Summary

Big Data Engineering
(formerly Informationssysteme)

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

July 20, 2023

Discussed Applications & Topics in the Lecture

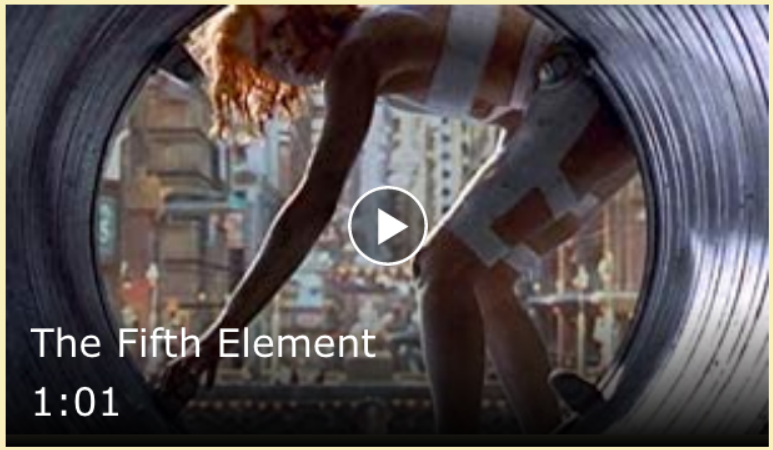
1. **IMDb**
2. NSA
3. Basics: Query Optimisation
4. Trading, Banking, Ticket Systems
5. From Databases to the Web
6. Data Journalism
7. Data in the Wild

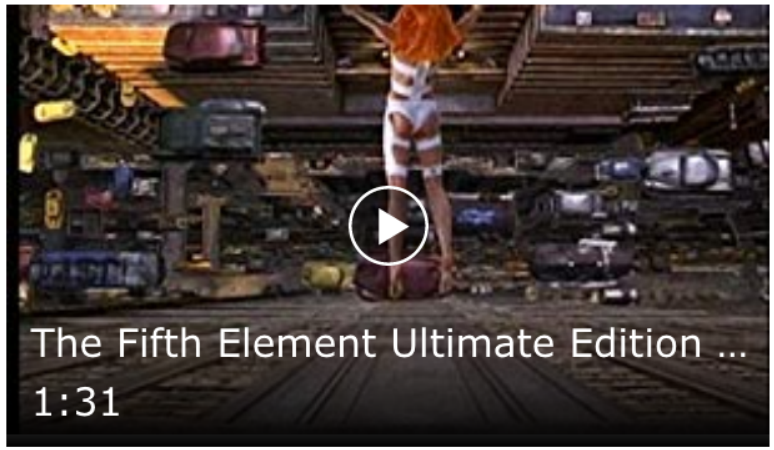
IMDb

the fith element

All

The Fifth Element (1997)
Bruce Willis, Milla Jovovich





The Fifth Element (1998)
Action, Adventure, Sci-Fi

Elementary: Season 5 - The Fifth Elementary (2017)
Nelsan Ellis, Jon Michael Hill

The Fifth Element (2002)

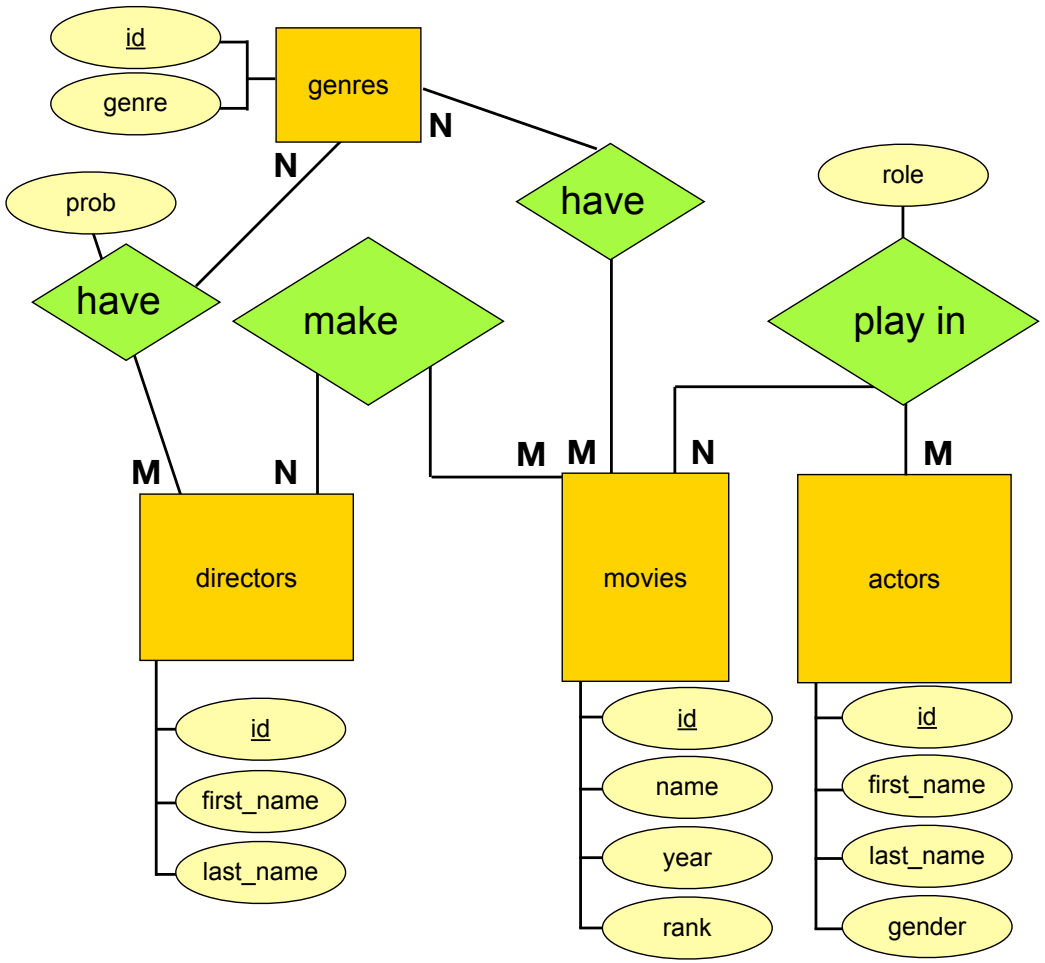
Beatboxing: The Fifth Element of Hip Hop (2011)

Shazam! Soars
Top
Weekend Box Office

Browse trailers »

Zachary Le

Functionalities (notation without arrows)

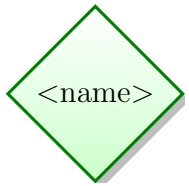
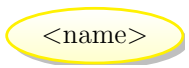
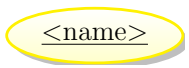
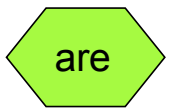


All good.

This **is** a correct ER diagram.

We do not use arrows in the following, as they could otherwise lead to confusion with other notation elements and one of the zillion variants of ER.

Overview of Entity Relationship Modelling Elements

Symbol	Meaning
	entity type
	relationship type
1, N or M	functionality (Chen notation)
	attribute
	key (-attribute)
	inheritance

The colours of the symbols do not matter.
The inheritance “are” is also sometimes called “is a” (which is inconsistent because of the plural of the entity types).

IMDb (Part 1)

4. Transfer of the basics to the concrete application

see various examples on IMDb above, summarised:

Question 1

How is the data on films, actors, directors etc. modelled and stored in IMDb?

Modelling: entity-relationship model, relational model
Data storage: This is a degree of freedom about which we do not have to make a decision at the moment: Any form of data storage that conforms to the relational model can be chosen!

This property is called **physical data independence**, i.e. as a user we don't need to know anything about how the data is physically stored and represented. In other words, we do not have to worry about how the relational model is implemented[2].

[2] For performance considerations, the physical implementation can then matter a lot after all. More on this later and in the core lecture Database Systems

IMDb

Question 2

How are links between these data modelled and stored in IMDb?

through foreign key relationships in the relational model

The Periodic Table of Relational Algebra Operators

Symbol	Name		Classification	
	German	English	Main operator class	unary/binary
σ	Selektion	selection	basic	unary
π	Projektion	projection		unary
$\cup$	Vereinigung	union		binary
$-$	Differenz	minus		binary
$\times$	kartesisches Produkt (oder Kreuzprodukt)	cross product		binary
ρ	Umbenennung	rename		unary
$\cap$	Schnitt	intersection	derived	binary
$\bowtie$	Verbund	join		
$\bowtie_\theta$	Theta-Verbund	theta join		
$\bowtie_{[L],[R]}$	Equi-Verbund	equi join		
$\ltimes$	Linker Pseudo-Verbund	left semi join		
$\rtimes$	Rechter Pseudo-Verbund	right semi join		
$\triangleright$	Linker Anti-Pseudo Verbund	left anti semi join		
$\triangleleft$	Rechter Anti-Pseudo Verbund	right anti semi join		
$⟕$	Linker äußerer Verbund	left outer join		
$⟖$	Rechter äußerer Verbund	right outer join		
$⟗$	Äußerer Verbund	full outer join		
γ	Gruppierung (mit Aggregation)	grouping (group by)	extensions	unary
Γ	Co-Gruppierung (**ohne** Aggregation)	co-grouping		binary
	...			

IMDb

Question 3

How do we query this data?

With relational algebra!

Attention

Relational algebra only abstractly describes **WHAT** has to be calculated, but not **HOW** this calculation is implemented algorithmically!

Unfortunately, expressions in relational algebra are sometimes a bit confusing. Therefore, another option is to hide the relational algebra under another language, i.e. design another query language and then translate it automatically into the relational algebra in each case.

Fundamental Theorem of Software Engineering (FTSE):

"We can solve any problem by introducing an extra level of indirection . . . except for the problem of too many levels of indirection."
[David Wheeler]

Main Learning Objectives

Entity Relationship Modelling

ER, Entity types, Relationship types (including n-ary), Domain, Attribute, Key, Functionalities (Chen notation), Functional Determined, N:M, 1:N, 1:1, N:M:P, Generalisation

Relational model

Relation, Tuple, Relational Schema, translating ER to RM, Key, Foreign Key

Main Learning Objectives

Relational Algebra

Operators (unary and binary), Selection (not to be confused with SELECT in SQL), Projection, Union, Difference, Cross product, Intersection, Theta Join, Group by vs Aggregation, Co-Grouping

Discussed Applications & Topics in the Lecture

1. IMDb
2. **NSA**
3. Basics: Query Optimisation
4. Trading, Banking, Ticket Systems
5. From Databases to the Web
6. Data Journalism
7. Data in the Wild

Whistleblowers related to Mass Surveillance

- since the existence of secret services, there have always been whistleblowers.
- the best known is Edward Snowden, who worked as a sysadmin at the NSA until May 2013 (Film adaptation: Snowden, documentary: Citizenfour)
- from June 2013, he began to gradually publish secret NSA documents documenting mass surveillance by the intelligence services
- other important whistleblowers were Martin and Mitchell, William Binney, Russ Tice, Thomas Tamm,Thomas Drake, Katharine Gun (movie: Official Secrets), Julian Assange, Chelsea Manning, Reality Leigh Winner
- whistleblower on Wikipedia

Laura Poitras / Praxis Films

NSA: Nationales Sicherheitsamt, Andreas Eschbach

Book idea

Imagine if computer technology had developed 70 years earlier. In the Weimar Republic there were already computers, the 'world network' and later on mobile 'people's telephones'. And extensive data collections. This treasure trove of data fell into the hands of the Nazis when they seized power. What effects would this have had?

- the book idea is brilliant
- data analysis is described technically correct (except for minor technical things) and in detail up to examples in "Structured query language"
- link to the book (in German)

Conceptual Execution Order for Grouping

```
SELECT   [A], [Aggregate functions F]    5. Aggregation and projection
FROM     <Input tables>                  1. Cross product over all input tables
WHERE    <Condition P1>                  2. Selection of tuples with condition P1
GROUP BY [B]                             3. Grouping
HAVING   <Condition P2 on agg-fct. H>    4. Selection of groups with condition P2
```

SELECT $A_1, \ldots, A_n, \underbrace{f_1([G_1]), \ldots, f_{k_F}([G_{k_F}])}_{=:F}$

FROM $T_1, \ldots, T_m$

WHERE P_1

GROUP BY $B_1, \ldots, B_l$

HAVING P_2 with conditions on $\underbrace{h_1([G1]), \ldots, h_{k_H}([G_{k_H}])}_{=:H}$

Rules

1. $[Q] = [T_1] \circ \ldots \circ [T_m]$ (common scheme of all occurring attributes of the request)
2. $[B] \subseteq [Q]$. ($[B]$ is a possibly empty subset of $[Q]$)
3. $[A] \subseteq [B]$. ($[A]$ is a possibly empty subset of $[B]$)
4. $[G_1], \ldots, [G_{k_G}], [H_1], \ldots, [H_{k_H}] \subseteq [Q]$. If $[G_i]$ or $[H_i]$ are empty, this is signalled by '*'..
5. P_2 may formulate conditions by means of aggregate functions H: even those that are not in the SELECT!

It must hold: $\{A1, \ldots, An\} \subseteq \{B1, \ldots, Bl\}$.

Most Common Mistakes when Dealing with SQL 1/3

"SQL is a language for writing and reading individual tuples."

⇒ "I use SQL mainly for reading and writing individual tuples: CRUD (Create, Read, Update, Delete), i.e. some sort of a tuple-like file system".

That's like using an entire factory production line just as a bottle opener.

- the true strengths of SQL thus remain unused.
- functionality that is actually available in SQL is re-implemented, with all the (hidden) costs: quality assurance, testing, bug fixes, ...
- a clear violation of the Laziness Principles (slide set 00)

Most Common Mistakes when Dealing with SQL 2/3

"SQL and especially joins are slow."

⇒ "I prefer to use NoSQL, Hadoop or implement it myself".

- SQL and the performance of a program generated from SQL are **two different dimensions**
- eventually, SQL is used to translate it into an executable program
- the performance of that program depends on many factors, but has **nothing** to do with claimed limitations of SQL!
- the most important influencing factors: Indexes, (query) optimisation algorithms, Physical design.
- more on this later in this and in the core lecture

Most Common Mistakes when Dealing with SQL 3/3

"SQL cannot handle more structured data such as JSON, objects, graphs"

⇒ "I prefer to use a key/value store".

- the relational model was already extended for SQL 1999
- basic idea: Domains can be of any type (especially structured!) and not just "atomic types"
- rich datatypes: arrays, nested tables, composite types, ..
- SQL 2016: JSON
- good overview video on this: Markus Winand, The Mother of all Query Languages: SQL in Modern Times

A lot has happened since SQL-92...

But in many projects only SQL-92 or little more is used. That means a lot of potential and money is often wasted.

NSA

Question 1

How do we phrase more complex queries?

SQL!

NSA

Question 2

... and what ethical problems arise from these kind of data collections? How do we deal with them?

- civil disobedience
- technical approaches
- social approaches
- legal approaches

Main Learning Objectives

SQL

core idea, most common errors, basic structure of an SQL-92 query, conceptual execution order, joins, group by, outer Joins, HAVING, sub-queries, (dynamic) views

Big Data arithmetic

apples + pears = whatever; in other words, the information gain that can result from joining two data sets is usually unpredictable

Big Data: countermeasures

data economy, anonymisation, encryption, noisy data (differential privacy), anti-hardware administration, anti-software engineering, social approaches, legal approaches, corona warning app principles

NSA

Question 3

... and what if the data gets bigger? How do we actually get from SQL to an efficient program?

automatic query optimisation

Discussed Applications & Topics in the Lecture

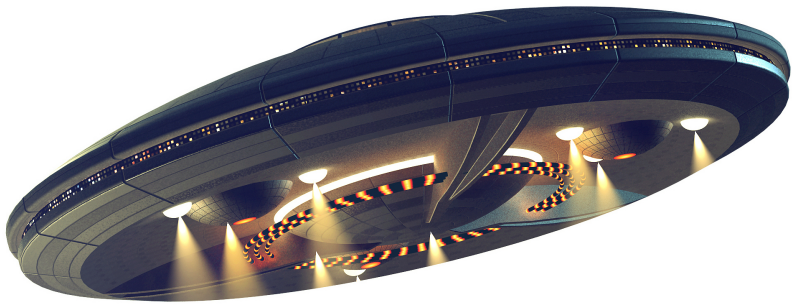
1. IMDb
2. NSA
3. **Basics: Query Optimisation**
4. Trading, Banking, Ticket Systems
5. From Databases to the Web
6. Data Journalism
7. Data in the Wild

Overview of SQL Processing Systems (SQL Engines)

Sqlite

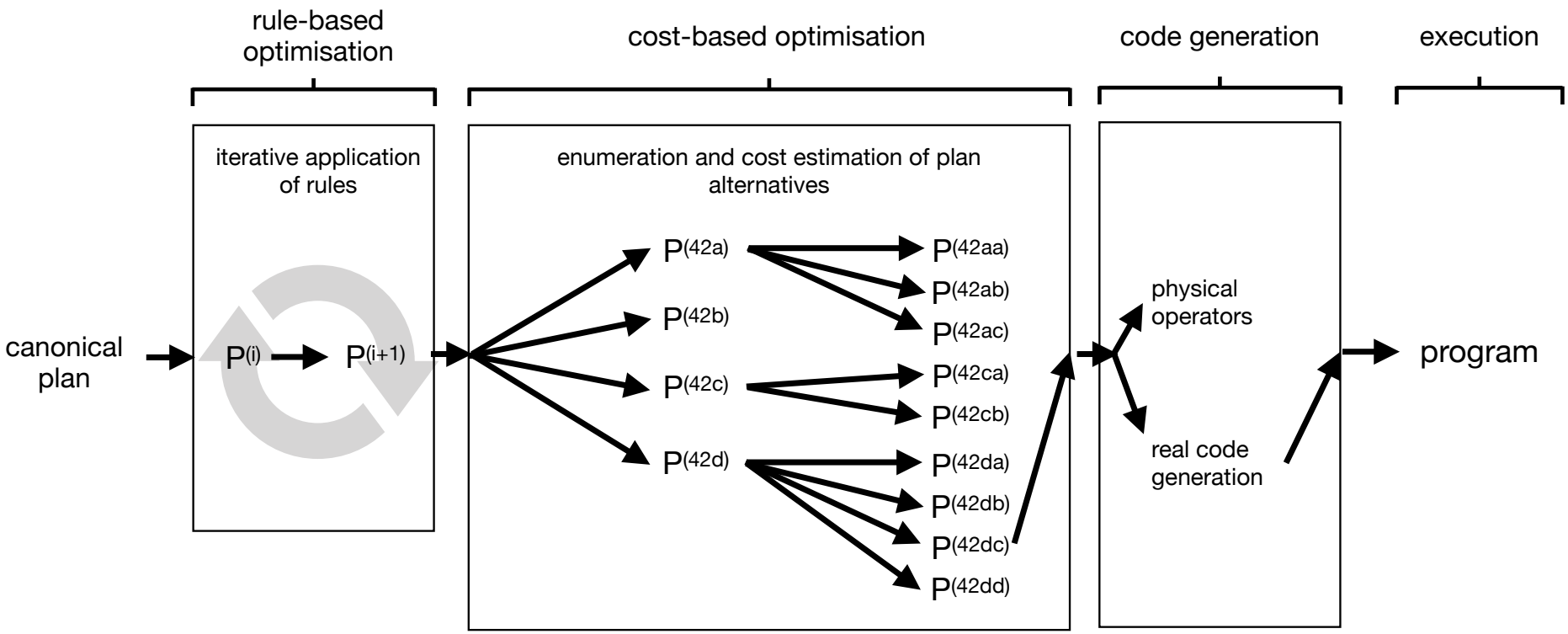
MySQL

PostgreSQL



Modern DBMS

Optimisation, Code Generation and Execution: Overview



Main Learning Objectives

Automatic query optimisation

basic functionality, canonical translation, heuristic optimisation, cost-based optimisation, code generation

Physical operators

logical vs physical operators, scan-based, nested-loops, hash-based

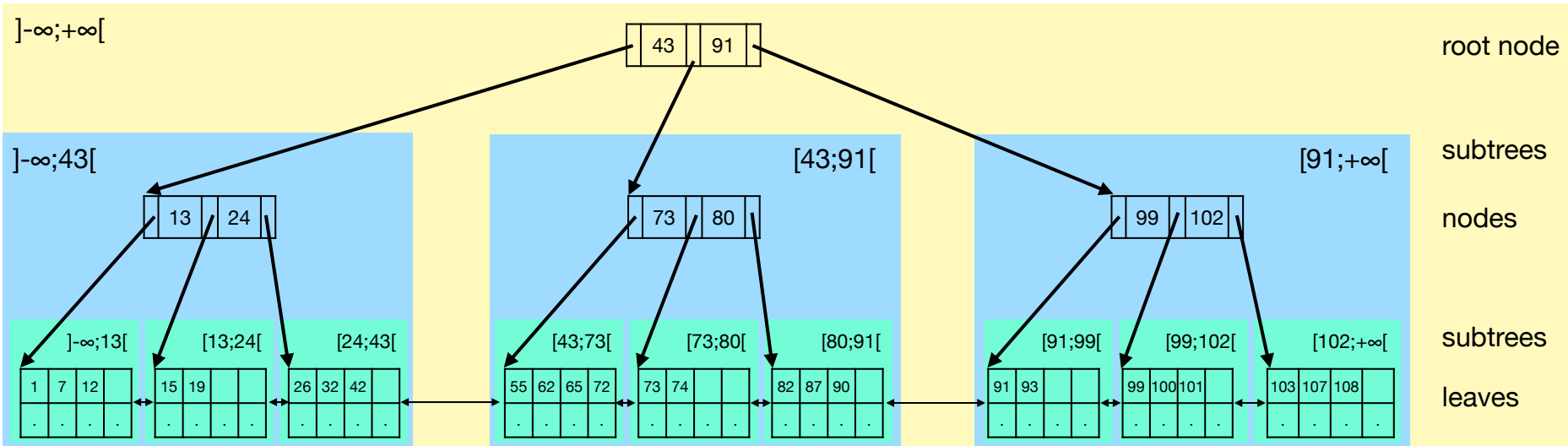
Heuristic optimisation

Rules, basic algorithm, iterative application of rules, predicate pushdown, join predicates and cross products to joins

Indexing

B-Trees

One Root Node and its Subtrees of Subtrees



Here, each of those subtrees is implemented by a single root leaf.

Main Learning Objectives

Cost-based optimisation

selectivity, cost models (intermediate results, I/O costs, total runtime, energy consumption), model vs reality, plan diagrams (Picasso diagrams)

Different Dimensions of Cost-Based Optimisation

Decisions on a **logical** level:

i. Which join order to take?
Examples: Which join order has less runtime: $R \bowtie (S \bowtie T)$ or $(R \bowtie S) \bowtie T$ or ...?

Decisions on a **physical** level:

ii. Which physical operator to use?
Examples: Hash-based join, sort-based join or XY join?
(cf. logical vs physical operator discussion)

iii. Use index or not? If yes, which index to use?
Examples: Scan relation or use binary search? Which type of index to use? Hash-based, tree-based or ... ?
(cf. Picasso-Notebook)

iv. Which resources to use for which sub-plan?
Examples: How many threads to use where? Which part of the plan gets how much computing time/main memory?

Main Learning Objectives

Join ordering

tree structure of the plan, left-deep, right-deep, bushy, order of input relations

Plan variants

join selectivity, join graph, commutativity of plans

Pipelining

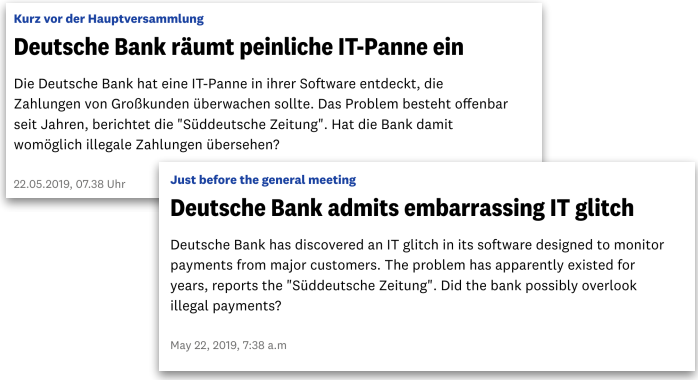
HashJoin algorithm, materialisation of intermediate results, pipeline breakers, plan sections

Physical optimisations

plan interpretation vs plan compilation, horizontal partitioning and indexes, multi-threading, example in Spark

Discussed Applications & Topics in the Lecture

1. IMDb
2. NSA
3. Basics: Query Optimisation
4. **Trading, Banking, Ticket Systems**
5. From Databases to the Web
6. Data Journalism
7. Data in the Wild



Kurz vor der Hauptversammlung

Deutsche Bank räumt peinliche IT-Panne ein

Die Deutsche Bank hat eine IT-Panne in ihrer Software entdeckt, die Zahlungen von Großkunden überwachen sollte. Das Problem besteht offenbar seit Jahren, berichtet die "Süddeutsche Zeitung". Hat die Bank damit womöglich illegale Zahlungen übersehen?

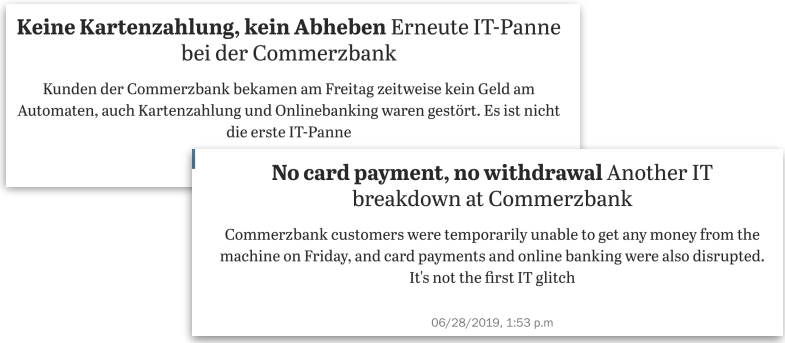
22.05.2019, 07.38 Uhr

Just before the general meeting

Deutsche Bank admits embarrassing IT glitch

Deutsche Bank has discovered an IT glitch in its software designed to monitor payments from major customers. The problem has apparently existed for years, reports the "Süddeutsche Zeitung". Did the bank possibly overlook illegal payments?

May 22, 2019, 7:38 a.m



Keine Kartenzahlung, kein Abheben Erneute IT-Panne bei der Commerzbank

Kunden der Commerzbank bekamen am Freitag zeitweise kein Geld am Automaten, auch Kartenzahlung und Onlinebanking waren gestört. Es ist nicht die erste IT-Panne

No card payment, no withdrawal Another IT breakdown at Commerzbank

Commerzbank customers were temporarily unable to get any money from the machine on Friday, and card payments and online banking were also disrupted. It's not the first IT glitch

06/28/2019, 1:53 p.m

START INS JAHR

IT-Probleme bei Bank Austria führten zu verspäteten Buchungen

Ein Programmfehler von Rechnern hatte verspätete Buchungen, auch von Pensionen und Gehältern, zur Folge. Informiert hat die Bank nicht

Renate Graber
8. Jänner 2019, 17:37

START OF THE YEAR

IT problems at Bank Austria led to late bookings

A bug in computers resulted in late postings, including pensions and salaries. The bank did not inform

Renate Graber
January 8, 2019, 5:37 p.m

Datenverlust

Gravierende IT-Panne bei der UBS – bis zu 1500 Kunden betroffen

Mi 05.12.2018 - 10:01 Uhr
von beat schmid, ch media

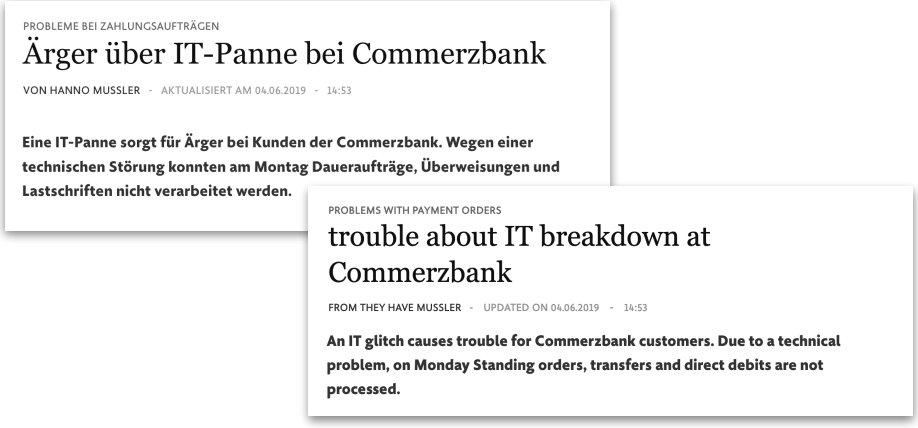
Ein IT-Fehler hat bei der UBS zu einer Datenpanne geführt. Dokumente von bis zu 1500 Kunden könnten verloren gegangen sein.

data loss

Serious IT breakdown at UBS - up to 1500 customers affected

We 05.12.2018 - 10:01 Clock
from beat schmid, ch media

An IT error has led to a data breach at UBS. Documents from up to 1500 customers could have been lost.



PROBLEME BEI ZAHLUNGSAUFTRÄGEN

Ärger über IT-Panne bei Commerzbank

VON HANNO MUSSLER - AKTUALISIERT AM 04.06.2019 - 14:53

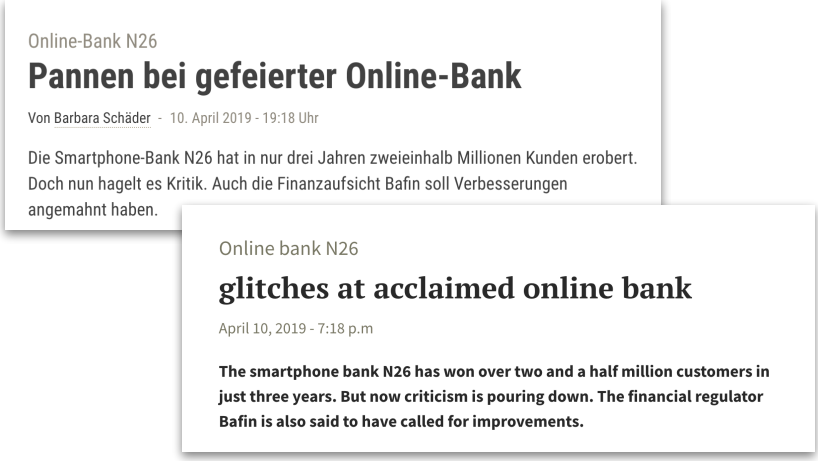
Eine IT-Panne sorgt für Ärger bei Kunden der Commerzbank. Wegen einer technischen Störung konnten am Montag Daueraufträge, Überweisungen und Lastschriften nicht verarbeitet werden.

PROBLEMS WITH PAYMENT ORDERS

trouble about IT breakdown at Commerzbank

FROM THEY HAVE MUSSLER - UPDATED ON 04.06.2019 - 14:53

An IT glitch causes trouble for Commerzbank customers. Due to a technical problem, on Monday Standing orders, transfers and direct debits are not processed.



Online-Bank N26

Pannen bei gefeierter Online-Bank

Von Barbara Schäder - 10. April 2019 - 19:18 Uhr

Die Smartphone-Bank N26 hat in nur drei Jahren zweieinhalb Millionen Kunden erobert. Doch nun hagelt es Kritik. Auch die Finanzaufsicht Bafin soll Verbesserungen angemahnt haben.

Online bank N26

glitches at acclaimed online bank

April 10, 2019 - 7:18 p.m

The smartphone bank N26 has won over two and a half million customers in just three years. But now criticism is pouring down. The financial regulator Bafin is also said to have called for improvements.

Main Learning Objectives

Database management systems (DBMS)

What distinguishes a DBMS?

Database Management Systems (DBMS)

Relational database systems (RDBMS, usually just DBMS) have been developed since the 1970s. Modern DBMS typically have the following features:

1. extended relational model: JSON, arrays, text, spatial data, etc.
2. very extensive SQL dialect (depending on the system)
3. **automatic** query optimiser (rule- and cost-based)
4. very high performance (mostly, depending on the system)
5. massive support for physical design (index structures, partitioning, caching, materialisation)
6. support for "modern" hardware: DRAM, PRAM, FPGAs, GPUs, ...

Database Management Systems (DBMS)

And as far as avoiding and dealing with error situations is concerned, in particular:

7. extensive support for transactions and ACID
8. concurrency control (**automatic** concurrency control) **today!**
9. **automatic** crash recovery, replication, backup
10. **automatic** consistency control (unfortunately often not used)
11. dynamic views, access rights (logical data independence)

And that is what I promised with the teaser slide when announcing this lecture:

Summary: Analogy ESP (Electronic Stability Program)

before 1997

controls that prevent the problem

+ incredibly talented driver

after 1997

+ automated braking of individual wheels

(ESP)

mandatory since 2014

Trading, Banking, Ticket Systems

Question 1

How do we allow concurrent modification of data without creating erroneous data?

Concurrency Control

Question 2

How do we design this so that the process and the resulting overall system are efficient?

For example, through S2PL with predicate locks or weakened guarantees by means of isolation levels.

Modern DBMSs use multi-version concurrency control (MVCC) or a combination with S2PL (see core lecture).

Main Learning Objectives

Transactions

read vs write operations, ACID: atomicity, consistency and consistency conditions, isolation, durability, concurrency control automation.

Serialisability theory

schedule, conflict operations, schedule with total vs partial order, serial schedule, conflict equivalent, conflict serialisable, swapping non-conflict operations, conflict graph, cycle, conflict serialisability in conflict graph, creating a cycle-free conflict graph, batch vs operation-at-a-time.

Main Learning Objectives

Two-Phase Locking (2PL)

locks, read lock (shared lock, R or S), write lock (exclusive lock, X), read/write lock compatibility, 2PL vs conflict serialisability, cascading rollback, strict 2PL (S2PL), phantom problem, predicate locks, deadlocks, conservative 2PL (C2PL), dirty reads, wait-for-graph

Isolation Levels

weakening of isolation guarantees, `READ UNCOMMITTED`, `READ COMMITTED`, `REPEATABLE READ`, `SERIALIZABLE`, the default is usually **not** `SERIALIZABLE`!, impact on applications

Concurrency Control Summary

Isolation

Most database systems guarantee atomicity and isolation for transactions. Concurrent execution of transactions increases the performance of these systems enormously without creating problems.

Caution with Isolation Levels

Isolation levels are sometimes difficult to interpret. If in doubt, always use the stronger isolation level! Always check which isolation level is set by default and what this actually means in the used database system!

Conflict Serialisable $\neq$ `Serializable`

In the database world, there are unfortunately a confusing number of similar-sounding but different concepts.

Conflict Serialisable $\not\Rightarrow$ `Serializable`

The schedule is conflict equivalent to a serial schedule.

The problems with that:

1. What if not all transactions in the schedule commit? This potentially violates isolation.
2. What if a single operation reads or writes more than one tuple?

That's why we need to do more:

`Serializable` vs Serialisable

`Serializable`, perfect Isolation

The isolation level `Serializable` is much stronger than the theoretical concept "conflict serialisable" and also prevents problems that can occur due to aborting transactions. This applies both to individual tuples and to ranges/sets of tuples that can lead to phantom problems.

Serialisable...

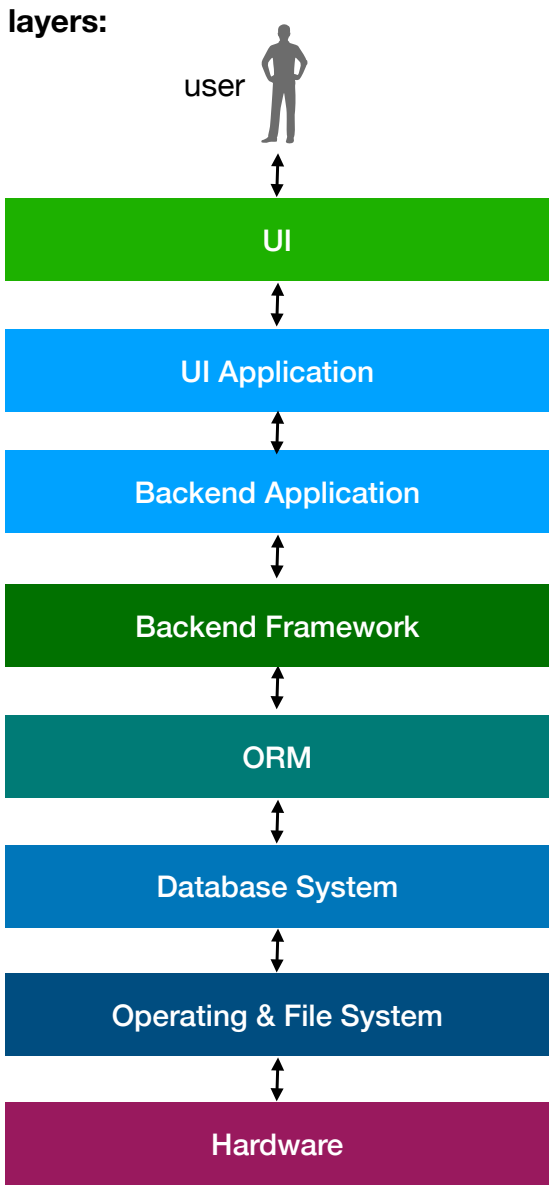
...can mean:

1. Synonym for 'conflict serialisable',
2. Synonym for `Serializable`, i.e. perfect isolation in the sense of the I in ACID

Discussed Applications & Topics in the Lecture

1. IMDb
2. NSA
3. Basics: Query Optimisation
4. Trading, Banking, Ticket Systems
5. **From Databases to the Web**
6. Data Journalism
7. Data in the Wild

Separate Backend- and UI-Application-Layers



example technologies:

application-code written in Javascript, Vue

UI-code written in javascript, PHP, React

application-code written in Python, Rust

Django

object-relational mapper, Django ORM

PostgreSQL, MySQL, Oracle, SQLite, DuckDB

Linux, Windows, OS X, Android, iOS

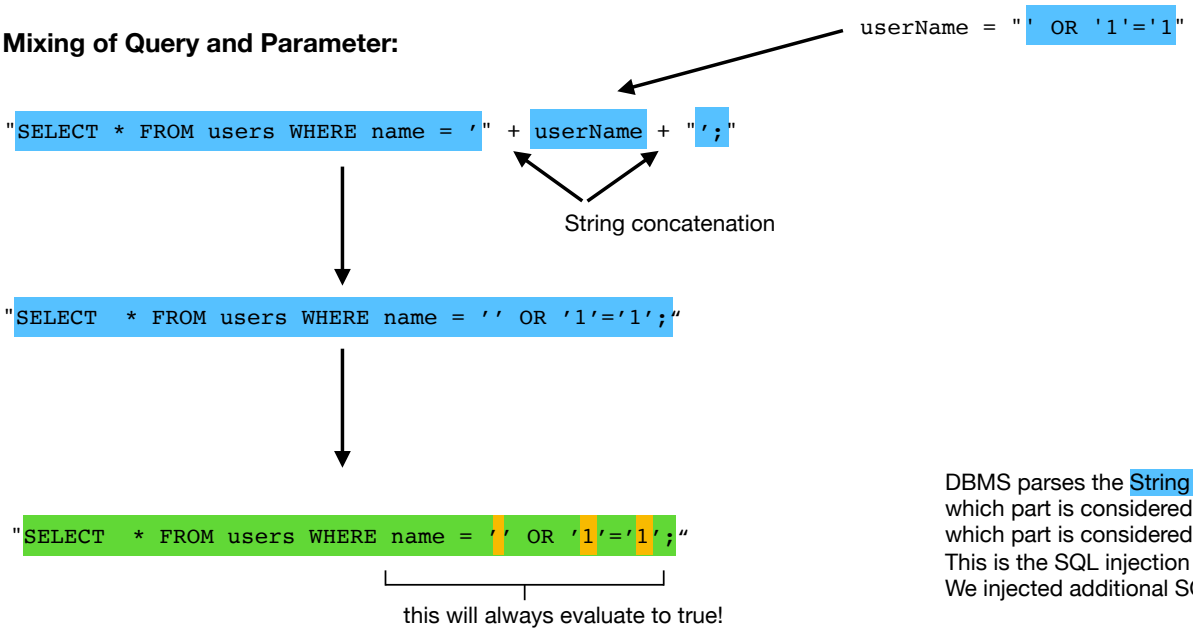
CPU, DRAM, SSD, hard disk

Decouple UI- and Backend Application

Allows for different frameworks for UI- and backend

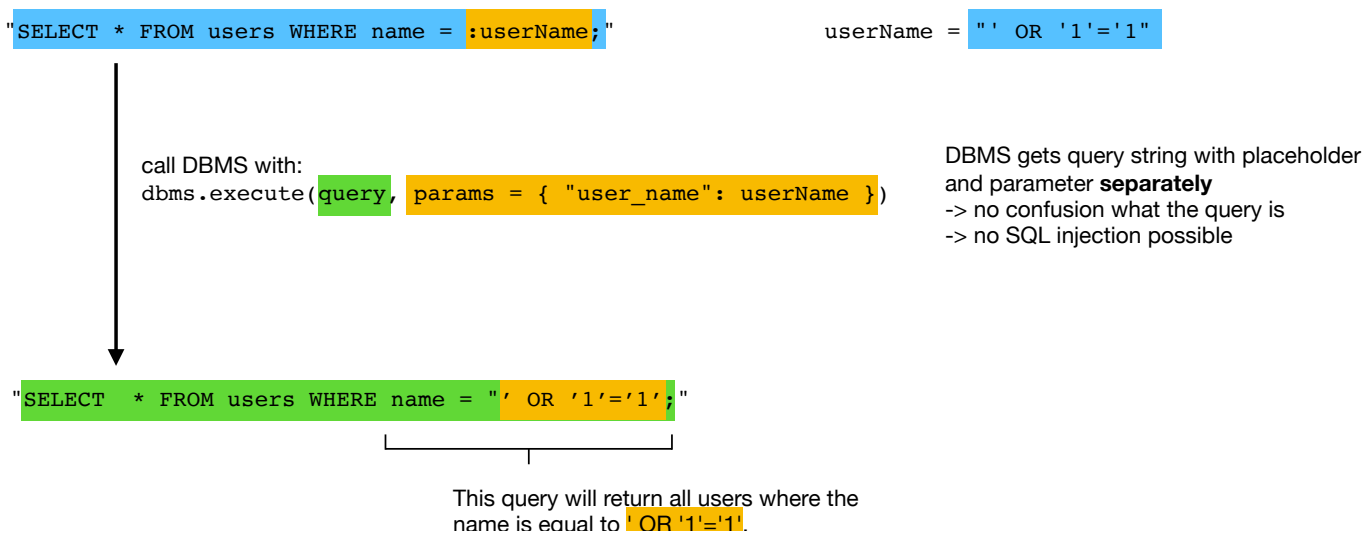
SQL Injection vs Separated Query and Parameters

Mixing of Query and Parameter:



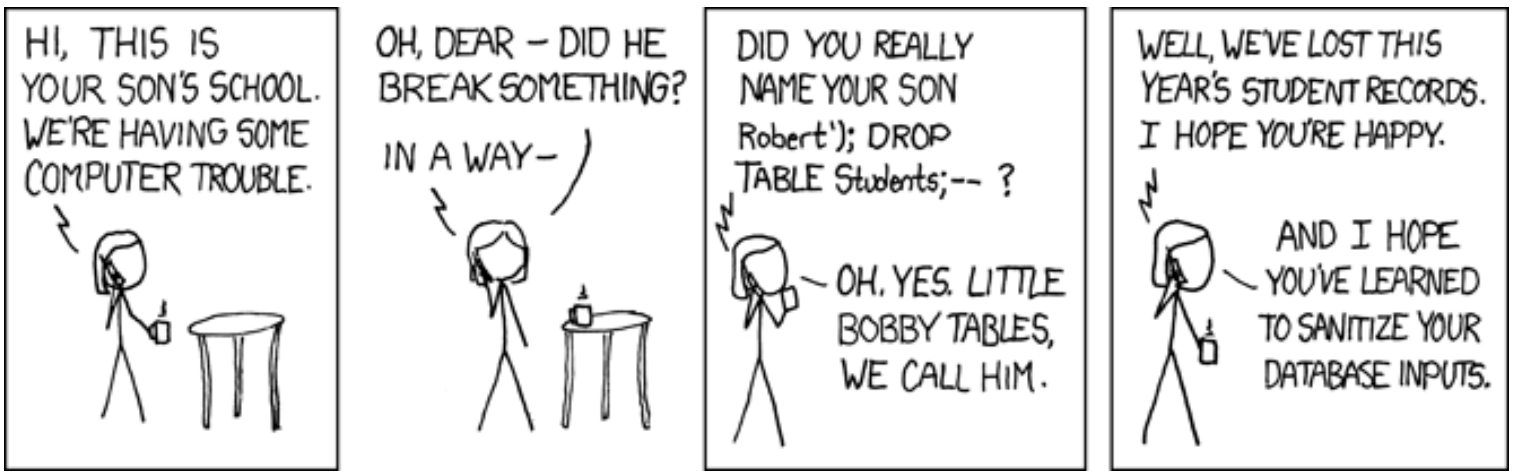
DBMS parses the String and determines itself
which part is considered the query and
which part is considered the parameter
This is the SQL injection **problem**:
We injected additional SQL into the original query!

Clear separation of Query und Parameter:



DBMS gets query string with placeholder
and parameter **separately**
-> no confusion what the query is
-> no SQL injection possible

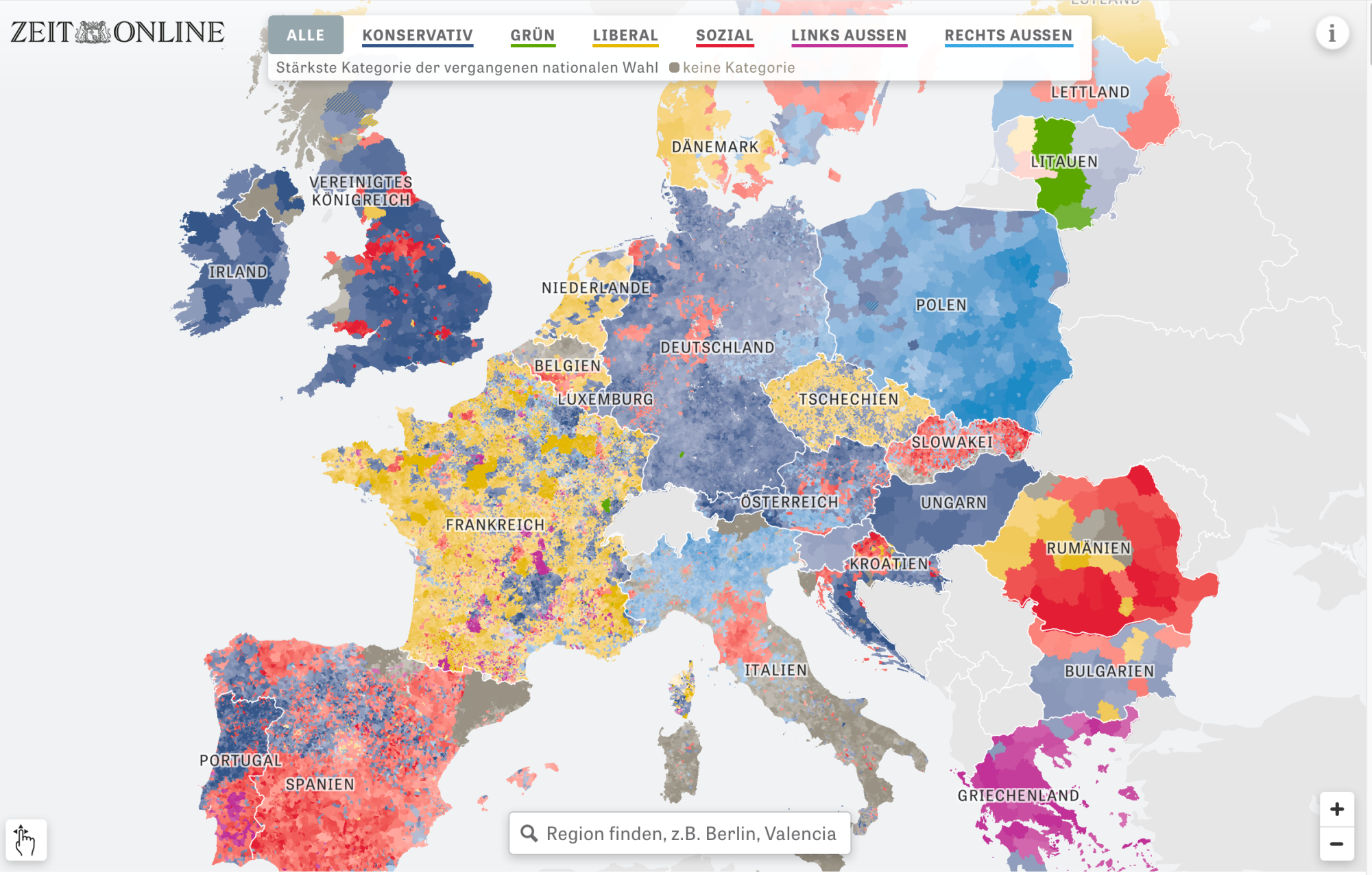
Little Bobby Tables



[Source: https://xkcd.com/327/]

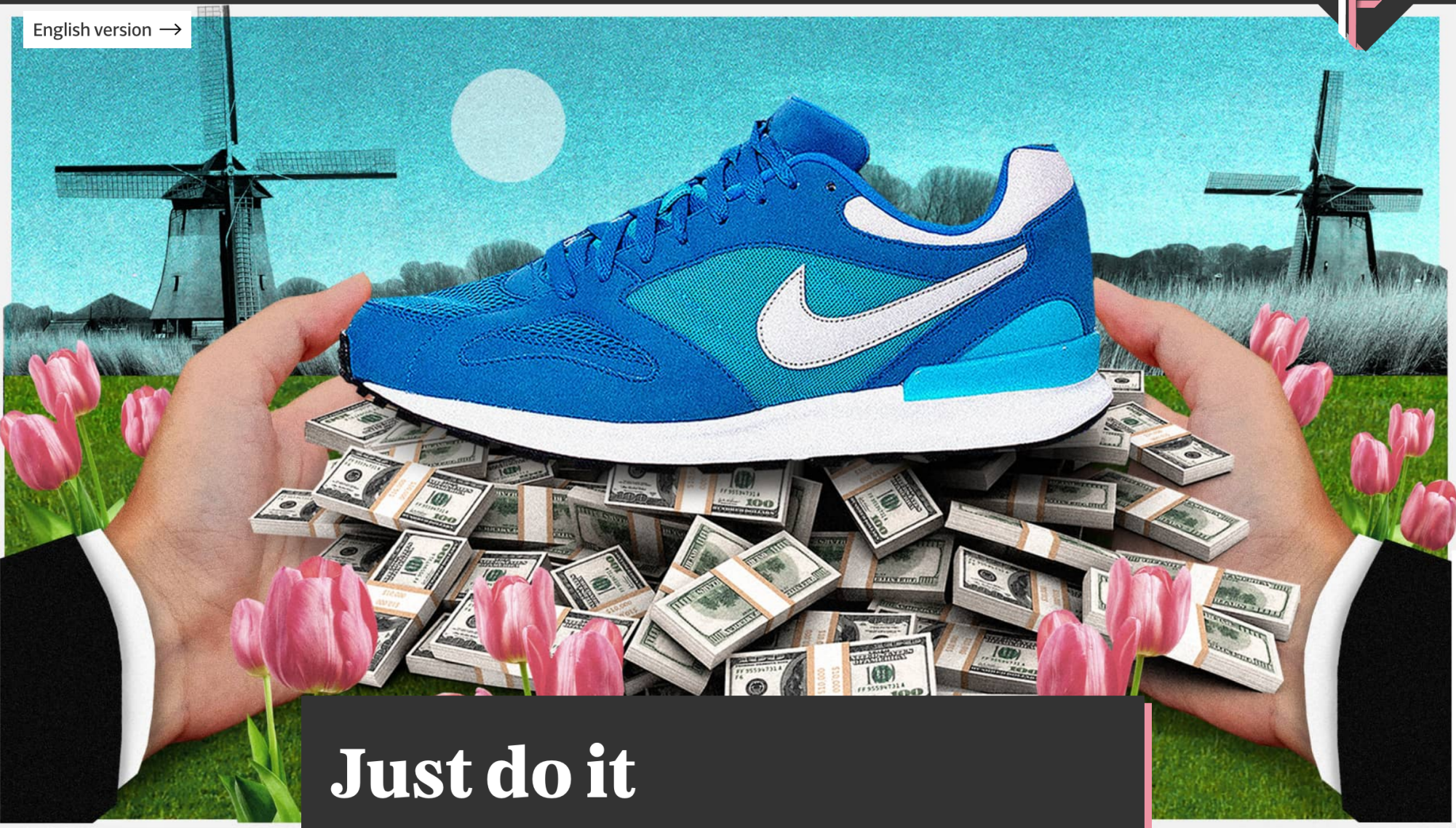
Discussed Applications & Topics in the Lecture

1. IMDb
2. NSA
3. Basics: Query Optimisation
4. Trading, Banking, Ticket Systems
5. From Databases to the Web
6. **Data Journalism**
7. Data in the Wild



https://www.zeit.de/politik/ausland/2019-05/parlamentswahlen-eu-laender-wahlergebnisse-europakarte

English version →

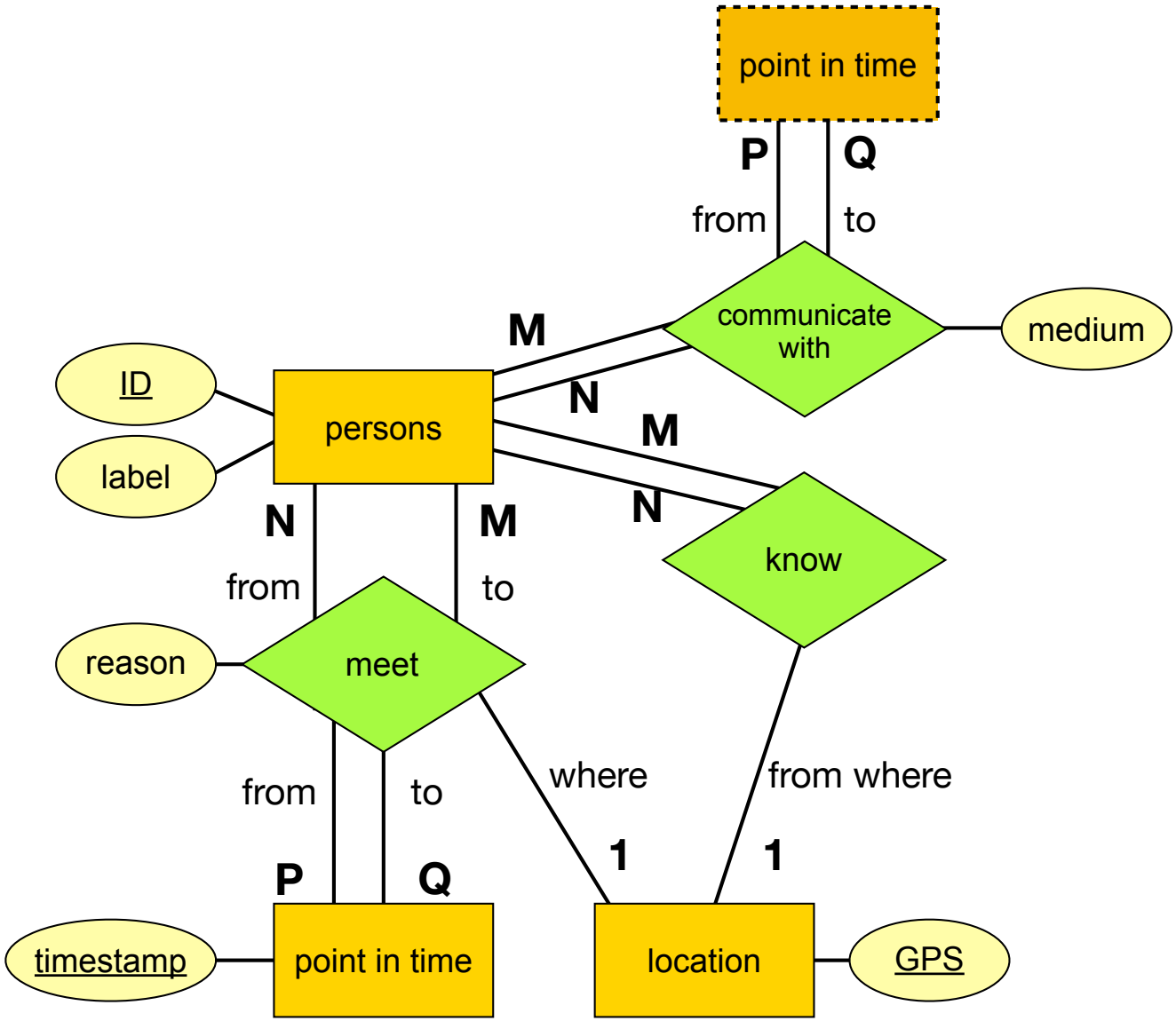


Just do it

Nike ist bei sportlichen Wettkämpfen auf der ganzen Welt präsent. In einer Disziplin aber ist das Unternehmen selbst kaum zu schlagen - im Vermeiden

https://projekte.sueddeutsche.de/paradisepapers/wirtschaft/nike-und-die-niederlande-prellen-den-deutschen-staat-e116625/

A Graph in the Relational Model: Third Approach



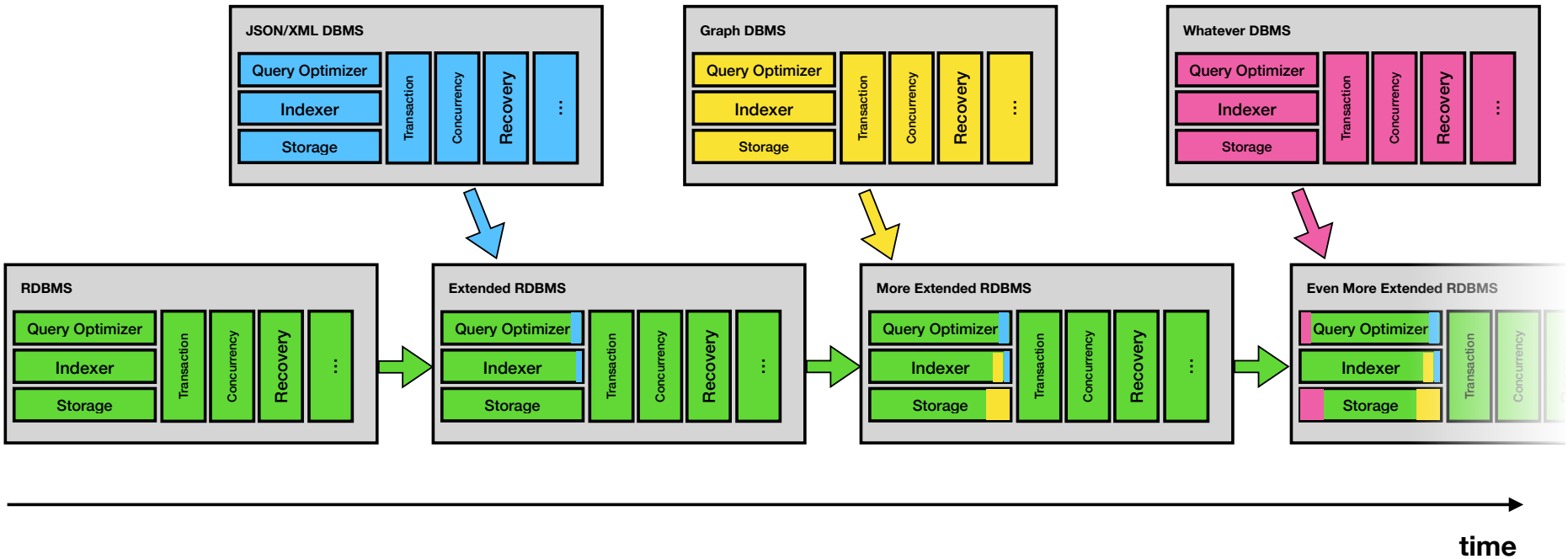
Integration of Cypher in SQL

Integration by means of a function call:

```
SELECT * FROM cypher('graph_name', $$
   MATCH (v)
   RETURN v
$$) as (v agtype);
```

The function cypher() generates a table as return value and can be integrated into the existing query optimiser very easily as a sub-query.

Development of Specialised DBMSs Over Time



- (claimed) advantages through specialised DBMSs are usually only relevant for a transitional/short period of time
- RDBMS vendors each extend their systems so that the advantages of specialised DBMSs fade over time
- however, not all extensions from research always make it into freely available DBMSs

Data Journalism

Question 1

How do we manage graphical data?

ER/relational model

Question 2

How do we query this data?

Cypher (preferably fully integrated in relational DBMS)

Main Learning Objectives

Pivot tables

basic principle; a heat map is a special case of a pivot table

In ER and RM we can model everything

Always. For every application.

Have you found a counter-example? Then come to my office hours and we'll talk about it ;-)

`WITH RECURSIVE` in SQL

implementation, `UNION` vs `UNION ALL`, graph queries using it

Main Learning Objectives

Cypher

basic principle, simple queries, `MATCH`, `RETURN`, ASCII syntax, graph "schema", schema first vs schema later

SQL Injection

basic problem: string concatenation to create SQL query; workaround: sanitise user input (in application and/or DBMS) and/or prepared statements.

Basic safety measures

physical separation of the (partial) database, snapshots, monitoring, checking default accounts and ports, hiding database connection from web browser, automatic testing, testing, testing, ...

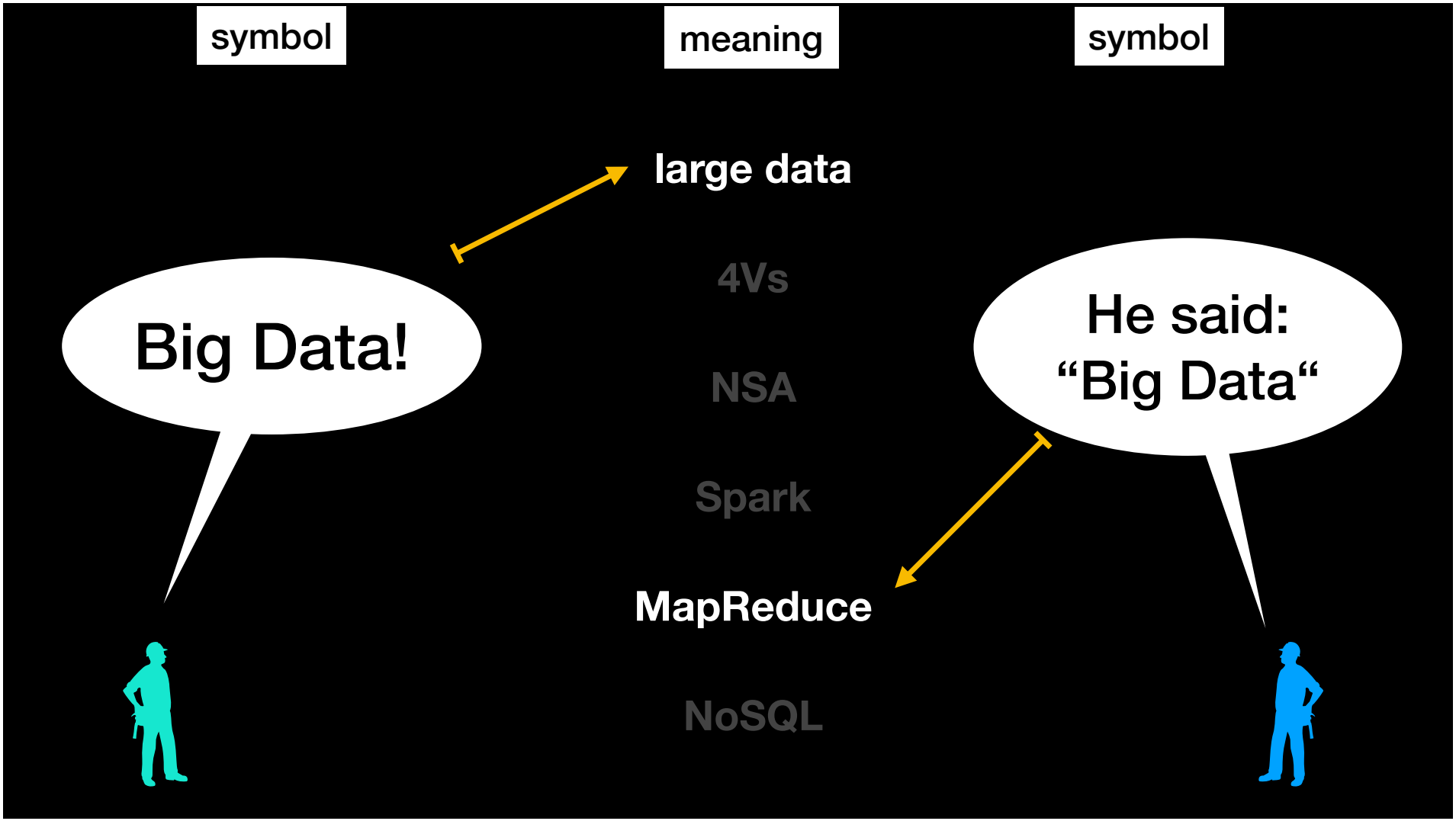
Discussed Applications & Topics in the Lecture

1. IMDb
2. NSA
3. Basics: Query Optimisation
4. Trading, Banking, Ticket Systems
5. From Databases to the Web
6. Data Journalism
7. **Data in the Wild**

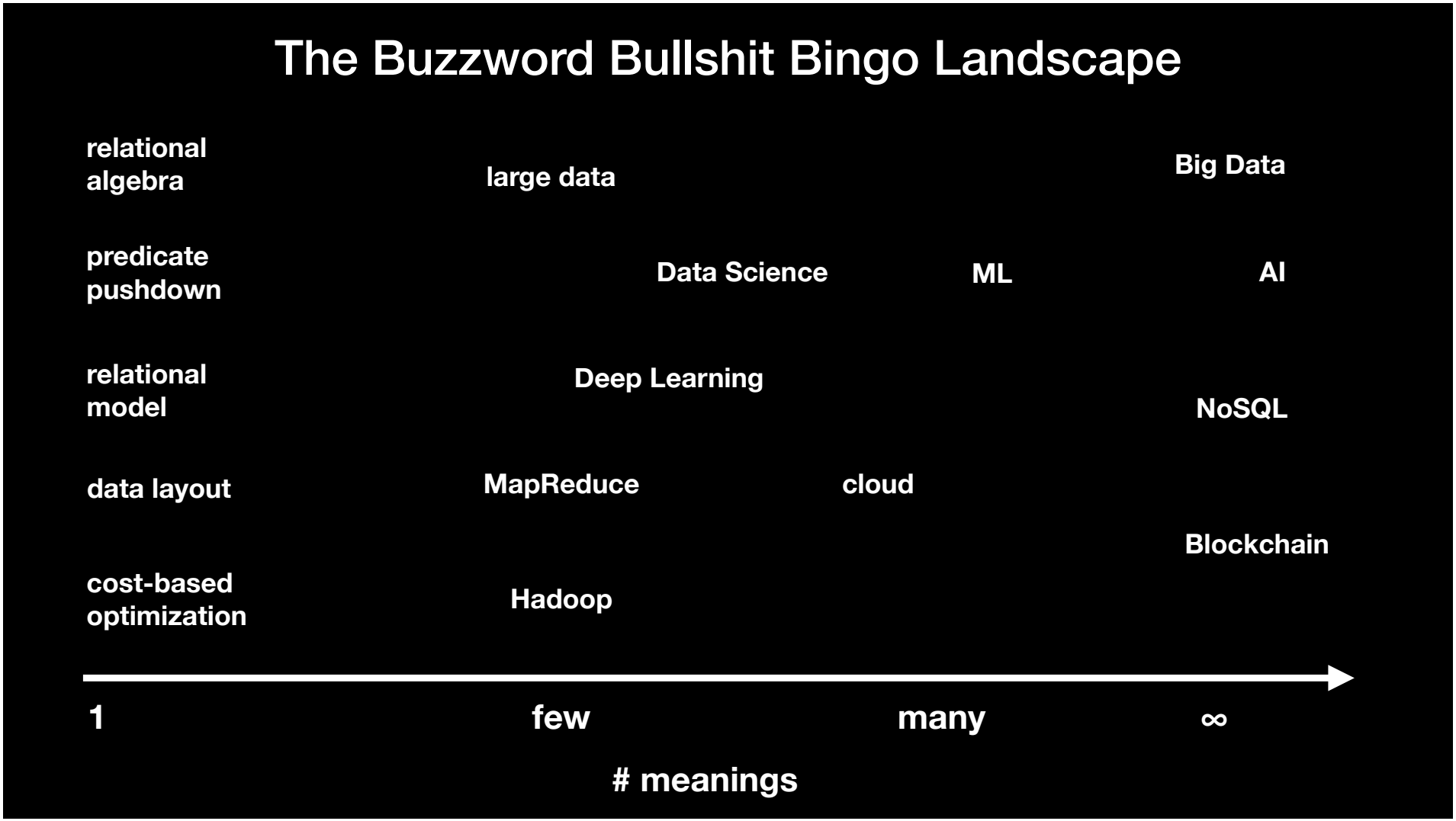
Uni vs Reality (Part 1)

Uni	Reality	Note
ER	No modelling, no interaction with customers, prefer to get started straight away; alternative: another data model: XML, JSON, OO, Graph; no sub-schemas, no logical data independence	logical data independence (LDI)
RM	Tables, iteratively extended over years (attributes and tables continuously added), partially hidden by object-relational mapping (ORM) wrappers (like e.g. in Django, but can also be separated!), many redundancies, no normalisation, many legacy fields (who actually uses this field?), no views, too broadly defined domains	subsequent normalisation, ORM, Django, LDI
RA	Some programming library that reinvents the wheel (e.g. Pandas), based on CSV files; queries formulated procedurally in library, hard-coded query plans	hard-coded vs Spark

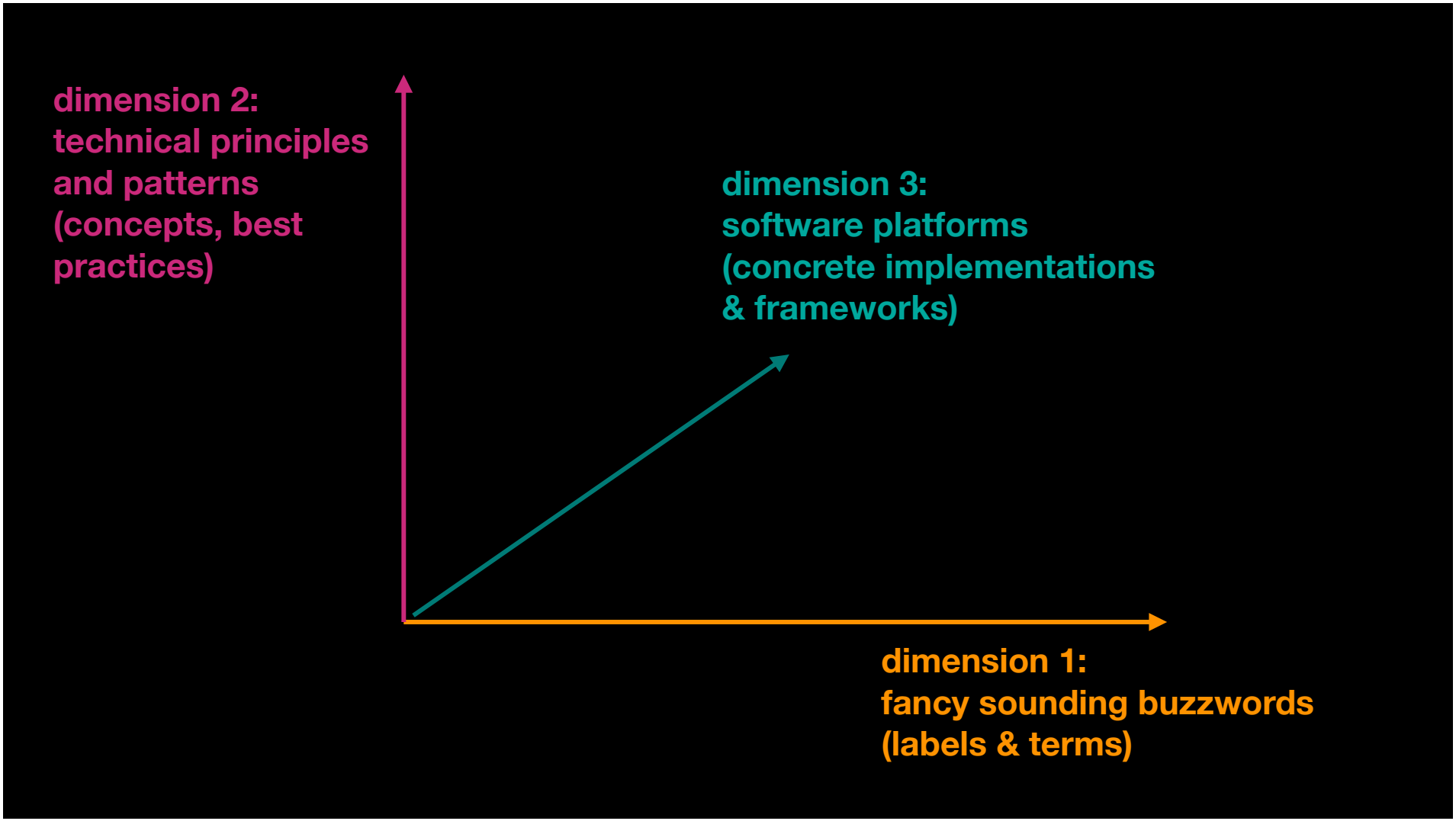
Avoid Confusion Through Ambiguous Terms!



If Possible, Use Only Unambiguous Terms!



Never Confuse These Three Dimensions!



Main Learning Objectives

Abstraction levels of a DBMS

physical data independence, logical data independence, ORM

Reverse engineering of databases

logical data independence, normalisation, hard-coded plans/programs, where to implement functionality?

Best engineering practices for databases

ECA und trigger, deployment environment/staging, physical design advisory

Linguistic confusions

buzzword bullshit bingo, concept vs technology

What's Next?

Proseminar: Techniques for Building Scalable and Robust Web Applications, WS 23/24

This proseminar will continue where IMDb (Part 3) ended.

Question: How to deploy, scale, test, and (performance) monitor a Web Application (WA)?

We will look at state-of-the-art techniques and tools. Organized as a block-seminar. Each student must give a presentation, **and** a hands-on demo.

Major focus in your presentations will be given to how much you explain techniques along Dimension 2: technical principles and patterns (concepts).

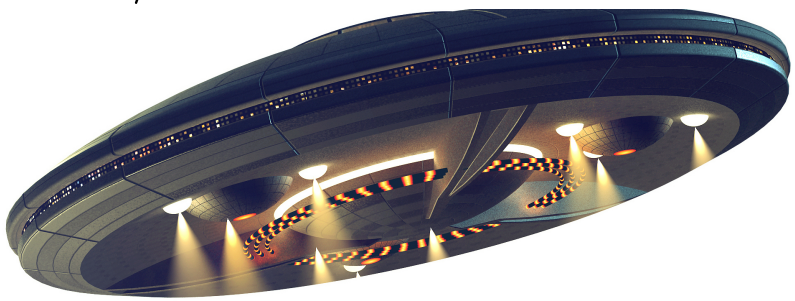
What will you learn?

1. How to give a presentation.
   (extra lecture at the beginning of the proseminar included)
2. How to conceptualize, i.e. how to drag developer gibberish to dimension 2.
3. A lot of technical stuff about Web Applications.

Outlook Core Lecture in WS 24/25

No core lecture in WS 23/24, but in WS 24/25.

PostgreSQL



Modern DBMS

Questions:

- How does an aircraft turbine work in detail?
- How do we fly at a hundred times the speed of light?
- Are there wormholes? And how do we fly through them?
- How can we guarantee the safety of the passengers? Even in the event of a crash?
- How do we build a distributed database system the size of a planet, a galaxy, the universe?

Tutors for Big Data Engineering 2024

Criteria

If you:

1. achieve at least a 1.7 in the final exam,
2. have a talent for explaining technical content, and
3. would like to join our team in 2024,

Then: apply directly to me.

The content of the lecture in 2024 will not differ significantly from 2023 (modulo minor improvements here and there).

Tasks

1. giving a tutorial,
2. participation in weekly team meeting,
3. (co-)conception of exercises,
4. correction of exercises and exams.

Good Luck in the Exam!

I hope you enjoyed it and learned a lot that will help you in your professional career. If you end up working with databases somehow, feel free to drop me a line and tell me about your positive and negative experiences.

Prof. Dr. Jens Dittrich

and the entire lecture team:

Joris Nix, Despina Constantinidou, Elina Celik, Naya Rudolph, Nicolas Regel, Robert Rabbe, Simon Rink