



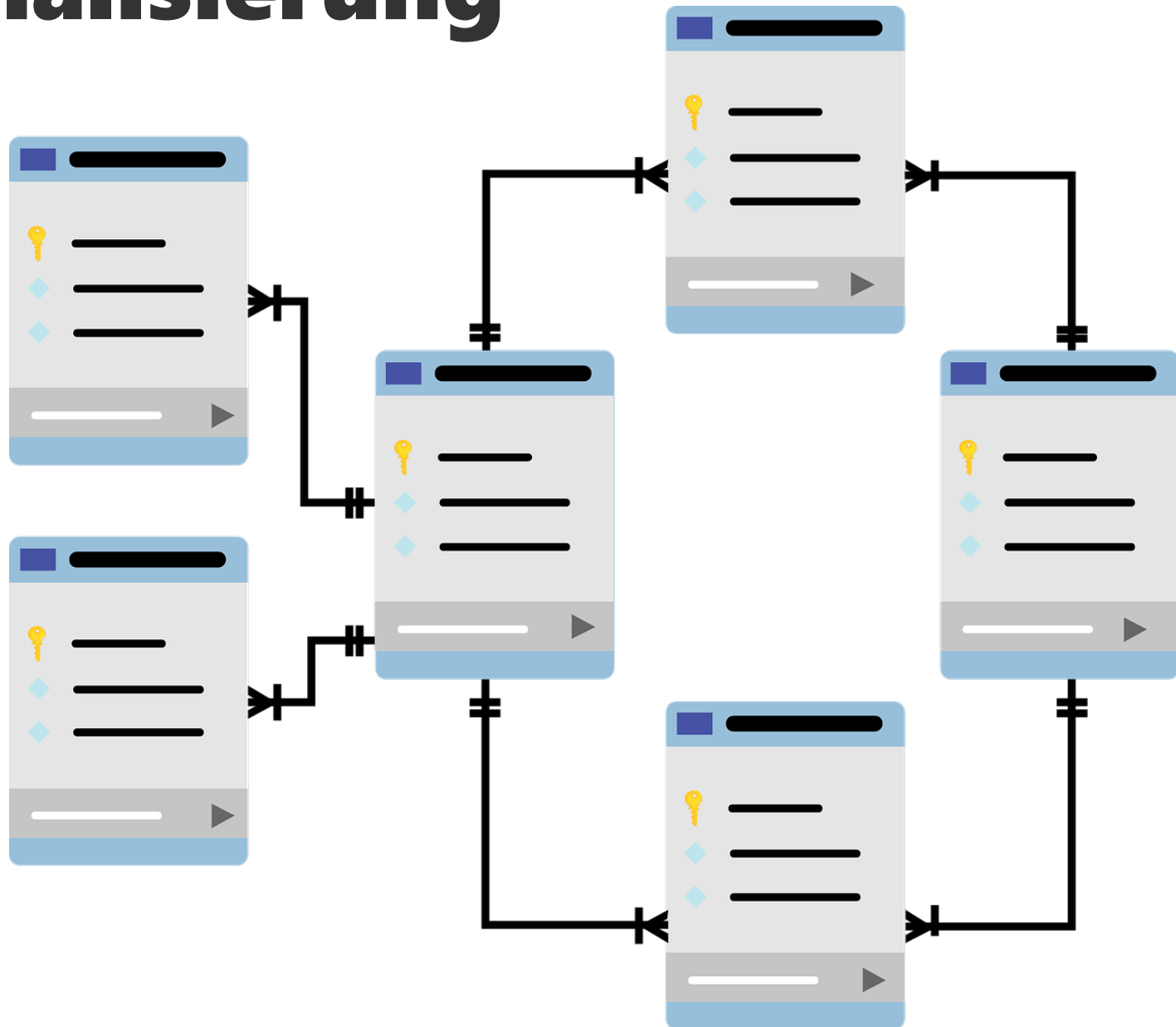
Voll Normal!

Unbürokratische Datenmodelle

Perspektive



Normalisierung

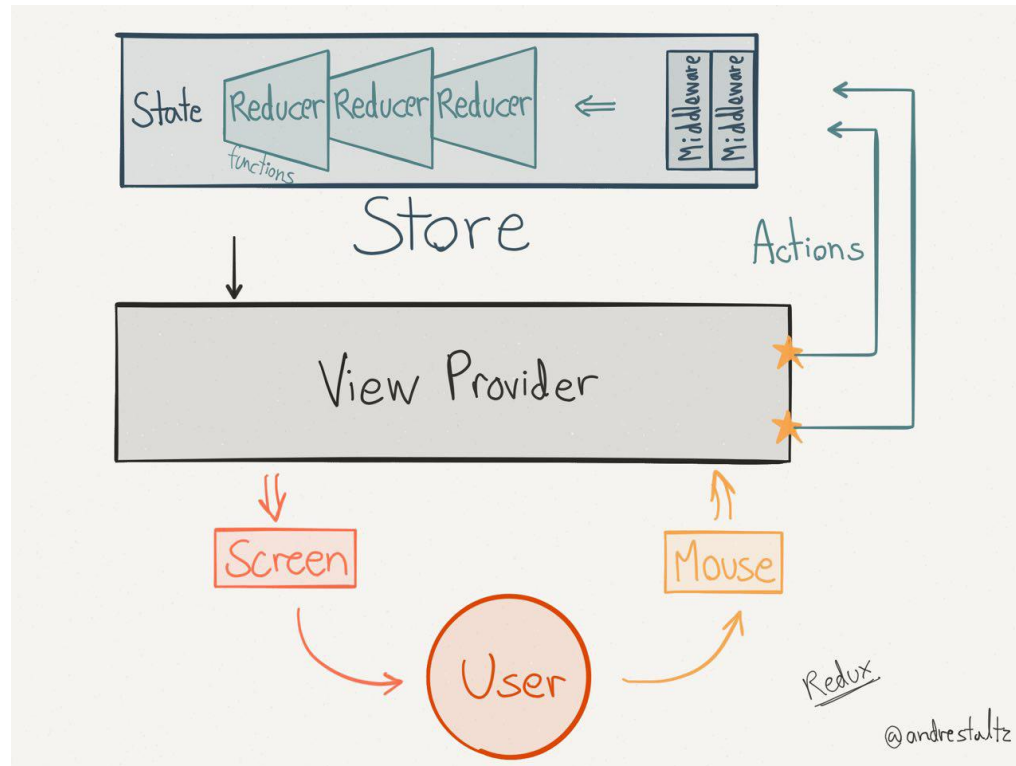




Wikipedia

„Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity.“

Unidirectional UI



<https://staltz.com/unidirectional-user-interface-architectures.html>

Redux



Normalizing State Shape

- Duplizierte Daten updaten ist schwer
- „trying to update a deeply nested field can become very ugly very fast“
- Geschachtelte Daten vertragen sich nicht gut mit der Kombination Observability/Immutability

↑
Coupling


<https://redux.js.org/recipes/structuring-reducers/normalizing-state-shape>



Performance

„For maximum rendering performance in a React application, state should be stored in a normalized shape [...]“

[Haha, „maximum“...]

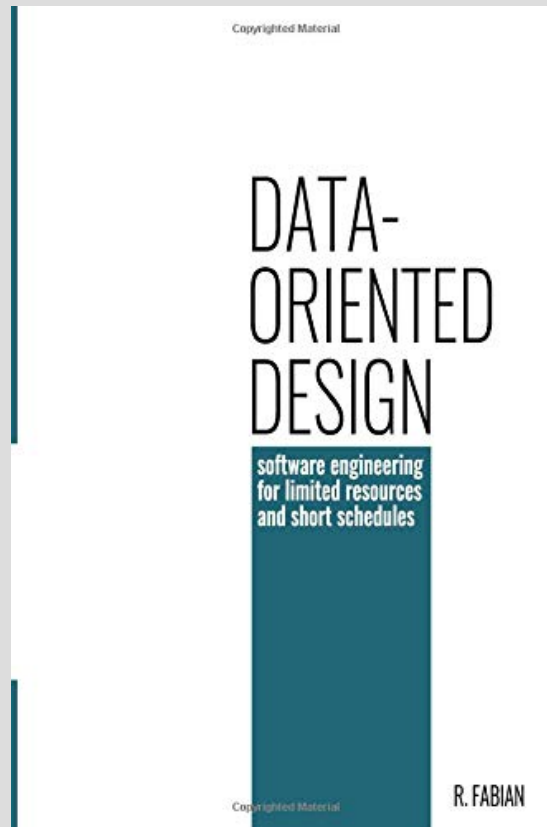


C++?!



<https://isocpp.org>

Data-oriented design



<http://www.dataorienteddesign.com>


OOP is dead?

cppcon | 2018
THE C++ CONFERENCE • BELLEVUE, WASHINGTON


Performance analysis

	OOP	DoD
Animation Tick time average	6.833 ms	1.116 ms

DoD Animations are **6.12x** faster



CppCon 2018 | @stoyannk 38



STOYAN NIKOLOV

OOP is dead, long live
Data-oriented design

<https://youtu.be/yy8jQgmhbAU>

Wieder normal?

- [Data-Oriented Design](#)
 - [It's all about the data](#)
 - [Data is not the problem domain](#)
 - [Data and statistics](#)
 - [Data can change](#)
 - [How is data formed?](#)
 - [What can provide a computational framework for such complex data?](#)
 - [Conclusions and takeaways](#)
- [Relational Databases](#)
 - [Complex state](#)
 - [What can provide a computational framework for complex data?](#)
 - [Normalising your data](#)
 - [Normalisation](#)
 - [Primary keys](#)
 - [1st Normal Form](#)
 - [2nd Normal Form](#)
 - [3rd Normal Form](#)
 - [Boyce-Codd Normal Form](#)
 - [Domain Key / Knowledge](#)
 - [Reflections](#)
 - [Operations](#)
 - [Summing up](#)
 - [Stream Processing](#)
 - [Why does database technology matter?](#)

Also?

- Einfacher zu Programmieren
- Korrektheit ist leichter
- Kuppelt weniger
- Ist performanter!



Meta-Architektur

- Gute Typen!
- Werkzeuge für den Rest
- Der C++ Hammer
- Kapselung hier wichtig!

Vokabular

Datenmodell

- Komposition (structs?)
- Nur Daten in Containern
- Zusammenhänge durch Handles (IDs, ABCs)
- Gehört „der Applikation“
- Nicht kapseln!

Transformationen

- Zustandslos*
- N Tabellen auf M Tabellen
- OOP / Polymorphismus
- Behavioural Patterns
- Die meiste Komplexität?

Und nun?

- Einfach starten!
structs, Funktionen, `std::vector`
- OOP Libs als Daten sehen? (Qt?)
- Daten statt Funktionsaufrufe
- „Tiefe“ Patterns einschränken (hierarchical structures, DI, Callbacks)