The state of databases in the current landscape

July, 2023

Michael Widenius
Creator of MySQL and MariaDB
CTO @ MariaDB

Overview

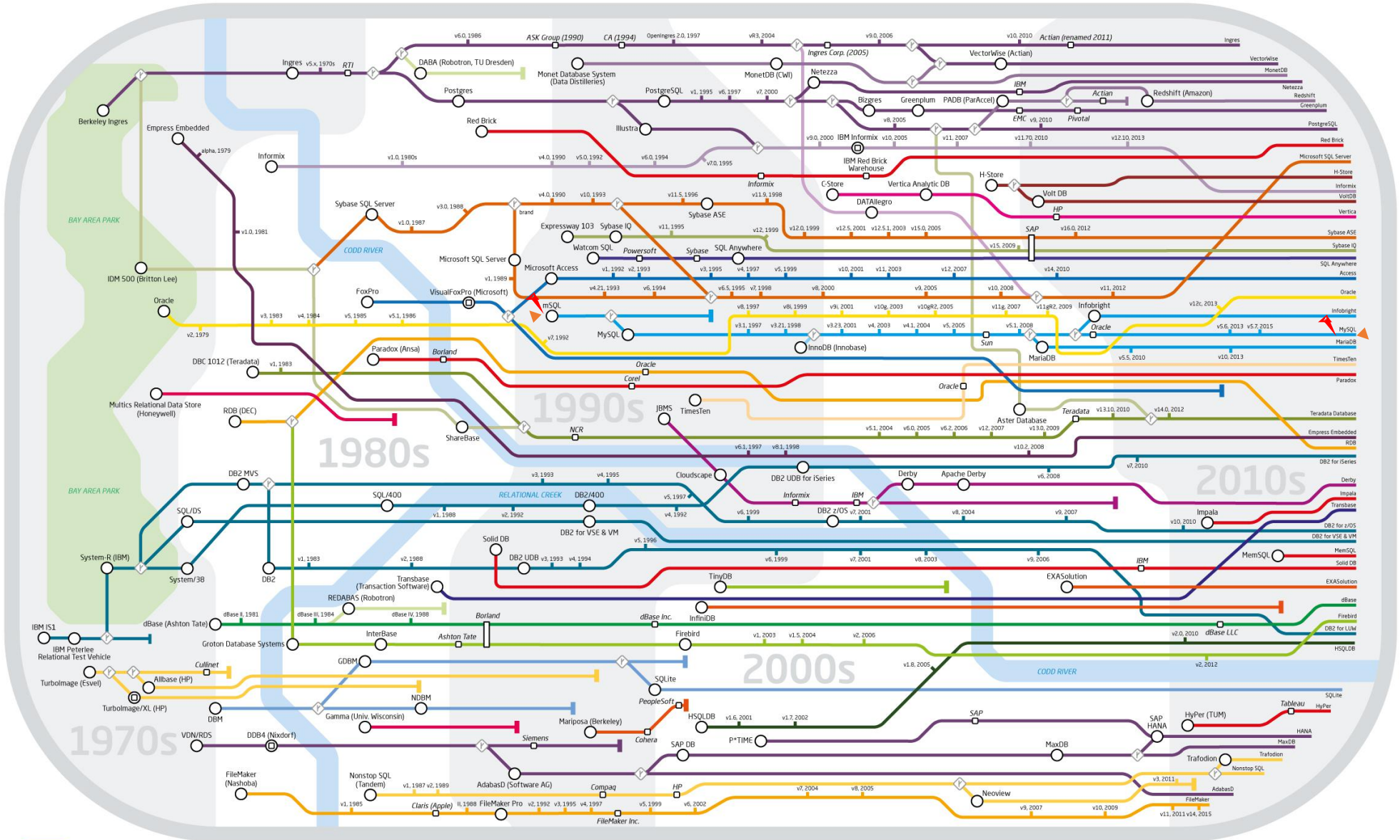
- The History of RDBMS and MariaDB
- Big Data is driven by Open Source
- AI and databases
- Databases and the cloud
- What I have been working on lately

MariaDB

The history of RDBMS

Genealogy of Relational Database Management Systems

We Start Here



Here Now

Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Key to lines and symbols

DBMS name (Company) | *Acquisition* | v9, 2006 Versions | Discontinued | Branch (intellectual and/or code) | Crossing lines have no special semantics

Felix Naumann, Jana Bauckmann, Claudia Exeler, Jan-Peer Rudolph, Fabian Tschirschnitz
Contact - Hasso Plattner Institut, University of Potsdam, felix.naumann@hpi.de
Design - Alexander Sandt Grafik-Design, Hamburg

Version 6.0 - October 2018
https://hpi.de/naumann/projects/rdbms-genealogy.html

What made MySQL successful?

- When MySQL was started (1994) there were big players such as Microsoft, IBM, Oracle, all having a significant portion of the market.
  - Internet was new and everyone needed a web-optimized DB
- However, the big players did not see the Internet as a viable business platform!
- MySQL was already proven stable before release (used for data warehousing and web)
- We created a “Virtual company”, which made it easy to find good people
- New “free” license scheme (this was before Open Source)
  - Free for most, a few have to pay
  - Second program (ghostscript was first) to use dual licensing.
    - MySQL was first to do it with GPL.
- Very easy to install and use (15 minute rule)
- Released source and tested binaries for most platforms
- MySQL was a needed, stable and easy to use product with the right price

What made MySQL successful?

- MySQL started from the bottom–up, providing a cheap (or free) solution for the web industry.
  - We were friendly and helpful towards community
- I personally wrote 30,000+ emails the first 5 years to help people using MySQL
  - MySQL then started growing in other industries (enterprise sector).
  - MySQL followed an open source development model.
- "1 customer in 1000 actually pays" – Still enough to grow very fast.
- The community provided testing, marketing and simple support needs.
- MySQL Ab provided full support for those that wanted/needed it.

- We (the MySQL founders) waited with investments until product was “good enough”

Why MariaDB was created

"Save the People, Save the Product"

- To keep the MySQL talent together
- To ensure that a free version of MySQL always exists
- To get one community developed and maintained branch
- Work with other MySQL forks/branches to share knowhow and code

- After Oracle announced it wanting to buy Sun & MySQL this got to be even more important.

How was MariaDB disruptive?

MariaDB follows in the same original footsteps of MySQL (which Oracle has not done):

- It is a true to Open Source project, following proper Free Software / Open Source practices.
- Development happens in the open, working together with the community.
- **MariaDB Foundation** was created to ensure that MariaDB would always be Open Source.
- This process made MariaDB stand out.
  - MariaDB has been integrated into most major Linux and other free OS distributions as the default “MySQL” variant.

MariaDB Ecosystem

MariaDB Foundation
Works with the community

Builds and tests binaries
- Develops MariaDB buildbot

Drives adoption
- Works with OS to ensure MariaDB is included everywhere

Works with community developers
- Reviews architecture and patches
- Approves and pushes changes

Insures that MariaDB is always free.
Founded trough sponsorship's.

MariaDB Corporation
Works with customers

Provides paid support and subscriptions for MariaDB (Enterprise and Community)

Employs most of the MariaDB developers
- Main driver of MariaDB development

MariaDB Enterprise
- Longer End-of-life
- Stable features are backported to earlier versions to minimize needs for upgrades.

Provides NRE (paid development of new MariaDB features).

MariaDB future plans

Monty's view

- Work with customers to help them migrate from commercial closed source database to MariaDB
  - The MariaDB Oracle compatibility layer makes this easy
- Improve optimizer to handle very complex queries
  - This is important when moving complex application to MariaDB
  - MariaDB 11.0 has a new cost model that greatly improves handling of complex queries.
- Work closer with SAS database providers to make MariaDB work better in their environment
  - MariaDB multi–tenancy feature is part of this collaboration.

Big data is driven by Open Source

MariaDB

Big data driven by Open Source

- MySQL / MariaDB
  - SQL database with flexible replication, ColumnStore and Spider for scale out
- Apache Cassandra
  - Tunable consistency
  - Keys map to multiple values, which are grouped into column families
  - CQL language
- MongoDB
  - Stores structured data as JSON like documents
- CouchDB
  - MVCC, ACID, eventually consistent
- Hbase (Hadoop + distributed file system)
  - Written in Java. Compression, in–memory operation and Bloom filters (Is data part of a set).
- HAWQ, Pivotal HDB
  - Hadoop + SQL
- Redis
  - In memory database (+ snapshots to disk)
  - Optional durability
- Clickhouse
  - Analytical column oriented database

MariaDB

Why use NoSQL

- Faster replication
- Fast and easy key/value access
- Data is often stored in memory
- Note that with similar memory resources you can usually keep SQL data in memory too.
- Can handle unstructured data

- In traditional SQL, one can't easily implement a web store

- Most NoSQL vendors are considering adding SQL support.
  - Like Atlas SQL for MongoDB (not open source)

Why NOT to use NoSQL

- Lot of data duplication
  - Relational database are designed to keep only one copy of a relation (like a customers data)
- Can't (easily) join/combine data
- No standardized language; Hard to move data and applications between systems
- Complete lock in to one system
- Normally few connections to computer languages or existing applications that need a database.
- Allows one to initially ignore solving the database layout problem as one can store data 'as such'
- Initially easier to use, MUCH harder later

- No sustainable scalable business models for most open source NoSQL solutions (as most of them rely on the BSD or Apache licenses)

Problems to overcome with BIG data

- Storage (not normally the big problem anymore)
  - Memory (Very expensive)
  - SSD (Expensive but slower)
  - Hard disk (Inexpensive. Slow when doing random reads, fast on sequential reads. The "tapes of tomorrow")
- Access patterns
  - Some access patterns are good for big data (scanning in parallel) while others are hard
    - Even simple joins can take days on petabyte tables
- Get it up when things fail
  - Recovery; Can take days or weeks
  - Get the caches warm (you need > 90 % hit rate with buffers!)
- Replication and hot standby
  - Must have, but makes big data more expensive

Definition of big data is changing

- When I worked with data warehousing (1986–1993) , big data was (all credit card transactions for one Sweden's oil companies):
- 1M users, 4M transactions / month (30 byte / transaction)
  10 years of data = 1.3G

- Handled by a Sun SparcStation, 25Mz, 32M memory. 2G hard disk

- CPU's are now 700 times faster and have 10,000 times more memory.
  SSD/Hard disk seek speed is 6000 times faster than in 1986

- While machines have got faster, most data is still small (except for social and behavior data) and can easily be handled by one machine.

Why big data solutions are hype

- Most companies will not have as much real data as Facebook, Twitter, Bilibili, TikTok, etc. that needs to be accessed "at once"
- Solutions that they have to use are not applicable for others.
  - In the near future one can run what most companies think is big data on a few machines:
  - Memory now costs about 2100$/T; Most can afford 200G of RAM.
  - SSD are now 50$/T (Read: 560–3100MB/sec 98K IOPS, Write: 520–2250MB/sec write.
  - You can get a 100Tb SSD from Nimbus and a 960Tb from Dell. 16T can easily get bought.

- MariaDB today can easily handle 0.5T of active data. It is only after 1T when one needs to consider analytical databases.
- There are very few users that need more than 0.5T of data.

- The most common database size for big enterprises are **100G** for their largest production databases.
- 90% of queries processes less than **100 MB** of data.
  - See https://motherduck.com/blog/big-data-is-dead/ for details

MariaDB

AI and Databases

MariaDB

AI and databases

- Personally I don't see AI helping optimizing queries inside the database (data is changing constantly and the optimizer is already using statistics and histograms to make decisions).
- Optimizer also has to be very fast and handle a lot of concurrency with a limited amount of memory.

- AI can be used to optimize SQL queries for applications.
- AI can be used to translate natural language to SQL, which is useful for interactive queries but too slow to be used in production for applications.

- When it comes to programming, I would only trust AI to:
  - Initially set things up for a common problem (write a script that does 'easy explanation')
  - Try to find 'simple mistakes/bugs' in code like possible buffer overflows, missing arguments, things that can easily simplified etc.
  - Translating things from one language to another.
  - Do a simple change that can be repeated in many parts of the code.
    - Add a parameter of this type to this function and ensure that all callers are fixed.
    - The resulting code has to be carefully reviewed!

MariaDB

AI and databases

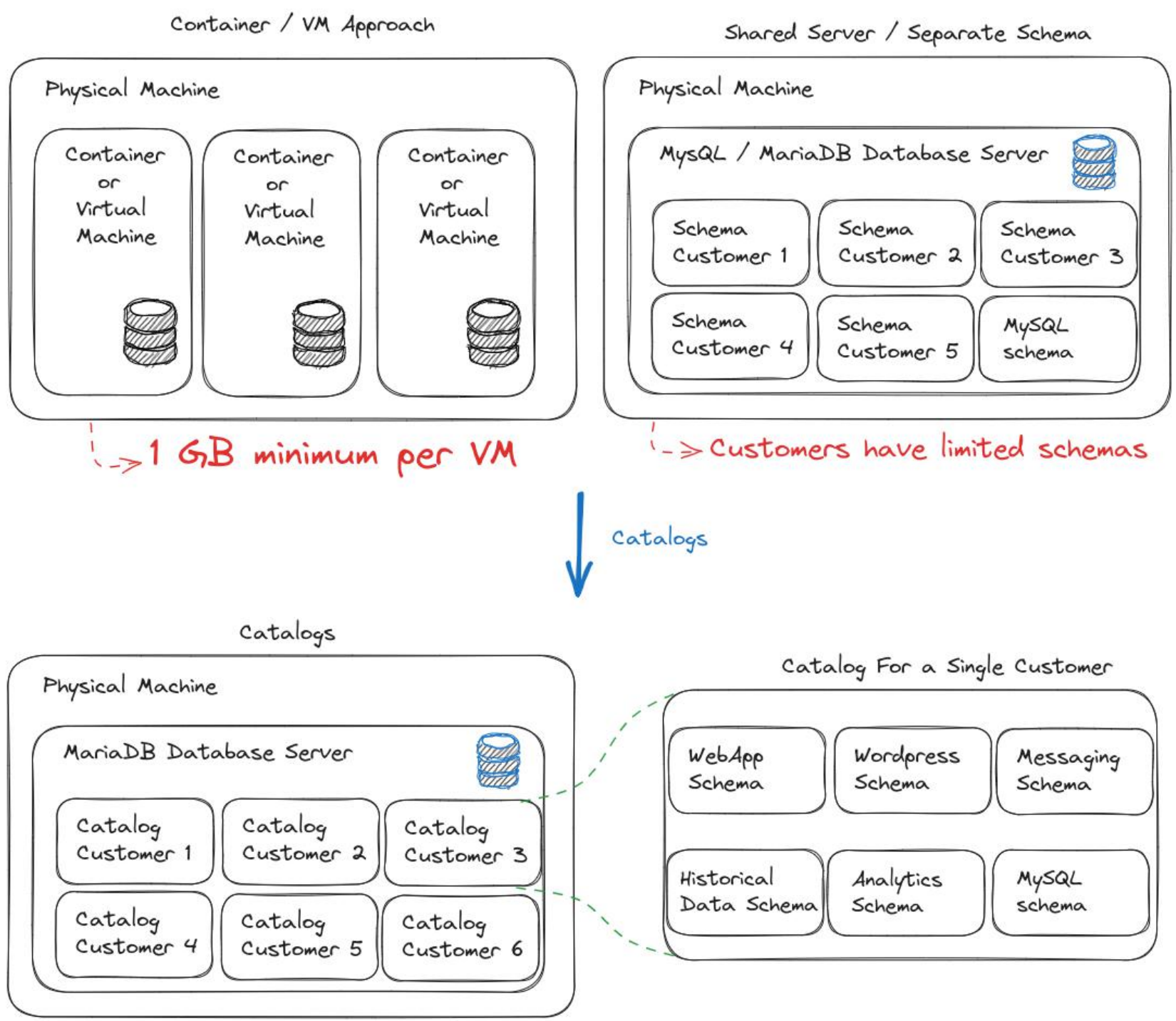
- I would never even try to use AI for doing changes in a complex project, like adding catalogs to MariaDB.
  - There are no similar patterns in existing software that it could be trained on.
  - Too many things in MariaDB are interconnecting in not obvious ways (if one does not know the code)
- If the AI would produce 'hard to understand code', I could not use AI to explain why it did things that way.
  - All MariaDB developers I am working with are experts and can be put on solving very complex problem in their domain. I don't see anyone being replaced by AI during their or their children's lifetimes.

Databases and the Cloud

Database scaling for SaaS providers

- SaaS providers want to optimize the number of customers they can host on their hardware.
- When offering managed services on MySQL/MariaDB, SaaS providers have the following options:
  - Create separate VMs for each customer.
    - Large overhead – costly
    - A typical “idle” MariaDB Server requires ~1GB of memory
    - Best user experience
  - Use shared instance
    - Force user restrictions, only grant limited database access.
      - cPanel has this model, many other database management systems share it
    - Very limited overhead
    - Poor user experience, artificial limitations enforced on users
    - Affected by “noisy neighbour”, hard to track

MariaDB


Container / VM Approach
Physical Machine
Container or Virtual Machine
Container or Virtual Machine
Container or Virtual Machine
1 GB minimum per VM
Shared Server / Separate Schema
Physical Machine
MySQL / MariaDB Database Server
Schema Customer 1
Schema Customer 2
Schema Customer 3
Schema Customer 4
Schema Customer 5
MySQL schema
Customers have limited schemas
Catalogs
Catalogs
Physical Machine
MariaDB Database Server
Catalog Customer 1
Catalog Customer 2
Catalog Customer 3
Catalog Customer 4
Catalog Customer 5
Catalog Customer 6
Catalog For a Single Customer
WebApp Schema
Wordpress Schema
Messaging Schema
Historical Data Schema
Analytics Schema
MySQL schema

New catalogs feature for MariaDB

- Catalogs provide the best of both worlds.
  - Shared instance
  - A catalog looks like a normal MariaDB Server
  - No user limitations
    - Each user has full control over their catalog
    - Root access on the catalog

- Catalogs still have the problem of noisy neighbour
  - However, statistics are now collected for each catalog
  - So it's trivial to detect the problematic catalog
  - Move the problematic user to a bigger machine, increase costs

- Catalogs can also enforce quotas
  - Each catalog have their own configuration file with their own limits
  - Force users to stick to certain performance limitations.
- Sysadmins have control over all catalogs.

MariaDB

Catalogs feature

- Catalogs enable hyper-scaling for SaaS providers.
  - For basic users, one can now host up-to 100x more users on the same machines.

- In MariaDB, this feature is still under development
  - If you would like to steer the roadmap for this, now is the time!
    - Tooling
    - Performance optimizations
    - Specific functionality

Career path for programmers

What I have been working on lately

I am still actively coding!

- Improving the MariaDB optimizer to handle complex queries for big databases
  - This is based on input from MariaDB customers and users
  - I spent almost one year on redoing and improving the cost model of the MariaDB optimizer for MariaDB 11.0.
    - This was what I was working on during my 10 days of Covid quarantine during my last trip to China!
  - Initial test shows that most of the recent queries that has caused problems with MariaDB 10.6 optimizer works very good in MariaDB 11.0!

- The last 6 months I have been working on the multi–tenancy catalogs for MariaDB.
  - This was based on input from several SAS provider at the last 2 CloudFest conferences.

MariaDB

Career path for programmers

- I created MySQL and MariaDB. After the initial setups (where I handled everything), I have always hired people to do management, customers and leading the company so that I can continue to **focus on architecture, development**, and leading the MySQL/MariaDB developer teams.

- A very common question I get in China is what is the best career path for a developer? Should I become a manager or start doing something else?

- Good coders are hard to find!
  - It takes a long time for a coder to reach their peak (8–10) years of experience.
  - Good programmers are valuable and can produce quality code for LONG time (>> 70 years)

- Don't waste a good coder by "promoting" him to a manager.
  - A good coder is not necessarily a good manager.
  - Promote them by giving them more responsibility:
    - Architecture – design bigger and more complex systems
    - Help others to work on their code (but not manage!)

- My advice to managers:
  - **Do not** follow the "**Peter Principle**" to promote people until they reach their level of **incompetence**!
  - The expected salary raise/year in China is not suitable for long term programmers (35 year rule!)

Thank you