

# Deep Learning (m)eats Databases

(shortened)

Jens Dittrich

Saarland University  
Saarland Informatics Campus  
d:AI:mond.ai

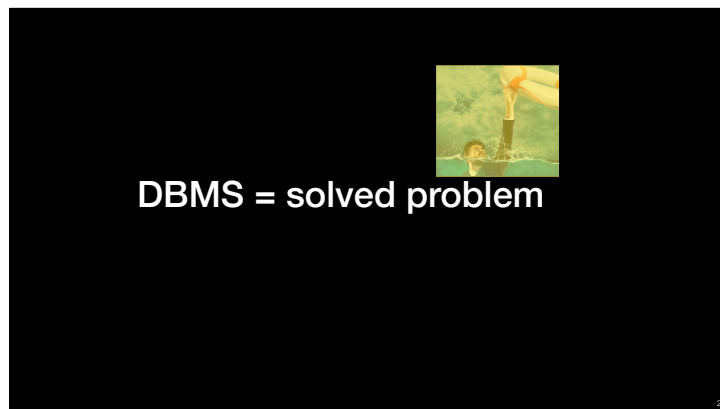
Note:

this is an annotated and shortened version of my VLDB 2017 keynote. In the following, I will talk a bit about the state of the database community.

I slightly enhanced this version here and there over the original presentation.

Let's start:

1



Database Management Systems is a solved problem.

Well, why is that? The reason is quite simple.

[Copyright [istockphoto.com](https://www.istockphoto.com) alphaspirt]

2

## > 40 years of DB research

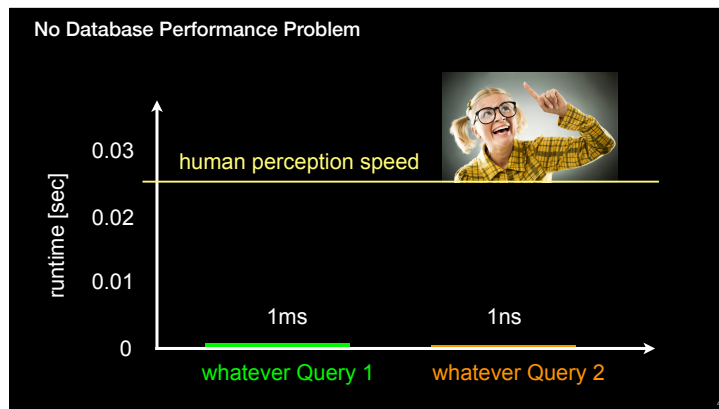
In the past >40 years we did a fantastic job as a community solving this problem.

We developed:

- fantastic algorithms
- great systems
- clever query processing and storage strategies
- a lot of brilliant stuff.

And due to these inventions and ideas DBMSs have become fast — really really fast.

3



DBMS have become so fast that it almost does not matter anymore which techniques and algorithms you use: the query will execute in a blink of an eye.

In other words, if you measure query performance in terms of human perception speed, which is ~25ms, whatever you do, the query will execute like that [snap fingers].

Just to give you an idea:

- when you look at TPC-C: we are currently able to execute ~half a million transactions per second ... on a single thread and partition, i.e. when executed in HStore mode; if you add concurrency control, you will go down to a couple of hundred thousand transactions/second; no one on this planet needs this performance, not even amazon
- when it comes to simple index-lookups, e.g., in a hash table, we are currently at 20 million operations per second. Again: this is single-threaded!

<https://infosys.uni-saarland.de/publications/p249-richter.pdf>

4

database performance = solved  
problem

Therefore: database performance is a solved problem.

And this also has a lot to do with hardware. In the good old days you required clever algorithms to process your query.

Today, in many scenarios, you can do stupid stuff: hardware will handle it, a clever algorithm won't make a difference.

This is one of the reasons why Python and other scripting languages as well as NoSQL systems, where some of these systems have very simplistic query optimisers, are often just good enough.

Hardware is eating the need for clever algorithms.

5

enter "Big Data"

how large is "big"?

enter „big data“

What is „big data“? Is that the same as „large data“?

Well actually „big data“ means more than „large data“! But that would be the topic of another talk.

Many people interpret „big data“ as „some large dataset“ and that is what I will do in the following.

6



For the database community „big data“ saves as a lifebelt. For small datasets no-one needs clever database algorithms and systems. You can do (almost) anything do process that data.

With big data, however, you need clever stuff again. So we can write a lot of papers about „databases and big data“.

However, when I look at these papers, I get the feeling that there are really not so many techniques that you need to understand in order to scale a „big data“ problem.

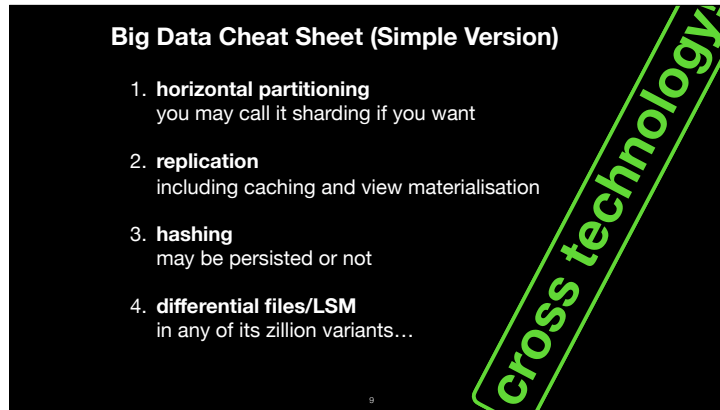
[Copyright [istockphoto.com](https://www.istockphoto.com) alphaspirt]

7

## 4 techniques to solve Big Data problems:

Actually there are really only four techniques that matter in order to scale “big data“:

8

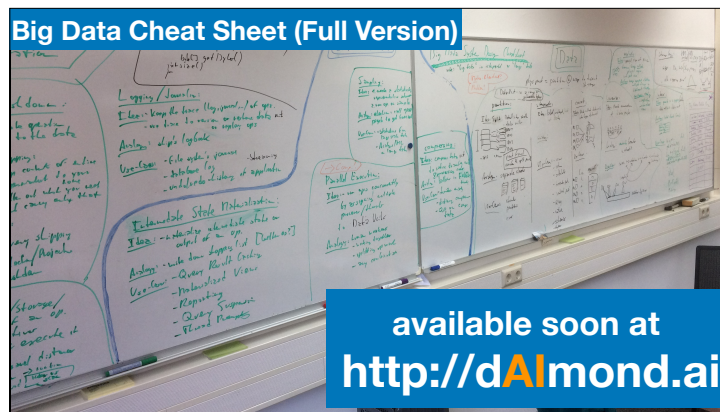


1. horizontal partitioning you may call if sharding if you want
2. replication, including caching and view materialisation; this includes caching across different storage layers
3. some sort of distributed hash table, may be persisted
4. differential files and indexing in any of its zillion variants (reinvented several times each year)

Notice that all of this is totally cross technology. I am not saying: use system X or Y or programming language Z. These four techniques are principal patterns (similar to software design patterns). You can use them with any system, and all decent systems are built on this.

Well actually, of course this is a bit oversimplified, maybe it is not only four techniques but rather more like a dozen or so. So I have been working on collecting these techniques in the past months. My idea is to collect them in a „Big Data Cheat Cheat“ poster.

9



The good news is: I already have it worked out on two whiteboards in my office.

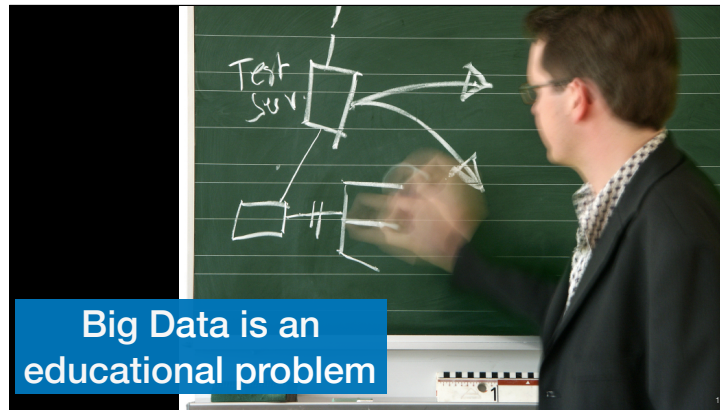
The bad news: it is not yet ready for download. Stay tuned.

This poster contains the central ingredients you will ever need to „cook“ a big data system. Obviously, just knowing the ingredients won't make you a great cook: you still need a lot of knowledge, skills, and experience.

In the end, designing a great system is an art, much like cooking.

However, having a clear understanding about the important ingredients helps you to become a better cook!

[\[http://daimond.ai\]](http://daimond.ai)



Anyways, to conclude, I really see „big data“ as an educational problem.

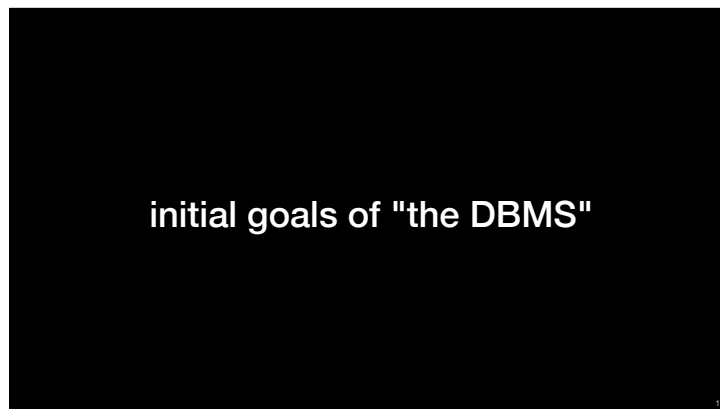
If you have a performance problem, you better make sure that you know the ingredients and use them skilfully.

That will solve >95% of the world's "big data" scaling problems.

If you can't scale your stuff, I would like to know more about your problem.

[© iStock.com mammamaart]

11



Back to our "DBMS is solved" claim.

Let's revisit the initial goals of "the DBMS":

12

## the 70ies through 90ies: databases as THE analytical tool

13

In the 70ies through 90ies: databases were THE analytical tool. Data was in a central place: the “database“. That was a great idea. And it was a good fit to the hardware available those days, i.e. mainframes and other expensive servers.

However, ...

13

## That story is over.

14

That story is over.

In the past two and a half decades and with the advent of personal computing, data has become much more distributed. More and more data is analysed outside the DBMS. And data analysis is done using a large zoo of tools. The DBMS is just one out of many possible tools.

14

**OLAP**  
-> **Big Data Analytics**  
-> **Data Science**

15

For DBMS we witnessed this development with OLAP and data warehousing:  
those environments gradually „opened up“ to

- „big data analytics“: this translates to „something with Hadoop and/or Spark/Flink“, and then:
- „data science“: this translates to „something with Python and/or R“.

These days, everyone is talking about data science.

The problem with data science, however, is the following:

15



Data Science does not need databases.

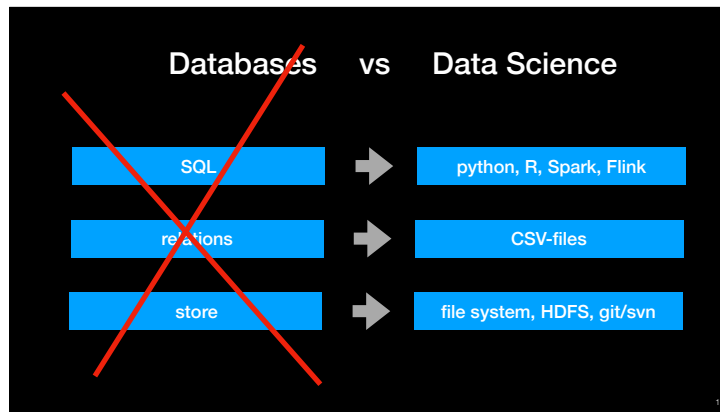
This is bad news for us.

But why is that the case?

[© iStock.com gbrundin]

16





Well, when you look at a typical data science workflow, it looks as follows:

In a database, we have a store, the relational model, and SQL. [Yes, this is a bit simplified.]

In data science, people don't put their data into a „store“. For them a store is a file system, HDFS, git/svn.

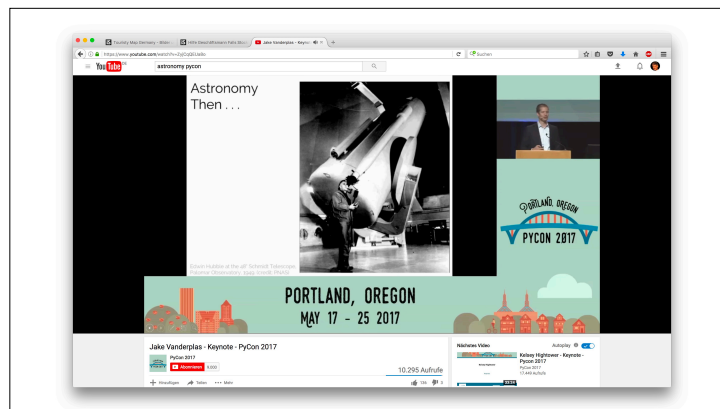
In data science, people don't shoehorn their data into relations. They leave it in CSV-files.

In data science, people don't use SQL, they rather use python, R, Spark and Flink.

So, you don't need a database to do data science.

Well, and if you know think: “hey, but what if the data is big?“. Then, I would like to invite you to check out the following video:

17



Jake Vanderplas is an astronomer. He gave a very interesting keynote at PyCon 2017 explaining how all the analysis done in astronomy using a fantastic array of Python tools and libraries.

Astronomy is “big data“. And even there Python is eating up the interesting bits.

Of course underneath there are still databases, e.g. Sloan Digital Sky Survey, but the cool analysis now happens outside the DBMS using lightweight tools — not inside.

<https://www.youtube.com/watch?v=ZyjCqQEUA8o>

18

=> DBMS is the wrong tool

19

So, in summary: the DBMS is the wrong tool.

19



Our community is under attack by various “data science“-tools. The life belt won’t help us.

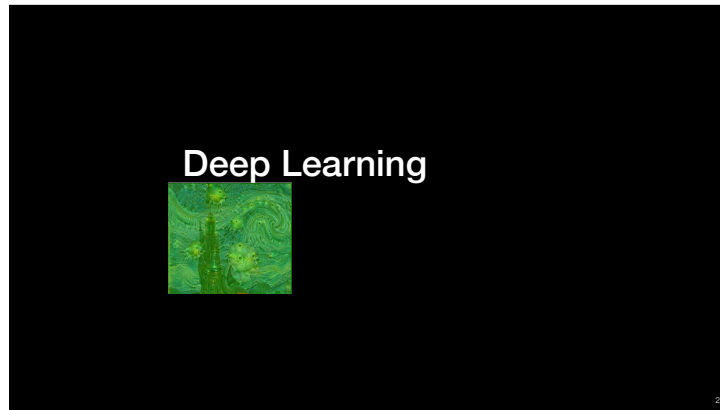
And it is even worse.

Because, there is also...

[© iStock.com lexaarts]

[© iStock.com alphaspirt]

20



...Deep Learning.

This is currently the hottest thing in IT.

How is the database community positioned w.r.t. deep learning?

21



It looks something like this. There is a tsunami coming.

[© iStock.com lexaarts]  
[© iStock.com alphaspirt]  
[© iStock.com pijama61]

22

# Hype?

23

Isn't Deep Learning just another hype?

I mean, we database people are real experts in hypes!

We had so many hypes including: Streaming, XML, NoSQL, MapReduce, more recently Big Data

Many people say: Deep Learning is just another hype, it will go away.

23

# Deep Learning vs Big Data

24

Well, let's check.

Let's measure Deep Learning and „Big Data“ (as a synonym for large data management, i.e. let's assume this is just us!) by their (and our) achievements.

24

## Deep Learning vs Big Data

What are the achievements of deep learning?

25



Go

Well, you probably all heard about Go. DeepMind developed various systems that beat the strongest Go-players on the planet. Go had been considered to complex to be solved by computers. That has changed with deep learning.

The most recent version of their system only knows the rules of Go. Everything else it finds out by itself.

This is amazing.

26



### Style transfer

Deep learning allows you to learn the style of a painter and apply it to any photograph. The result is a picture that looks as if it was painted by the artist.

27



Here is another example:

upper left: photograph

other images: generated artwork in different artistic styles

[[https://commons.wikimedia.org/wiki/File:Tuebingen\\_Neckarfront.jpg](https://commons.wikimedia.org/wiki/File:Tuebingen_Neckarfront.jpg) CC]

[A Neural Algorithm of Artistic Style <https://arxiv.org/abs/1508.06576>]

28










Google published a system where the network would over-emphasize certain aspects of the image allowing you to generate nightmarish pictures.

[<https://github.com/google/deepdream>]

29

### Generative Adversarial Networks (GAN)

Text description	This bird is red and brown in color, with a stubby beak.	The bird is short and stubby with yellow on its body	A bird with a medium orange bill white body gray wings and webbed feet	This small black bird has a short, slightly curved bill and long legs	A small bird with varying shades of brown with white under the eyes	A small yellow bird with a black crown and a short black pointed beak.	This small bird has a white breast, light grey head, and black wings and tail
256x256 StackGAN							

[Han Zhang et al, <https://arxiv.org/abs/1612.03242>]

This is my personal favorite.

The hottest subtopic within deep learning is generative adversarial networks (GANs).

And this one is an example of a variant of a GAN which does the following:

Given some textual description, the network generates an image that corresponds to that textual description.

This means, the GAN does not retrieve or look up an image that was used during the training phase, but rather synthesises an artificial image corresponding to the textual description. Maybe you want to read the last two sentences again.

This is insane.

[<https://arxiv.org/abs/1612.03242>]

30

## Deep Learning vs Big Data

31

31

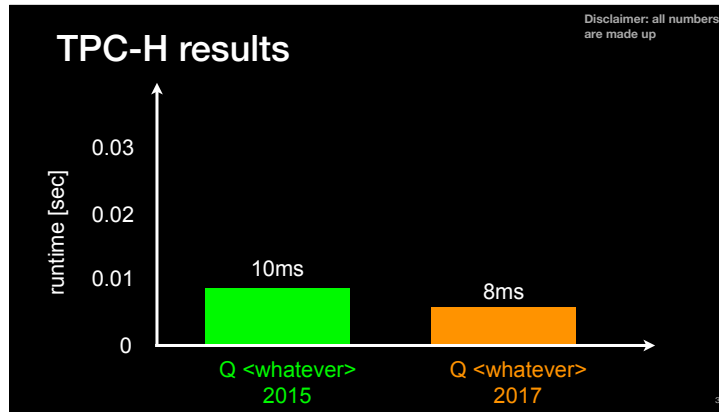
## Deep Learning vs Big Data

32

So in comparison to deep learning, what did we achieve?

32

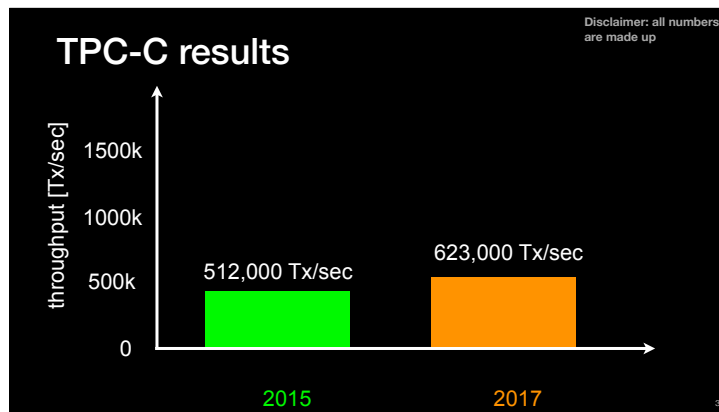




Well, yeah, we made some analytical query a bit faster, maybe from 10ms in 2015 to 8ms in 2017.

Great, isn't it?

33



And, yeah, we increased transaction throughput, maybe from 512,000 transactions per second in 2015 to 623,000 transactions per second in 2017.

This is so cool.

[Note: number are made up, but represent the "ballpark"]

34

## Deep Learning Myths:

I noticed that when I talk about deep learning with other people I am often confronted with myths about deep learning.

For instance:

35

**Myth:**  
**"Deep Learning is not new"**

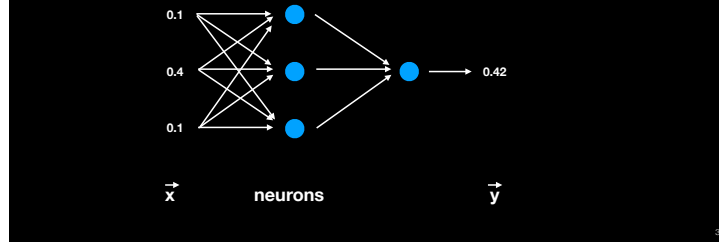
**Myth:**  
"Deep Learning is not new"

This myth kind of funny from the point of view of the DB community. I mean we also reinvent things here and there, I heard...

It is true that the first neural networks were designed in the 40ies and 50ies.

36

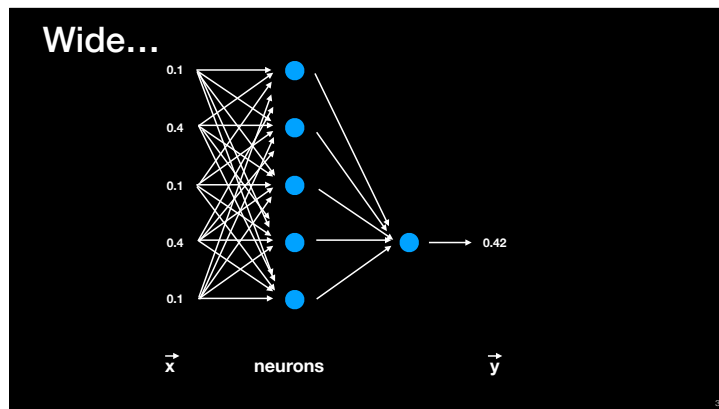
## Simple Perceptron



It started with a simple perceptron. This is called a feed forward network. The scalars of an input vector  $x$  are fed into neurons which add up the different scalar values and apply an activation function (sigmoid, relu, ...). The output is then sent to the next layer of neurons which eventually produce a final output: the vector  $y$  in this case, It has a single scalar value in this case (could be more).

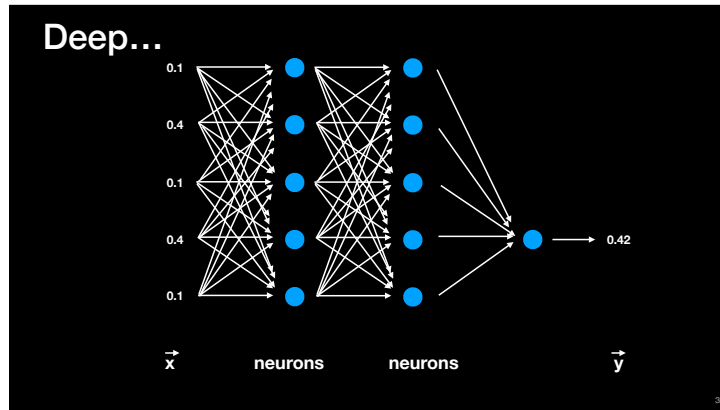
37

## Wide...



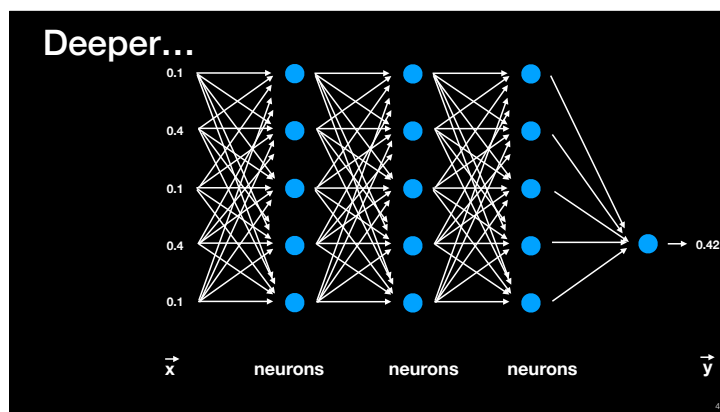
We can make these neural networks wider.

38



And/or add more layers, we make them deeper...

39



...and deeper.

Hence the name “deep learning“: it refers to networks having dozens of layers.

This has become possible in recent years due to powerful GPUs (or even specialized hardware like TPUs or dedicated GPUs like Volta).

So this all looks like a game of creating wide and deep perceptrons, but there is more.

Many specialized new neural network architectures have been developed.

40

## CNNs ~ Self-learned filters

41

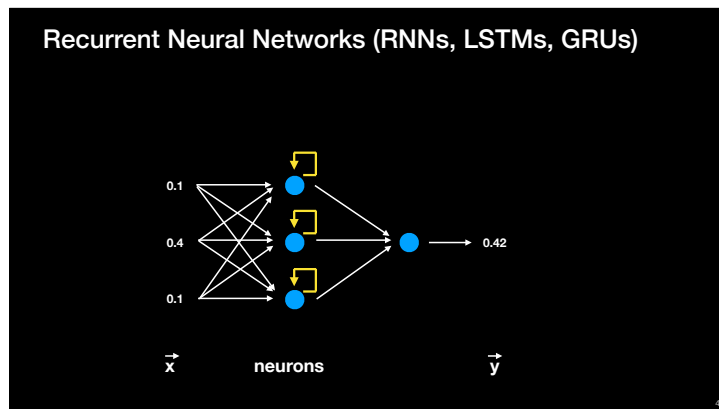
A very prominent example is convolutional neural networks (CNN).

In database lingo, you could basically consider this a rolling window filter combined with an aggregation.

The cool thing is: the filter conditions are not user-specified but `_learned by the system_`.

This is crucial for image classification and works super-well. Nevertheless it has issues, e.g. [<https://medium.com/@pechyonkin/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b>]

41



Another important invention is Recurrent Neural Networks (RNNs) which exists in different variants like LSTM and GRU.

RNNs are crucial for language translation on text and audio, but also video analysis.

So in summary,

yes: some of the techniques in deep learning are pretty old.

but: there is so much going on in that field, currently, many exciting inventions happen every week.

42

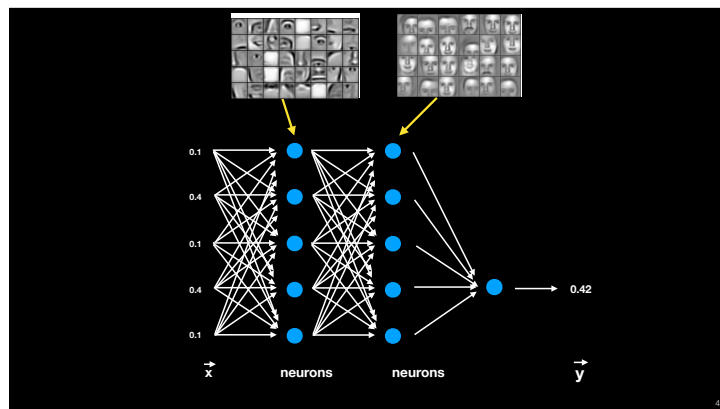
# Myth: "Deep Learning is a black box."

43

Myth:  
"Deep Learning is a black box."

yes, most of it, but: there is more and more understanding of what is going on.

43



Here is an example. In networks used for image classification it is possible to compute for any particular neuron the input required to maximize the neuron's output (this is actually an optimization problem).

Like that, for each neuron one can visualize the input image where it "fires" most.

The resulting maps show that lower layers of a network specialize on simple visual features like edges, lines, etc.

Higher layers specialize on more complex visual features like mouths, eyes, noses, etc.

If you add more layers neurons may specialize on types of faces or faces of a particular person.

This is just one direction opening up to black box of deep learning. There is much more work going on. Actually being able to explain deep learning is an important research topic in that field. This also has implications for accountability of systems

44

**Myth:**  
**"you need 'big' data to do deep learning"**

45

Myth:  
"you need 'big' data to do deep learning"

not really, it depends:  
if it is only a few hundred or thousand records: Sure that does not work too well,  
but even with a few hundred thousand or million records you can do amazing stuff.

You do not need Facebook or Google-sized datasets in order to do Deep Learning!

45

**Myth:**  
**"You need a Ph.D. in Machine Learning to do Deep Learning"**

46

Myth:  
"You need a Ph.D. in Machine Learning to do Deep Learning"

sorry, but absolutely no!

This is one of the big changes in machine learning and deep learning happening right now: the technology becomes so much more accessible and usable to laymen.

46

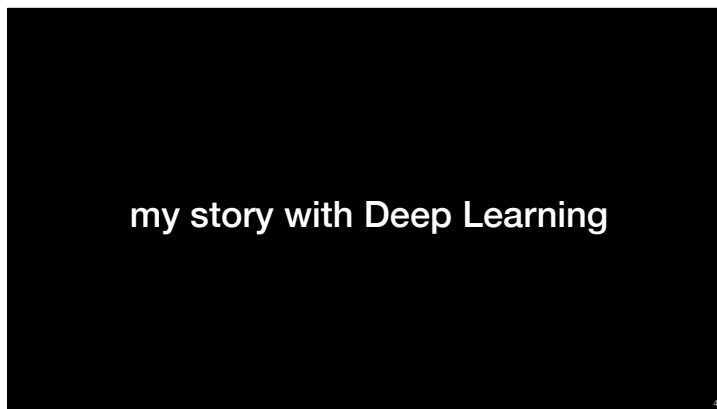


There are tons of resources on the web, almost no math required, most of it is very hands-on and engineering style, feels a lot like db system design!

Here are just two examples of good resources for learning:

- Siraj Raval [<https://www.youtube.com/channel/UCWN3xxRkmTPmbKwht9FuE5A>]  
He has > 200k subscribers! In order to get that many subscribers you usually have to do a beauty channel.
- Jason Brownie [<https://machinelearningmastery.com/>]

47



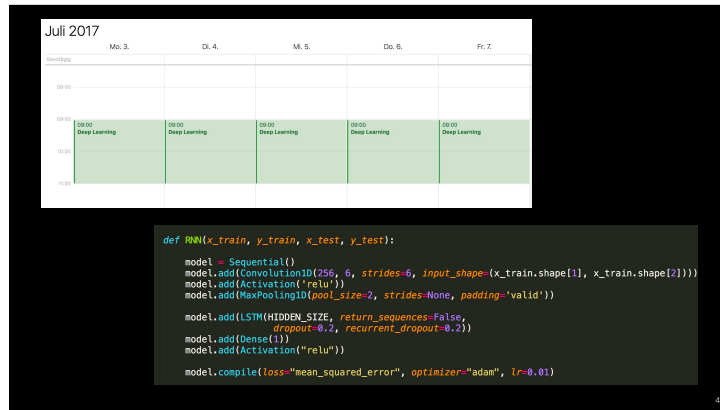
My story with Deep Learning started like three years ago when I got interested in this topic.

two years ago: gave first seminar with domain experts, yet that did not help that much.

I figured, that in order to really understand deep learning, I have to learn how to code this stuff.

48





So, I actually put it into my calendar: every morning from 9 to 11am: coding deep learning. That was a lot of fun!

I learned a lot by coding!

I started off with simple neural networks.

But eventually, it allowed me to completely automate my job:

49



I trained a network to give lectures and supervise my students.

[© iStock.com Kirillm]

[© iStock.com TadejZupancic]

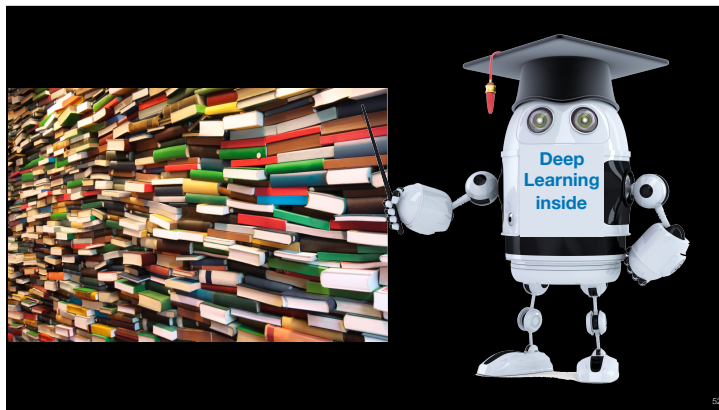
50



I trained a network to go to stupid Dean's meetings.

[© iStock.com Kirillm]  
[© iStock.com Rawpixel]

51



I trained a network to write tons of articles and grant proposals.

[© iStock.com Kirillm]  
[© iStock.com srebrina]

52



Of course I still receive the pay check myself!

[Andrew Magill CC BY 2.0]

[© Uwe Bellhäuser]

53

## Implications of Deep Learning for the DB community?

54

Now, as we have a vague idea about the tremendous progress in deep learning, what are the implications for the DB community?

54



Again: we are in an unfortunate situation: what should we do about this?

I see two general approaches to solve this:

1. revisiting database technology, and/or:
2. revisiting the „DBMS abstraction“.

[© iStock.com lexaarts]

[© iStock.com alphaspirt]

[© iStock.com pijama61]

55



56

## Deep Learning to improve DBMS performance

57

Obviously, we could use deep learning to revisit any kind of DBMS-performance “problem“:

- from 12-way to 14-way joins
- query optimisation: English to French? Here it is SQL to machine code!  
Correctness is an issue to look at, but I believe that it can be solved.
- result ranking (Google already does this)
- etc.

again: DBMS are already freakingly fast, are we working on a real problem?

57

## Deep Learning to replace humans in the DB-loop

58

Deep Learning to replace humans in the DB-loop

This sounds more interesting. This is not necessarily a performance problem in the sense of runtime, but more in the sense of “time to deploy a system“.

We should revisit any aspect of a DBMS where there are still humans in the loop, e.g.:

- automatic ETL
- automatic schemas
- automatic data integration
- automatic physical design:
  - + knob tuning
  - + index selection
  - + partitioning and replication

58



I don't think so: it is just another (second) lifebelt.

[© iStock.com Vaniatos]

59



This will neither save us from the deep learning Tsunami nor the Data Science Piranhas.

[© iStock.com lexaarts]  
[© iStock.com alphaspirt]  
[© iStock.com pijama61]  
[© iStock.com Vaniatos]

60

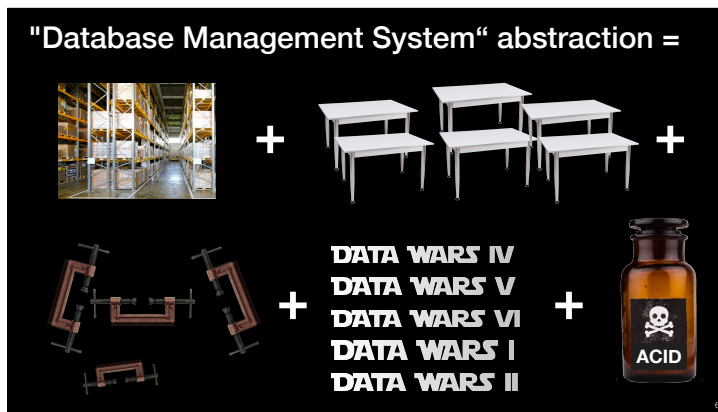
# Towards a DAI.ta Management System



61

The second approach is to revisit the “DBMS abstraction”.

61



What is the “DBMS” abstraction?

Well: you take a store, a bunch of tables, integrity constraints (including triggers), SQLs, and ACID

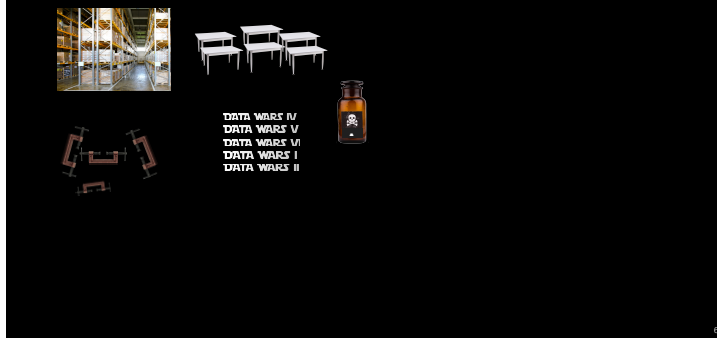
[copyright istockphoto.com s-cphoto]

[copyright istockphoto.com flyparade]

[copyright istockphoto.com designsstock]

62

## "Database Management System" abstraction =



You take all of this.

63

## "Database Management System" abstraction =

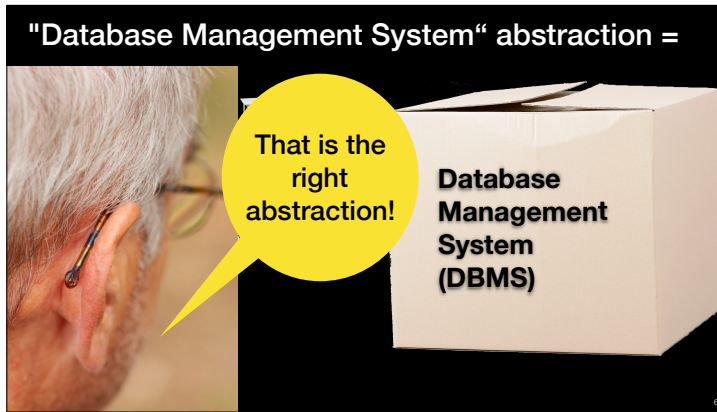


And put it into a box.

Then you put a label on that box: "Database Management System (DBMS)"

64

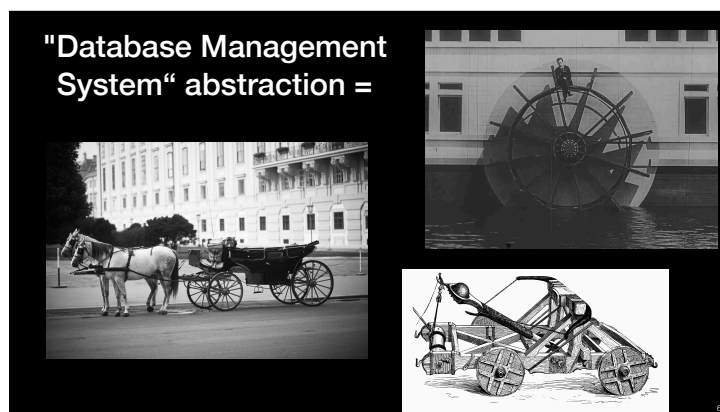




This design decision was made a long time ago, in the 70ies and 80ies, by the founders of our community.

[Copyright [istock.com](https://www.istock.com) jacekplacek1977]

65



However, when I look at this abstraction (I was born in 1972) it looks a bit weird to me.

It feels like a horse carriage, a still movie or a catapult.

All these abstractions were right at that time, but now they are somewhat dated. The same holds for the „DBMS abstraction“.

[Copyright [istockphoto.com](https://www.istockphoto.com) ooyoo]

[Copyright [istockphoto.com](https://www.istockphoto.com) WilshireImages]

[<https://www.youtube.com/watch?v=UWEjxkkB8Xs>]

66

## How do we enter the Data Science stack?

67

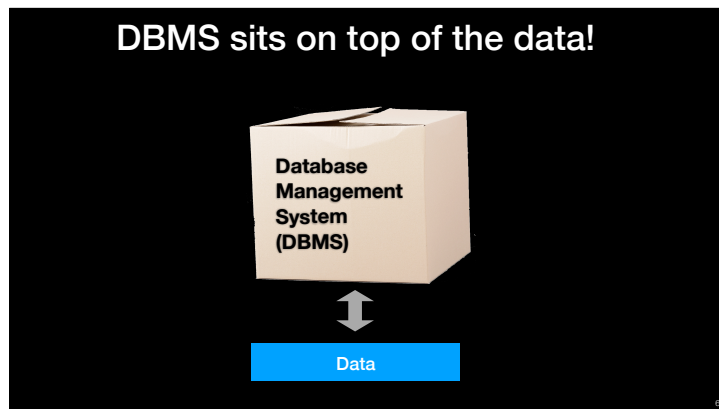
How do we enter the Data Science stack?

not in the sense that we force developers to use our stuff, i.e.  
“you must use our system, otherwise you are a bad developer“

no, it should rather be that these people approach us like:  
“oh your system is so cool, this even hotter than git and slack combined, where can I download it?“

In my view the solution to this problem has to do with the “DBMS“ abstraction:

67

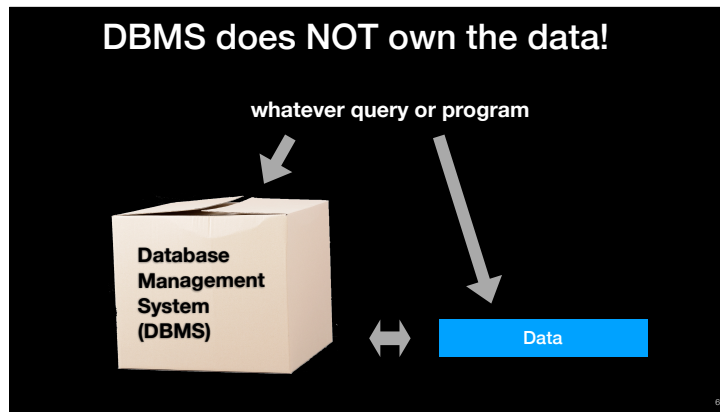


The DBMS sits on top of the data! It owns the data.

This is the root of all evil.

We need to give up that control! We need to give up this exclusiveness!

68

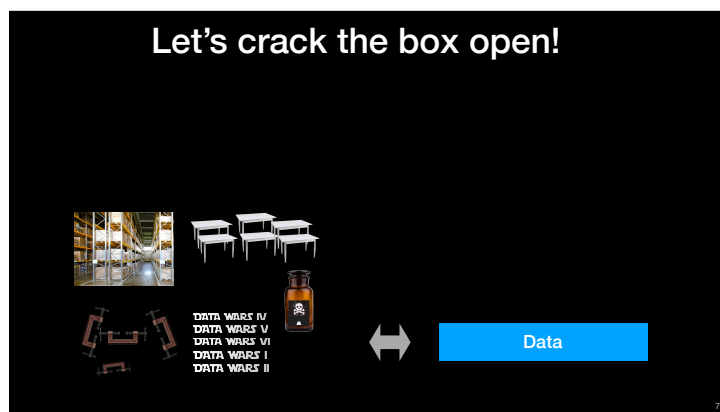


The DBMS should rather sit side by side the data.

If you execute whatever query or program it may go through the DBMS or bypass it.

The DBMS should be able to live with that! This has several implications.

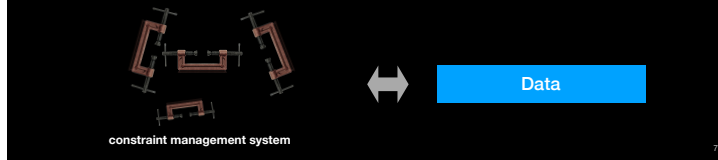
69



We should also crack the DBMS-box open. We rarely need all the bells and whistles.

70

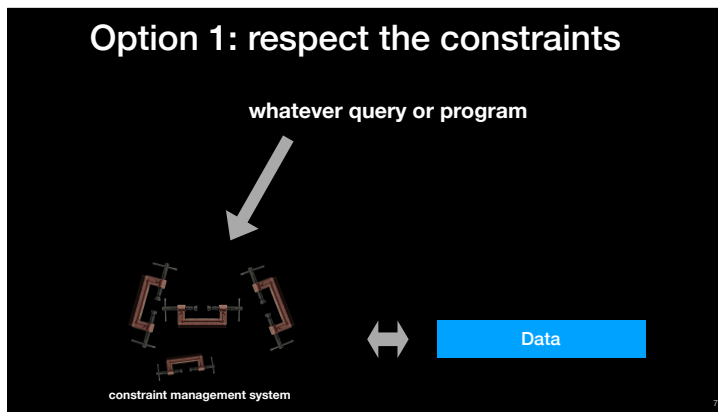
## Let's only consider constraints (for a moment)



For instance, let's assume for the moment, that we have a system that does exactly one thing: manage constraints including schemas, foreign key constraints, etc.

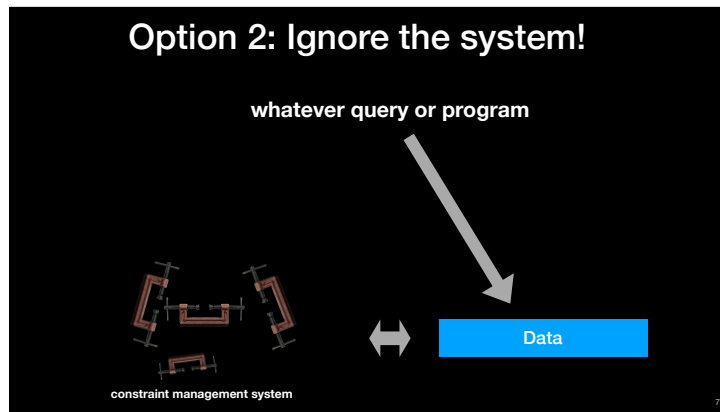
71

## Option 1: respect the constraints



Now, if we have a query, it can either go through that system (constraints may be checked)...

72



... or operate directly on the data.

In the latter case, we need to define what happens in case integrity constraints get violated.

The point is: we have two different views on the data:

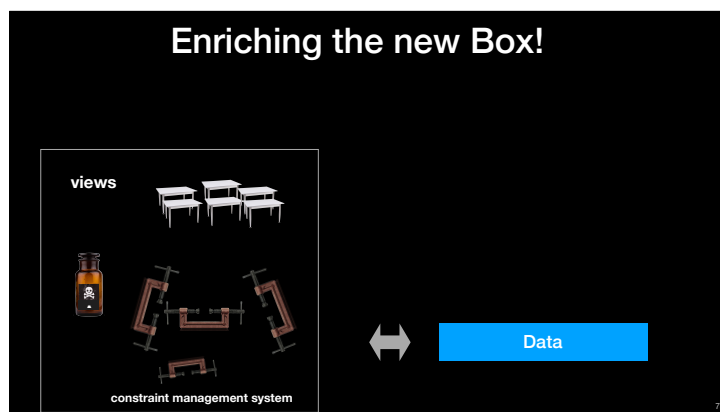
1. one raw view of the data
2. one with glasses on: we see the data through our integrity constraint glasses

A program can do whatever it wants with the data.

Only if we look at the data through the glasses we detect the violation of an integrity constraint.

However, in contrast to a DBMS, we will have to live with it (rather than rejecting the INSERT operation in the first place).

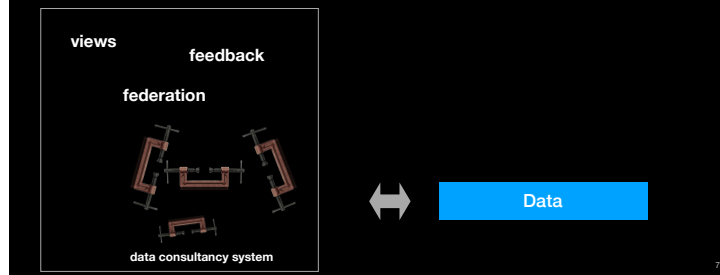
73



We could also add other features to this new box...

74

## Enriching the new Box!



...like views, (user-)feedback and (data-) federation mechanisms.

75

**constraints,  
views,  
federation,  
feedback**

76

76

constraints,  
views,  
federation,  
feedback

77

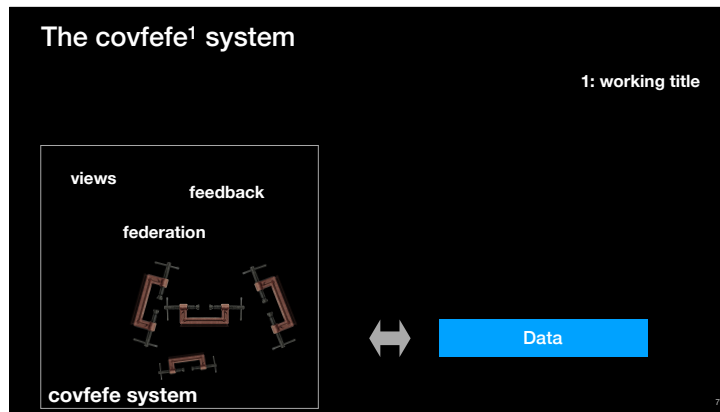
77



78

This was very visionary: in a single tweet: putting forward a visionary system idea!

78



Core idea:

provide DB technology as a toolset standing side by side the data, not on top (owning) it, but not as a library (as in Spark or Flink)

covfefe allows you to define how much power the covfefe system should exercise on your data, not only in terms of reading (search) but also writing and transforming data

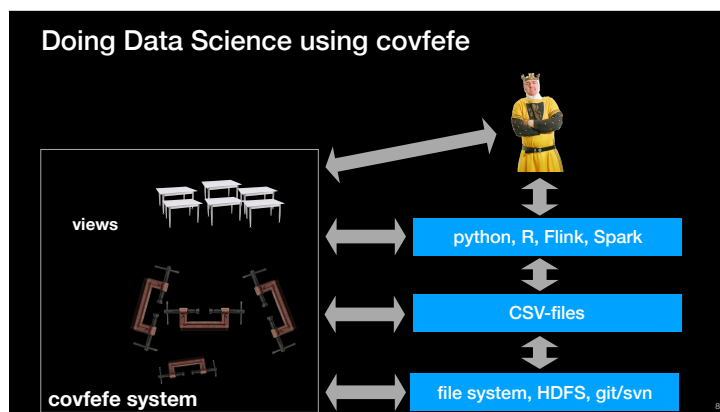
covfefe is more of a consultant, a data management advisory, indexing, and query processing service

think of it as a search engine PLUS data transforms and writes

how this works exactly, we will have to find out. and define it as a community.

But what it should allow you to do is at least the following:

79

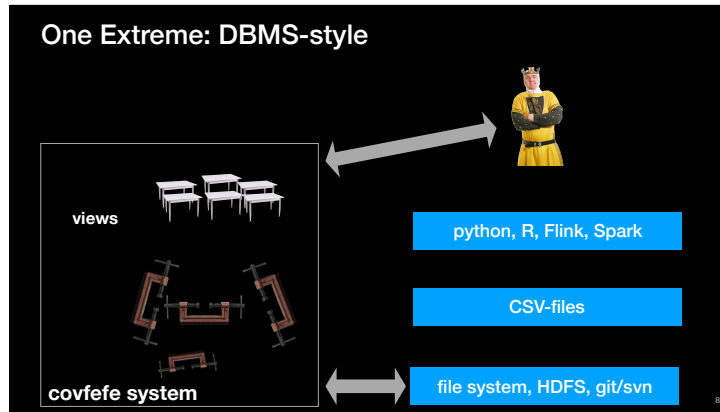


One feature of the system should be that it may help the user managing the data at any level.

[Copyright [istock.com](https://www.istock.com) hidesy]

80



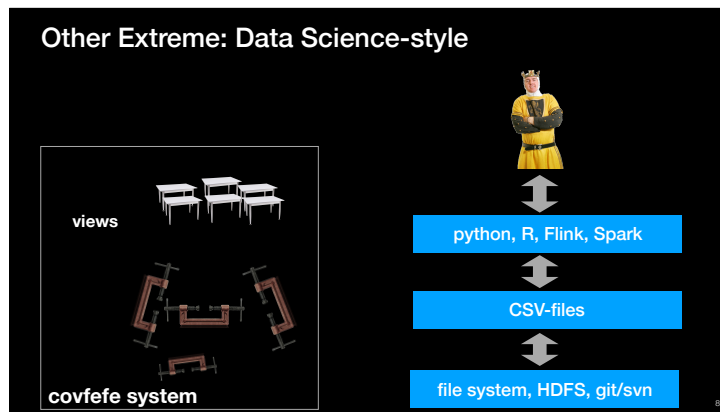


One extreme: DBMS-style

The user interacts with the system through SQL, all data processing happens inside coffee which in turn stores data on the file system.

Covfefe simulates a DBMS.

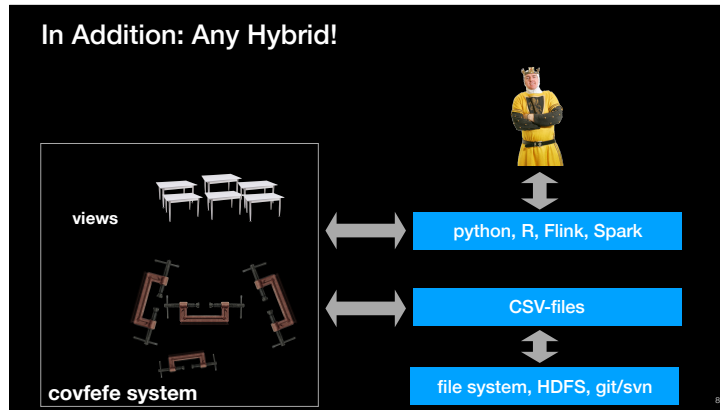
81



Other Extreme: Data Science-style

The user totally ignores covfefe. All data processing happens outside. Still covfefe may actively suggest things to the user on how to improve his/her data management.

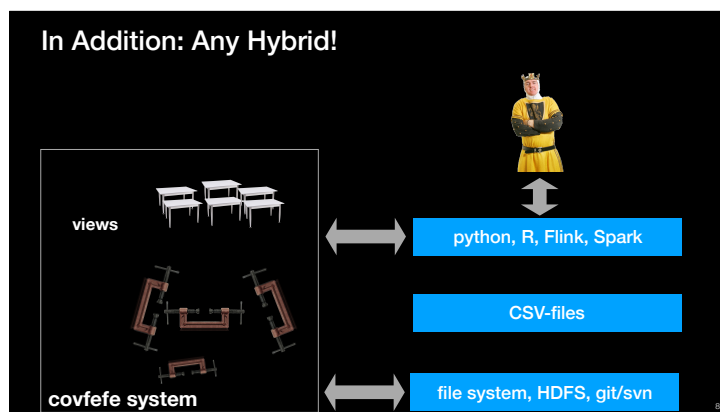
82



In Addition: Any Hybrid!

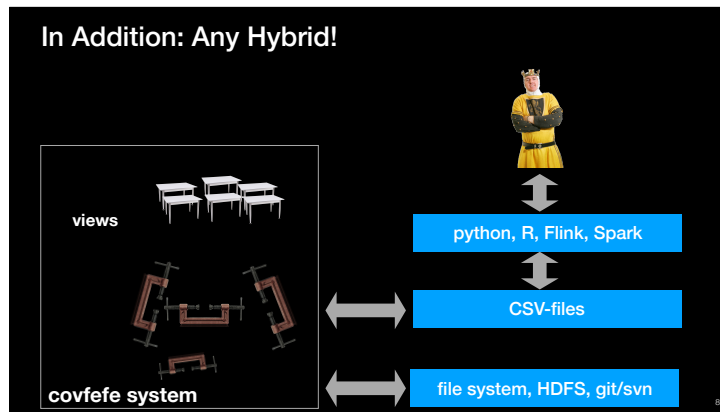
Here the user interacts with covfe through SQL which stores data as CSV-files which in turn are stored on the file system. Like that the system emulates NoDB [Ioannis Alagiannis, Renata Borovica-Gajic, Miguel Branco, Stratos Idreos, Anastasia Ailamaki. NoDB: Efficient Query Execution on Raw Data Files. Communications of the ACM, Vol. 58 No. 12, Pages 112-121]

83



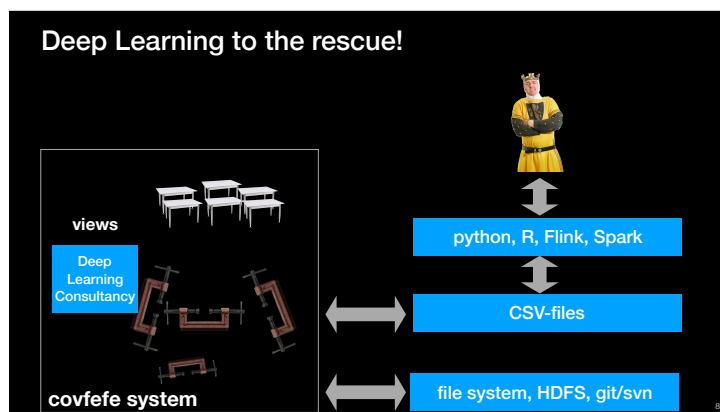
alternatively the user could write python scripts which are actually internally run on a database. Like that python is the frontend to the database. The user should not be aware that underneath is a database.

84



Or as another alternative the user could use python which operates on CSV-files which are actually just views provided by covfefe, rather than “real” CSV-files...

85



Anyway: to make this happen, there is a great opportunity here when it comes to the interaction of covfefe and the user.

Covfefe should consult the user, offer suitable views, queries, storage structures, etc.

This is an interesting research opportunity at the intersection of deep learning and databases!

86

# Towards a DAI.tabase Manifesto

1. all data must be kept in its original form
2. everything else are (super)views on that data, including „schemas“ & „tables“
3. superviews may or may not be materialised
4. superviews may or may not define integrity constraints both for reading and/or writing
5. the adherence to superviews' integrity constraints may be configured (this includes everything from data formats, domains, „keys“, „foreign key constraints“, up to „triggers“)
6. data may only be appended to, but never deleted, all data is versioned, versions are transactional
7. data may be processed using any language
8. ...

interested?

meet me after my talk  
or  
send me an email:

jens@daimond.ai

87

In any case, I believe the time is ripe to work on a new manifesto on how a future-proof data processing system should look like.

A system that serves traditional (semi-)structure requirements but also data science-workflows.

This system does not exist yet.

What kind of system should we build to support users in handling their data such that they are able to easily exploit our technology without constantly reinventing the wheel?

I would like to invite whoever is interested to work on such a Manifesto.

The goal should be a Sigmod Record/CACM or similar article.

I will coordinate that effort.

If you are interested, send me an email.

87



Going back to our metaphor of the database community drowning and being attacked.

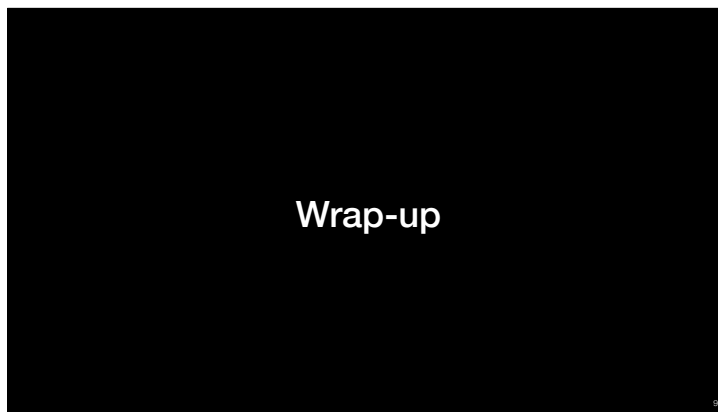
88



I believe that coming up by with such a manifesto as a coordinated effort, we could come up with a cool new type of data processing system that would really make a difference in the world of data science. A bit like the DB-communities mars-project.

[Copyright [istockphoto.com](https://www.istockphoto.com) Daniela Manguuca]

89

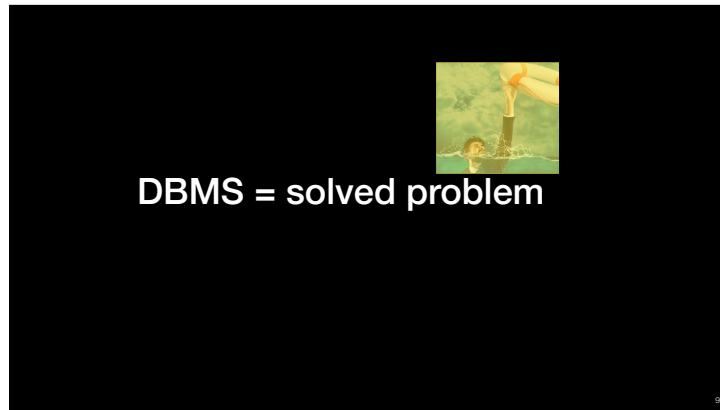


To wrap-up:

This is not yet another talk whining about a lost opportunity, we have had many of those talks in the past decade.

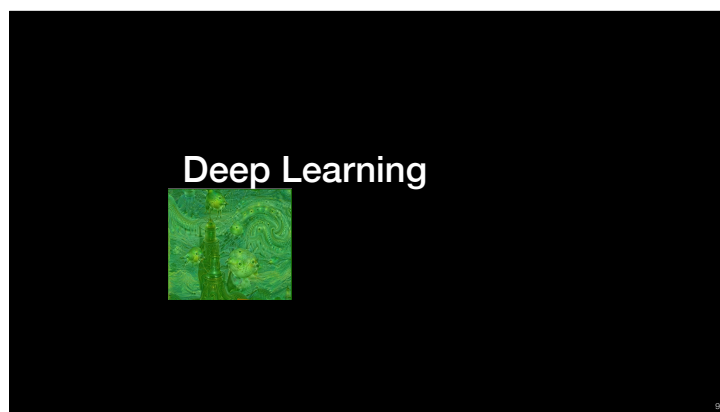
This is rather a talk showing how we can catch a spaceship taking us to new galaxies.

90



I explained that when it comes to performance, DBMS is a solved problem. If you configure your system well, almost everything can be executed in the blink of an eye. Our systems are already insanely great. We support workloads that simply do not exist.

91



I briefly sketched the foundations of deep learning and contrasted the achievements of deep learning with the achievements in the area of databases.

The achievements of deep learning are totally stunning.

92

## Revisiting Database Technology



93

I showed different opportunities for applying deep learning to databases.

There are basically two approaches:

(1a) is to revisit existing db technology and see whether deep learning can help. For instance we could get from 12- to 14-way joins...

(1b) is to reexamine everything where there are humans in the loop: automatic schema design, ETL, physical design advisory, ...

This is all cool, however, it won't help us to impress the data science folks.

We need a more radical approach:

93

## Towards a D.A.I.ta Management System



94

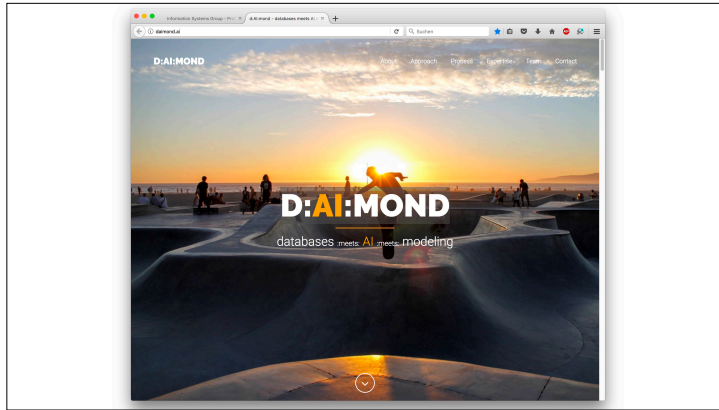
(2) We need to revisit the "DBMS abstraction". That abstraction was fine at the time. However, since then many things have changed, in particular: data is spread across a zillion devices in different formats and silos. And hardware has become extremely powerful. The good old „DBMS abstraction“ does not really work anymore.

We need to write a new manifesto.

If you are interested in thinking about this, let me know: [jens <dot> dittrich <at> infosys <dot> uni-saarland <dot> de](mailto:jens.dittrich@infosys.uni-saarland.de)

Thanks!

94



<http://daimond.ai>