Abstract + notes…

Microservices are a useful paradigm, and the supposed advantages are often cited.
But there are a few things to consider when microservices need to interact with a “traditional” database, RDBMS or data-store.

In some cases, you can even have your RDBMS (Oracle, PostgreSQL or mySQL)
take up the role of a microservice, and achieve remarkable efficiency.

We will provide a few use-cases and examples where it may be interesting to use the DB as the main component of the microservice.

After the presentation, the audience will have a better idea of the
Advantages, and hopefully some ideas to to avoid the Disadvantages of Microservices.
And in particular use cases, may choose to use the Database as the base for a microservice.

BASE – ACID
SOLID principles – including Liskov – substitution: this is your ”upgrade path”

Security: yes, another nightmare.

Notes: russia: Narva, Daugapils, only 800km from Moscow

Next slide is screen+intro,
Put abstract here, in case you need it.

Microservices and Databases
How to do it Smartly

Piet de Visser

The Simple Oracle DBA

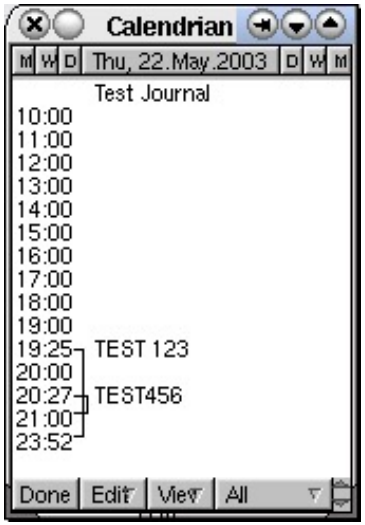
Combining Microservices and Databases can work,
and I have a few Ideas on how to do that Efficiently.

Piet de Visser - **PDVBV**

Microservices and Databases. A few ideas on how they can work, My real life is riding a motorcycle: Closest I've been to Moscow was Narva to Daugapils, only 750km.

Agenda (approx 45 minutes)

History	**(nah, KISS…)**
Microservices	**(decoupled…)**
State / Less / #SmartDB…	**(huh?)**
Some Ideas	**(ok, but…)**
PG	**(PL/pgSQL)**
Oracle	**(PL/SQL)**
YugaDB, CockroachDB, SQLite	**(Chcek, Test, verify…)**
10 min Discussion	**(Do Challenge!)**



Agenda. What you need to know.
You can still leave..

Microservices – Great Concept.

Microservices, are about:

- Decoupling (modular, SOLID …)

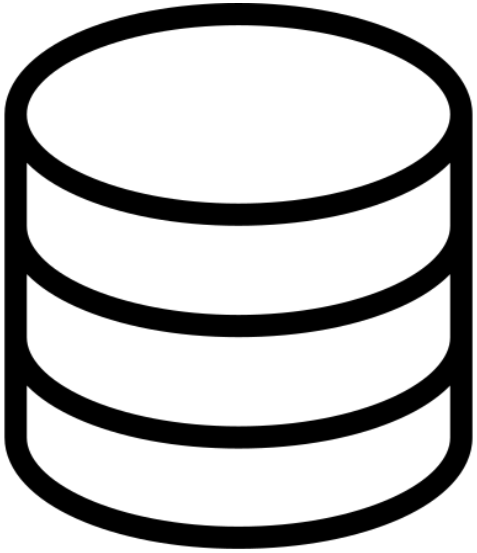
But a Microservice (90%) needs Data, or ”State”

Q+D Microservices de-coupling, scalable..(SingleTask, OpenExt, Liskovsubst, InterfaceSegr, Dep-inv)
But what about data ? - just about any “program” needs data… The Elephant in the Room!

Most MicroServices : a URL…

Microservice = URL-call, http(s), “in a cloud”
Microservice mostly GET / POST / PUT etc...

Hence some App-component + …
Probably some Database or datastore.



Sequence of activities:

- url calls microservice app…
- [app calls database(s)…
- Database(s) lookup + return…]
- Return result… (avoid 400 or 500 errors)

Worst case, there is a sequence of calls between components happening.
That takes time + resources. (Blame… every1 knows Databases are Slow…) – Ideally minimize the db-calls

State @ MicroServices – 3 x Data.

1 – Data is null, Stand alone program …

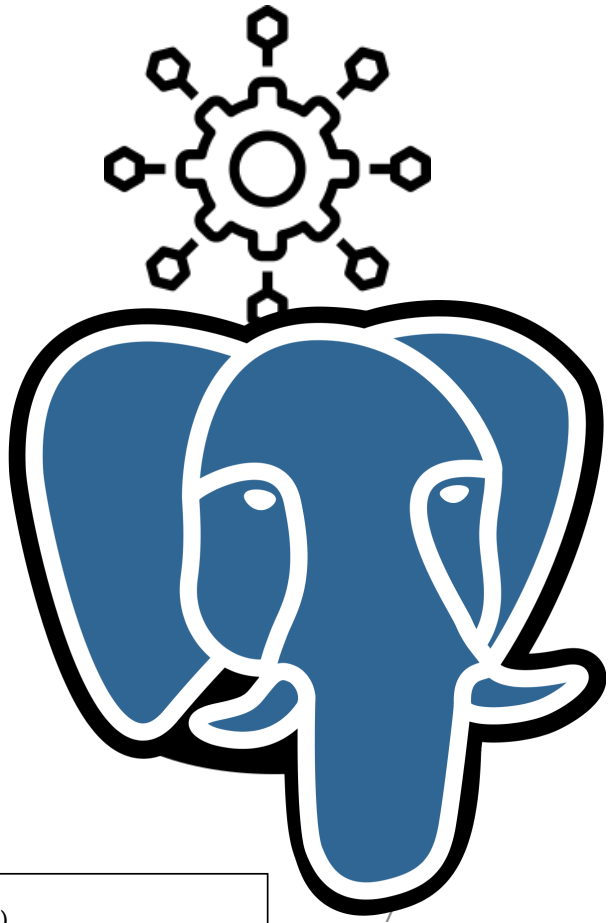
- Doesnt need State (at all ?)
- E.g. format-string, calculate Farh/Celsius/Kelvin.
- Simple, but Rare !!

2 - R/O, "StateLess", Read-only data:

- Dont "recieve" data, store + return some data,
- e.g. Catalog, Lookups, LoVs..
- May need a Database, a "read-store"

3 - StateFull: Needs to Store Data – ACID or BASE…

- Needs a Database or "Store"
- E.g. stock-keeping…
- Needs Updates!



3 kinds of "microservices". We Establish that most MicSrv nees some data, database.
BASE: Basic Availablility, Soft-State, Eventual consistent. (joke: will try to save the data..)

1. Stand-alone MicroServices – no Data.

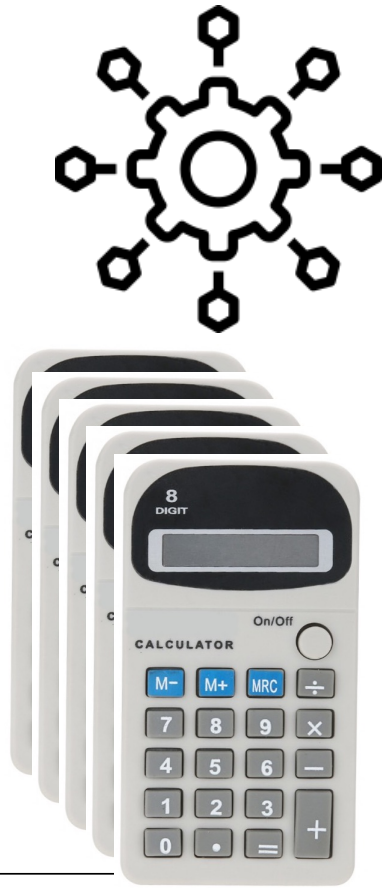
Just Code Executing…

- **Celsius -> Kelvin**
- **Sprintf (…); Format a string**
- **Compress a file…**

No data, no extra components.
Often small program: easy to containerize, k8s…
Deploy and Scale out on any platform; #Cloud.

If this is your "challenge", Lucky You.

You Have a Microservice.
You are Buzzword compliant!



A program that just "executes", without need for other data, a win-DLL ?
Fine. You built a calculator, and you can distribute 1000s of them with no problem

2. Stateless – R/O data in MicroService.

Your program needs “some Data”

Zip-code lookups

Catalog-data, possibly images (static!).

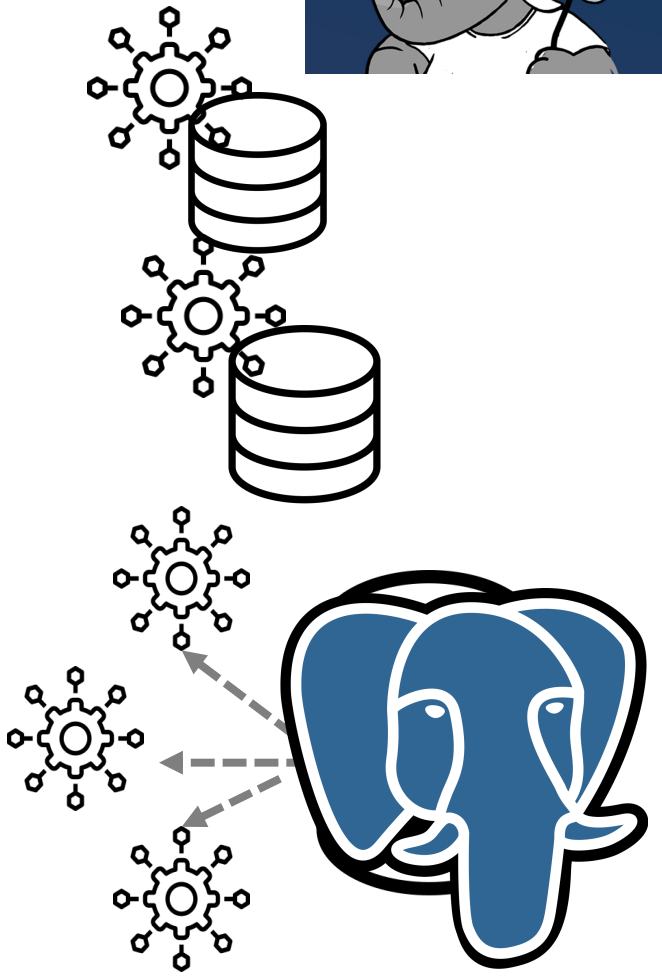
Your Choices are: “to Include“ or “to Call-out”

A) Include - your data “in the micro service”

- Timeliness of your data…(how old is your set)
- Fatter “microservice”; Slower startup..

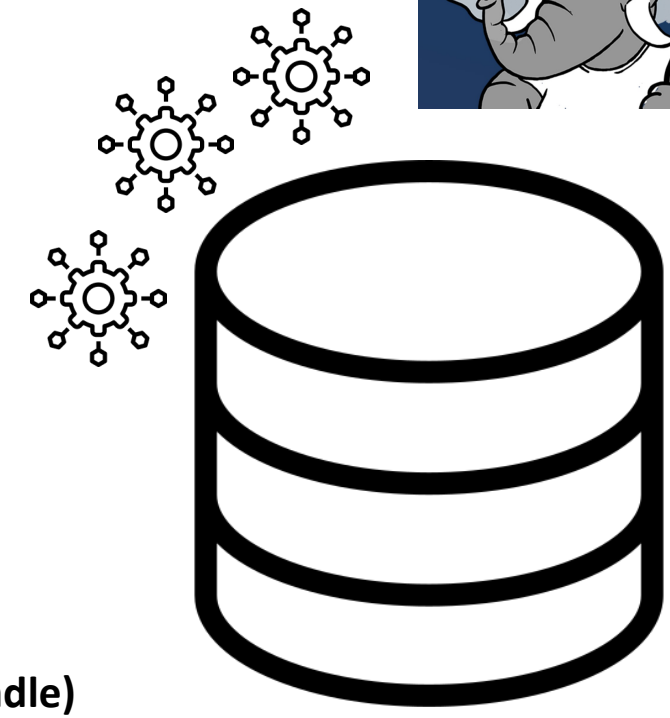
B) Call-out - to find data, somewhere.

- Effort + Time, Chattyness…
- Bottleneck: (calls to -) the Database ?
- “Coupled”, less Preferable, IMHO…)



When you need some “static” data, you can include, or call out.
Whichever fits your need/risk best, But notice the “microservice” concept is beginning to itch.

3. StateFull, service == data(base)



Your program needs to "Keep" data..

Some SPOT, a "System of Record"

Order-taking, stock keeping, payments…

You have a limited choice:

Need (a call to) some "storage", ACID.

Considerations:

- **Data-loss** **(what can you afford to...)**
- **Response-times** **(what can you do in 10ms…)**
- **Capacity** **(how much traffic can you handle)**
- **Error Handling** **(the fail-whale…)**

You need a Database…

When your "service" needs to Store data, you suddenly realize you need a database,
or something very much like it – in some cases "object storage" may be ok, but mostly, … an RDBMS..

Simplify + Minimize…

Engineering Rule of Thumb:

- minimize nr of components

- minimize nr of different components.

In IT + Database land…

- **Minimize nr of Microservices ?**
- **Use 1 database (hm, monolith…)**
- **Use same database (hm..)**

Microservices + Database;

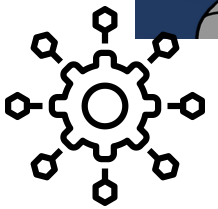
Simple, SOLID, approach…?

MAA : Mandtory Automotive-analogy, : less components and more similar componetns
Benefit maintenance… : make it Simple, Reliable, Fixable. Than deploy in 1000s…

Microservices – Concept again; add “data”

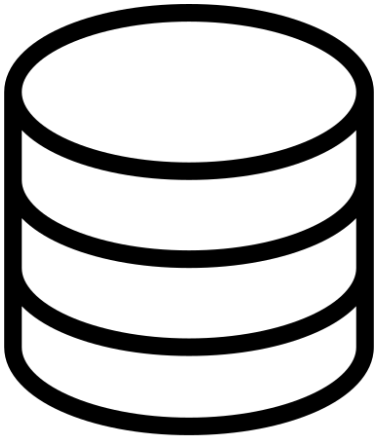
Microservices

- **Decoupling (stand-alone if possible…)**
- **Scalable (e.g. start Fast, start Many…)**



Fit data (read-only) in a container ?

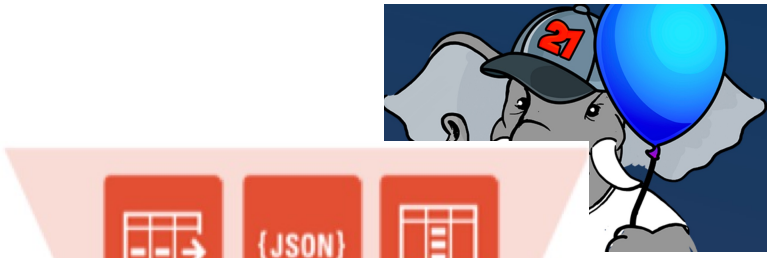
- **In code: *.csv, associative-arrays, …**
- **In code: SQL-Lite…**
- **Postgres (lightweight)**
- **Oracle[XE] (… “Fat”)**
- **#SmartDB #BringCode2Data …**



You Choose…

Vlad: Every mSrvce needs a database.. = Good Idea!
No you choose..: hardcoded, lightweight DB, Full-fledgedDB, #Smartdb: You can put you code “inside”

Two Keywords



#SmartDB

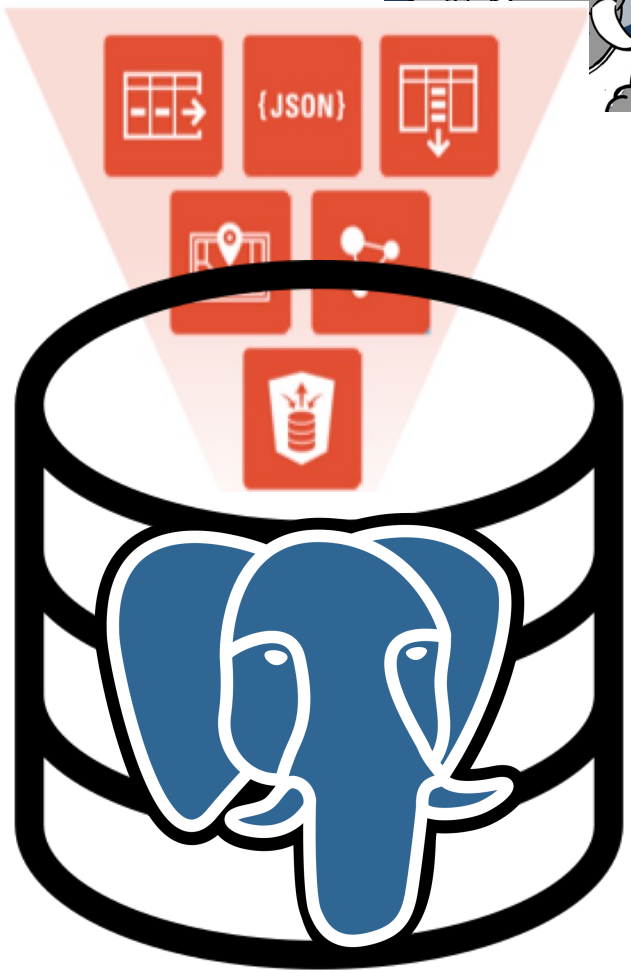
- SQL; Views, PL/pgSQL.
- “do work inside” – close to data
- #BringCode2Data

#ConvergedDB

- “Everything” goes in your database..
- Tables, JSON-obj, Docs, GIS, Graphs…

Keep these in mind..

(and beware of the Monolith….)



Explain #SmartDB and #ConvergeDB
No you choose..: hardcoded, lightweight DB, Full-fledgedDB, #Smartdb: You can even put you code “inside”

R/O data? - I'll Argue: Database!

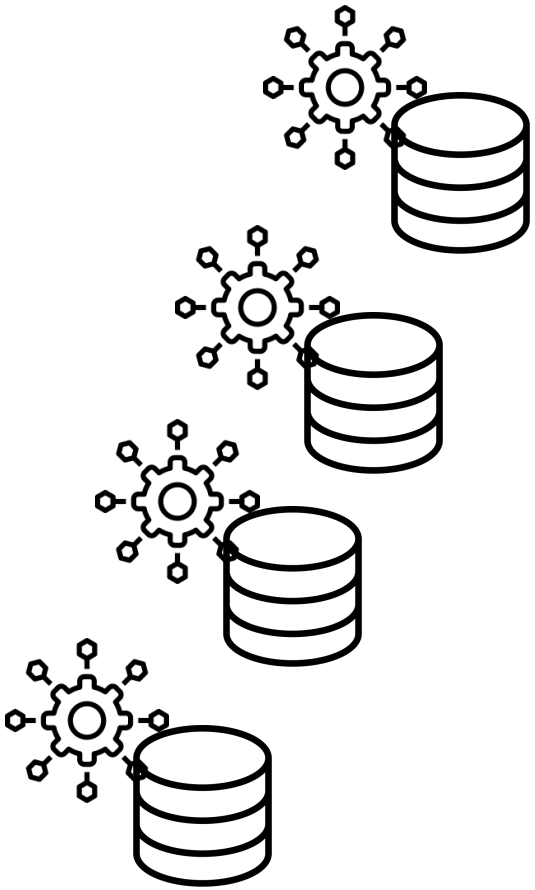
Put a (smart) Database in your Microservice.

SQLite	**(fits everywhere…)**
Consider PLpgSQL	**(fits in a small container)**
Consider PL/SQL	**(fits in Large container..)**
PostgREST ..	**(fits in …)**

Reduce calls To+From other components.

Reduce context switches.

Google: #smartDB or #ConvergedDB…



How many know that even SQLite supports views, joins, and stored-functions ?
Pl-pg-sql is the sweet spot: functional + small in size… Oracle/ORDS is a bit "Fat"

Poll time!

I Like SQLite

I Avoid SQLite

What is SQLite ?

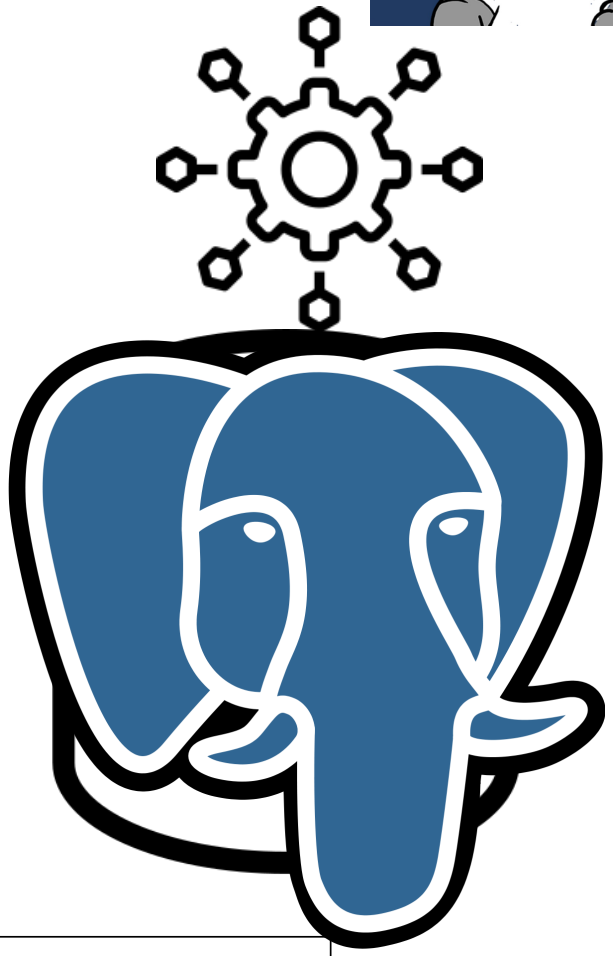
The Interactive part…See how knows.. (if time) –
Those who voted "like" will have to create a stored-function in SQLite… hehehe.

Recap: State in MicroServices…

1 - No State, Stand alone program …
- Easy…

2 - R/O, “StateLess”, Read-only data:
- Include data (or call out…), your choice.

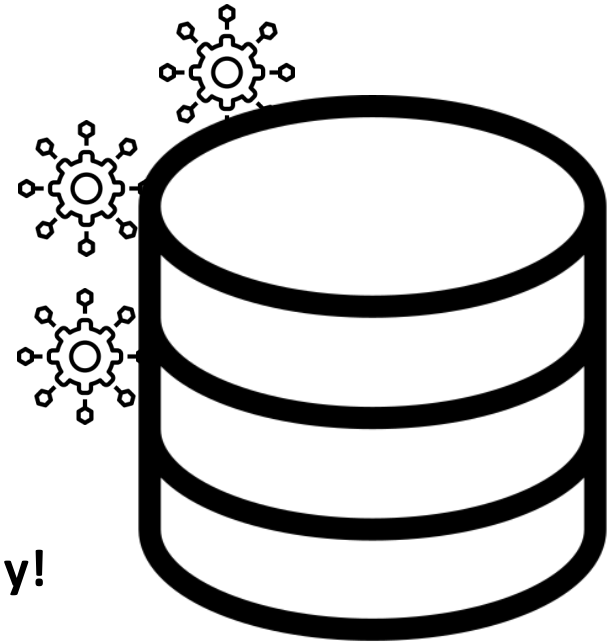
3 - StateFull: Need to Store Data
ACID (not just BASE…)
The Biggie…



Recap the 3 cases: with or without state..
Basic Availabilty, Soft-state, Eventually consistent… not always good enough..

ACID data? - Definitely: #SmartDB !

Put a (smart) Database behind your Microservice.



Reduce calls To+From database.

Have an “Overload-prevention” mechanism.

Everything will “scale out”, except your DB!

Again: #smartDB or #ConvergedDB…

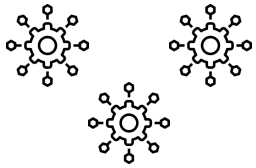
Stored Procedures are often the most Efficient way!

But.. You Choose…

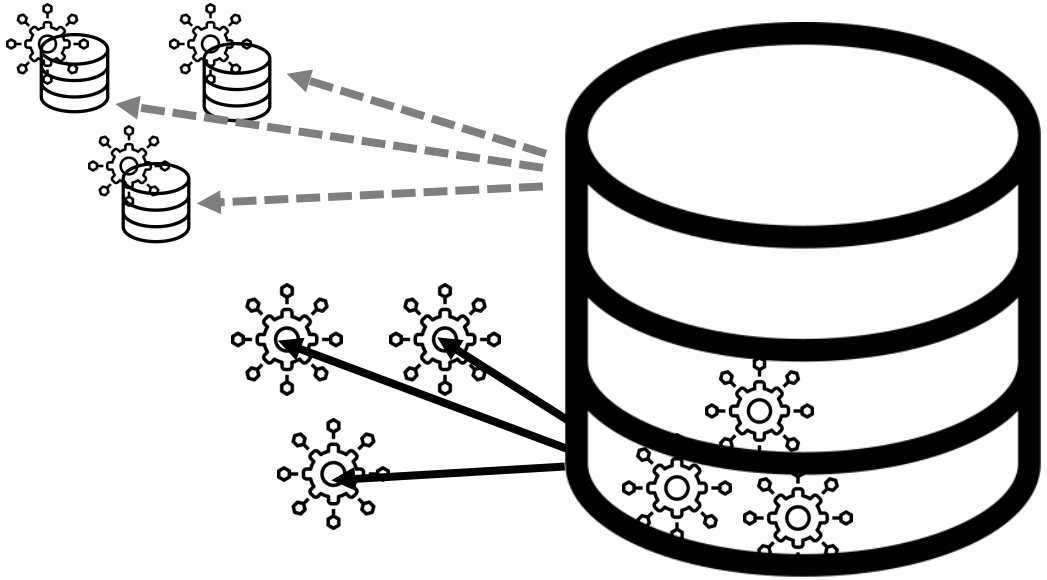
Storing, Saving , Committing data: needs a Database (incl replicas, backups etc..)
Swarms, Catlle, .. Your DB is your Pet!

The Nice Picture …

Stand Alone mSvc
Fine..



R/O data
Subsets ?
#SmartDB



ACID-processing

Arbitrary classification: Stand-Alone, R/O en ACID.. Each with their own approach, solution..
Swarm, Cattle and Pets. But DB centric view.. I would Recommend you Call Stored Procs…

Databases, Loose ends…

R/O, Stateless, data(base):

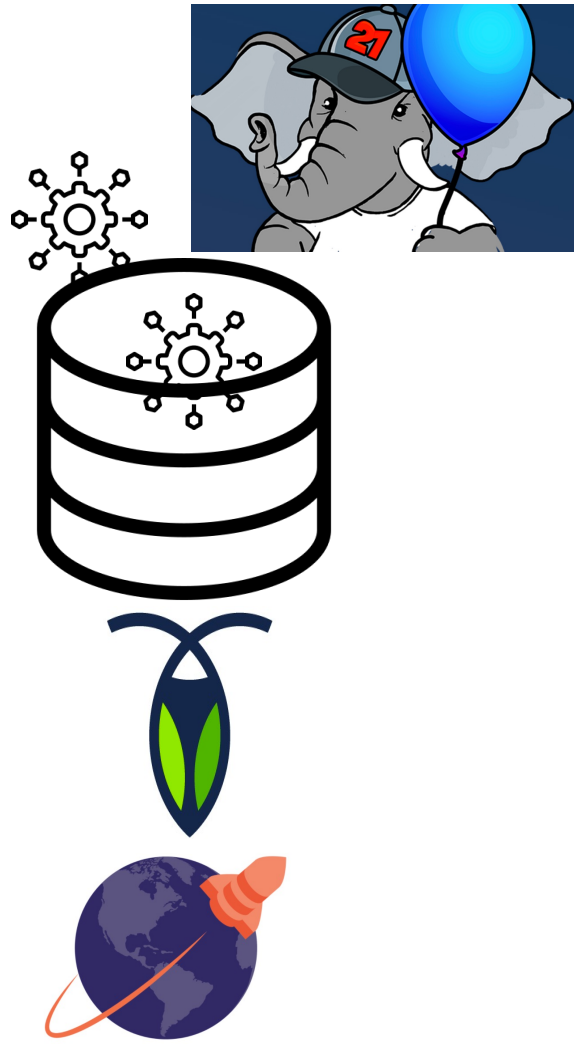
- put ”Logic” (joins, code) in your Data(base).
- Distribute Subsets of data? (CQRS…)

ACID datastore:

- limit nr of connections – Poolsize!
- Protect against Overload in general

Consider NewSQL (YugaByte, CockroachDB….)

Monitor; Measure usage, Spot Trends, Evaluate!



Some generic things we found…
Swarms, Catlle, .. Your DB is your Pet!

Main Messages – repeat, conclude…

When "State" is important:

- **Consider processing in or close to Database**
- **minimize calls, reduce chattyness**
- **Simplify Maintenance.. (less components)**

- **=> Stored Procedures,**
  - **called by thin, light, app-components.**

- **Mitigate the problems of "monolith"**
  - **Throttle App-level or DB-level, prevent overload.**
  - **Consider read-copies for GET-only calls**

Ask for watch, tell time..
CQRS, but only if you have to

Analogy…

#SmartDB

#ConvergeDB

#BringCode2Data

Most of us could not carry the Dolmen.. That is why we bring the hammer + chisel to the stone..
(and the marketing hashtags…)

Interesting times ahead

- Curious to see how this will develop
- NewSQL : Supersize, Distributed DB ?
- Many New systems...
  - Do you agree with my ideas ?
- Discuss
  - What are Mircroservices like next year ?

Watch this space. Lots of interesting new features + tricks.
Would love to test some of those for Real… But. Beware of over-engineering.

Don’t Take my word for it…

Google and RTFMs

Simplicity

- In case of doubt: Simplify!

[Simple Oracle Dba . Blogspot . com]

[@pdevisser] (twitter)

Goethe _____ (simplicity)

Majority of time I have been WRONG.
So go see for yourself – but don’t complicate life.

Quick Q & A (3 min) 3 .. 2 .. 1 .. Zero

- **Questions ?**
- **Reactions ?**
- **Experiences from the audience ?**
- **@pdevisser (twitter..)**

Question and Answer time. Discussion welcome (what about that Razor?)
Teach me something: Tell me where you do NOT AGREE.

Blank

End - This slide intentionally left Blank…

Somehow, the unit and concidiont of “database” got determined by the vendor, not by the customer
And vendor now has us y the balls