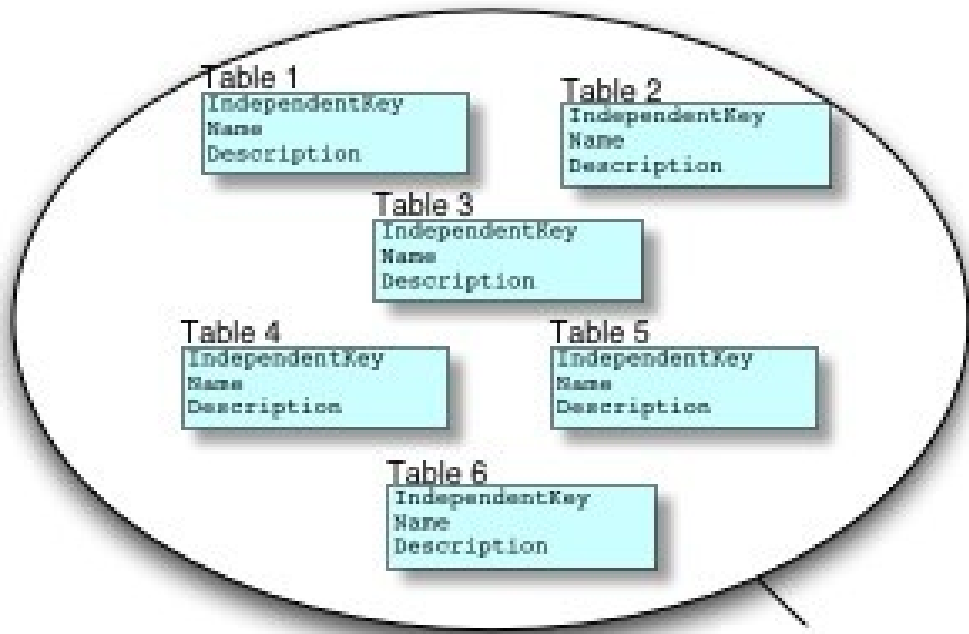
Query Cache vs. Memcached



Tables

(Summary Tables, Denormalized..)





Aggregation

Summary Table

IndependentKey
Name
Description

Join

Join Table

IndependentKey
Name
Description



Columns

(Denormalized, “Hot”, Preprocessed)



Tables != Objects

(Impedance mismatch. /discuss)



Proxy, Decorator, Adapter, and Bridge are all variations on "wrapping" a class. But their uses are different.

- **Proxy** could be used when you want to lazy-instantiate an object, or hide the fact that you're calling a remote service, or control access to the object.
- **Decorator** is also called "Smart Proxy." This is used when you want to add functionality to an object, but not by extending that object's type. This allows you to do so at runtime.
- **Adapter** is used when you have an abstract interface, and you want to map that interface to another object which has similar functional role, but a different interface.
- **Bridge** is very similar to Adapter, but we call it Bridge when you define both the abstract interface and the underlying implementation. I.e. you're not adapting to some legacy or third-party code, you're the designer of all the code but you need to be able to swap out different implementations.
- **Facade** is a higher-level (read: simpler) interface to a subsystem of one or more classes. Think of Facade as a sort of container for other objects, as opposed to simply a wrapper.

[link](#) | [improve this answer](#)

edited [Mar 20 '09 at 16:02](#)

answered [Dec 8 '08 at 18:56](#)



[Bill Karwin](#)

89.1k ● 10 ● 97 ● 213

<http://bit.ly/pXVQhV>



PALOMINO^{DB}

Proven Database Excellence

Problems with Cache



Cache Misses

(for Buffers)



Stale Data



Cost of Generation



Mass Invalidation



Dependency

(High Availability)



Using Time to Your Advantage

(Understanding your application)



Pre-Generation

(no misses)



Queues

(Async, Parallel)
(Errors, Dependency)



Pre-Processing

(Off-peak, intervals)



Event Processing vs Batch Processing



Streaming

(Frequently Asked Queries)



Schema`s

(and what to do about them)



Tables != Objects



Think Spreadsheets

(remember 3rd normal form?)



id	first name	last name	phone number	email
1	Bob	Smith	1234567	bob@smith.com
2	Jim	Jones	2345678	jim@jones.com
3	Frederic	Gryp	3456789	frederic@gryp.com

Look how well it fits



Some Guidelines on Schema Design



Set Datatypes Conservatively



..But make them large
just in case.

(You don't want to get errors)



Make sure you have
indexes for all your queries



But too many indexes can slow
down and bloat the DB



Normalize to the Nth form



Denormalize to
improve speed



The key is balance



and actual usage



How are you accessing your
data at the moment?

(query-digest)



Are your datatypes suitable?

procedure analyse(1,1)\G



Do you have too many indexes?

pt-index-usage
+ pt-duplicate-key-checker
USER_STATISTICS



Indexes != Data

Index to satisfy queries
(95% or more)



Can your indexes be more lean?

(Less data to go through)

(More to fit in memory)

(Primary keys in clustered indexes)



Useful queries for MySQL table sizes

<http://www.mysqlperformanceblog.com/2008/03/17/researching-your-mysql-table-sizes/>

<http://code.google.com/p/common-schema/>

Careful about running heavy information_schema queries

<http://palominodb.com/blog/2011/10/12/why-are-your-indexes-larger-your-actual-data>



Count (distinct left(column,20)) / count(distinct column)

c200	c100	c75	c50	c30	c20
100.00	100.00	100.00	99.97	99.81	92.86

Limiting Character Indexes



Write Scaling

(Partitioning, Sharding, etc..)



Or is the database
too big?

or is it both?



What doesn't help scale writes?

(Master/Master, DRDB, MMM)



Functional Partitioning

(Per Application)



Manual Partitioning

Regular Table and Archive Table
pt-archiver



MySQL Partitioning

(For Tables)



Sharding

(The Secret Sauce)



Complexity

Backups, HA, Monitoring..



Removing bloat Improves scaling

Less bloat = Less waste = More Lean



Queues

Dependency, Bypassing ACID



Alternatives..

Memcached w/ periodic flushing,
Cluster, Voldermort, Riak..



Almost at the end..



Scaling vs New Features

Don't hurt the business



Don't over-complicate things

Do just enough right now
Increase standards later



How can this add value?

How can I remove waste from this?



This tutorial did not
cover everything

It hopefully, gave you a new way of thinking.
Now it is time to learn more.



If we have time:

Trends with hardware

ORMs

NoSQL



Thank you very much!



Need Proactive Operational Database Support?

- * Proactive team integration
- * 24x7 Tier 1 or 2 emergency support
- * MySQL and variants, PostgreSQL, NoSQL and Key/Value stores
- * Data Modeling, Release Process Integration, Proactive SQL Reviews
- * Monitoring and Trending
- * High Availability Solutions
- * Backup, Recovery and Disaster Recovery
- * BI, Data Analysis and Warehousing



Contact:

jonathan@palominodb.com

skype: jonathan_palominodb

(We're Hiring!)

