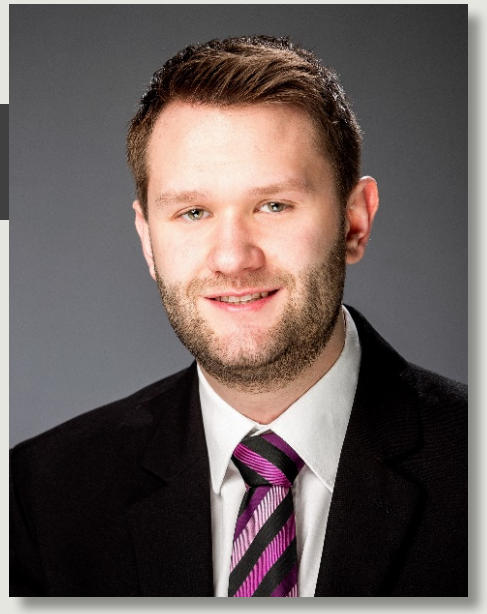
DRYING UP THE DATA SWAMP

Vernetzung von Daten mittels iQser GIN Server

dibuco
solutions for the digital age

Florian Pfleiderer



- Solution Engineer bei dibuco
- Schwerpunkte Big Data und Cloud Architekturen
- Erfahrungen in der Produktentwicklung

- Berater für Entwicklung einer Big Data Middleware
- Produkt GIN Server (= Global Information Network)

Agenda

1. Warum eigentlich „Data Swamp“?
2. GIN Server als Lösung
3. Herausforderung Big Data Entwicklung
4. Lessons Learned

Warum eigentlich „Data Swamp“?

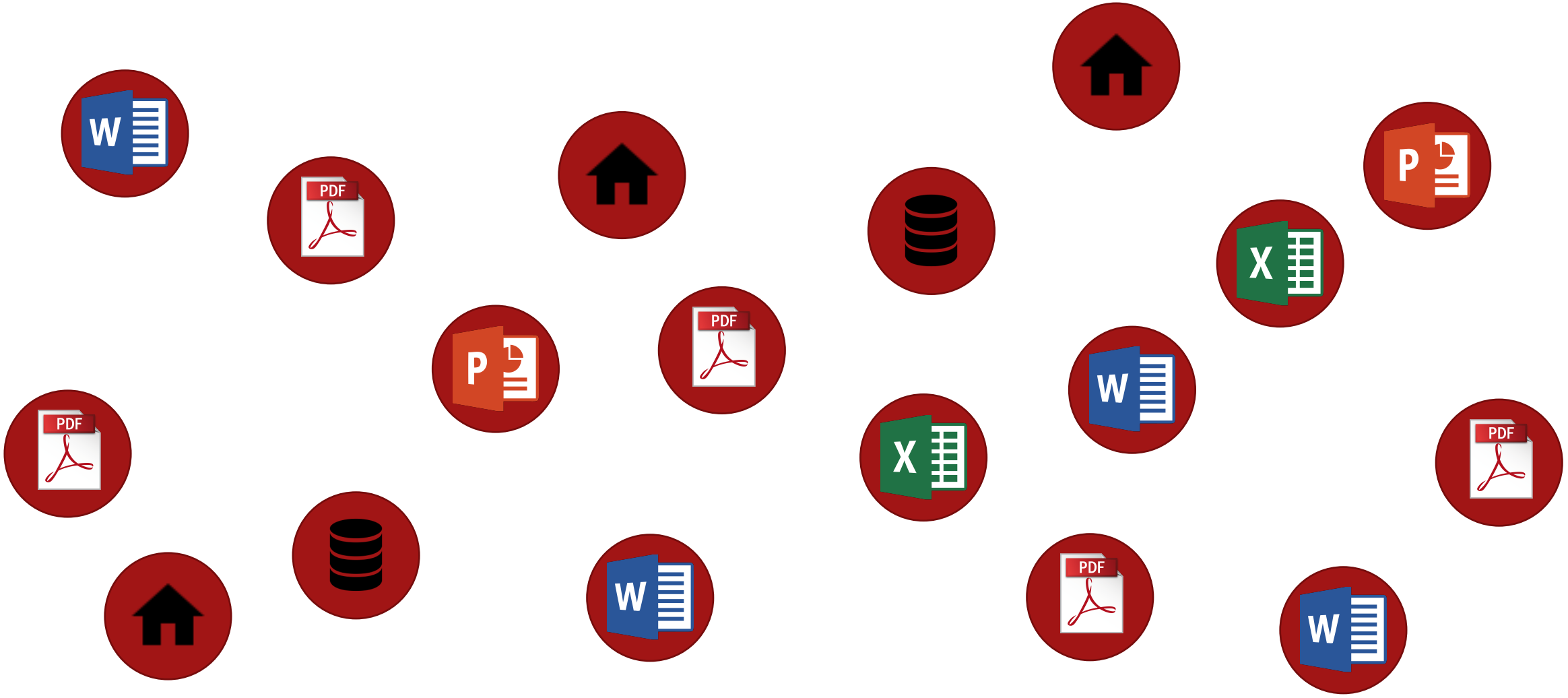
Die Idee des Data Lake stellt Unternehmen in der Praxis vor große Herausforderungen. Oftmals verkommt er nach und nach zum Data Swamp.

dibuco
solutions for the digital age

Degeneration des Data Lake

- Heterogene Daten aus zahllosen Quellen können nicht verstanden werden
- Datenqualität ist nicht bekannt
- Ohne gemeinsames Schema ist es oft schwierig, Anfragen zu stellen
- Gängige Technologien bieten nicht immer gute Lösungen
- Firmen glauben zu Unrecht, sie hätten ihre Daten unter Kontrolle

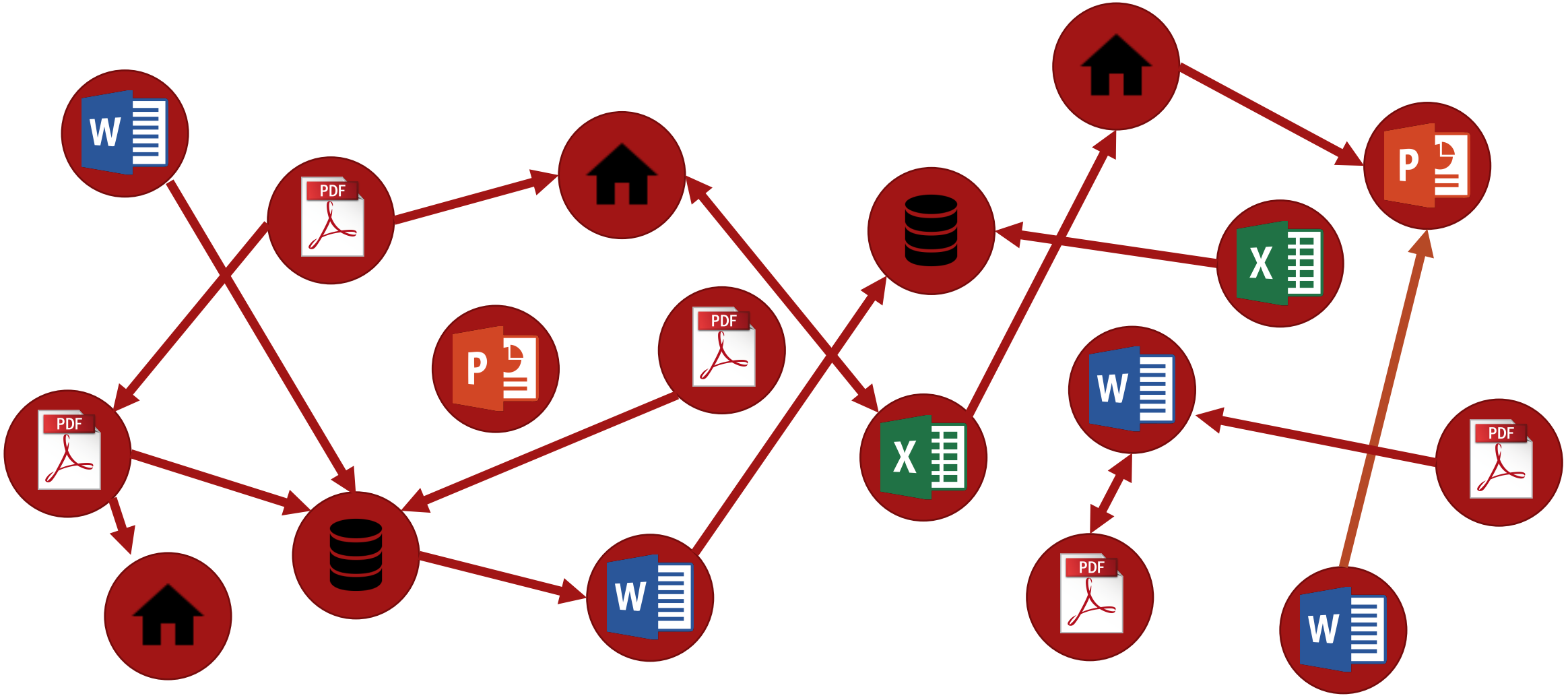
Daten, Daten, Daten…



GIN Server als Lösung

dibuco
solutions for the digital age

Daten, Daten, Daten... Verknüpfungen?



GIN Server

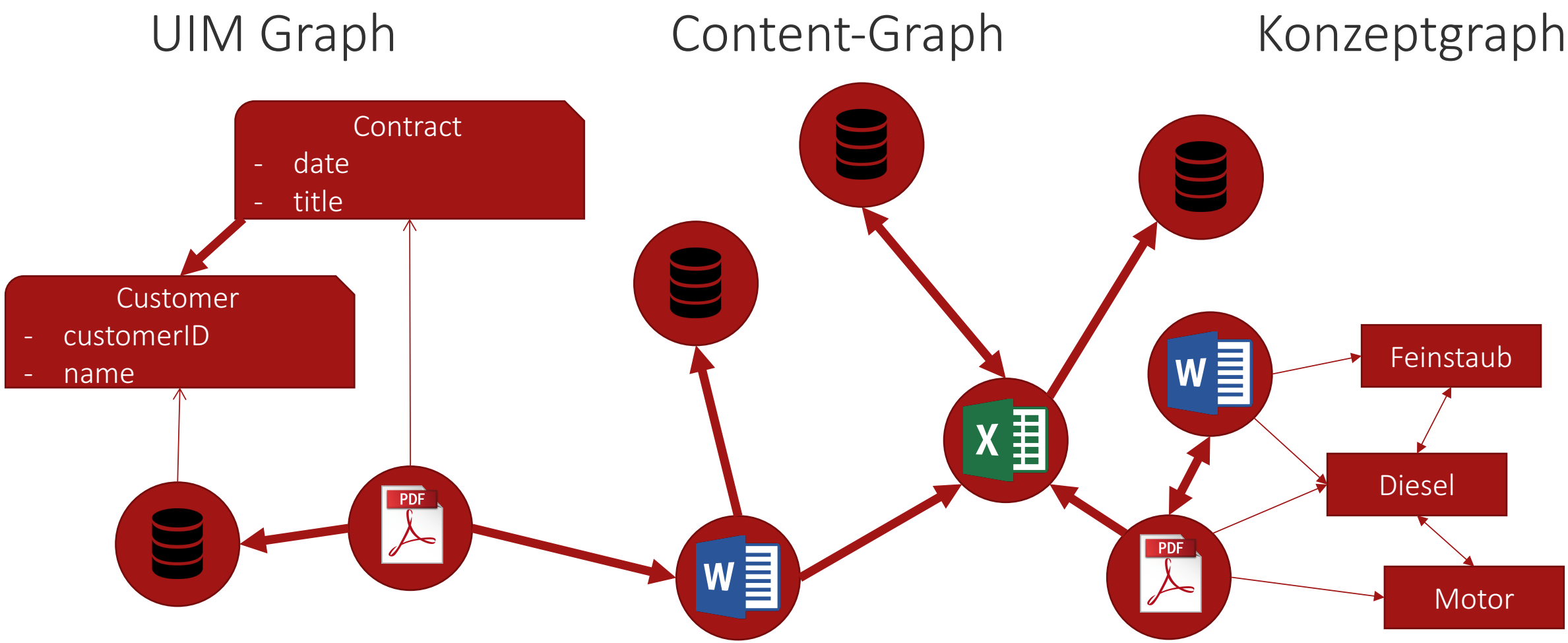
- Repräsentation der Quelldaten als „Content“ im GIN Server
- Daten werden an der Quelle in passendes Format gebracht
- Simples Format: Herkunft, (Meta-)Attribute, Volltext

- Verknüpfungen (Statements) zwischen Contents sind
  - gerichtet
  - begründet
  - gewichtet

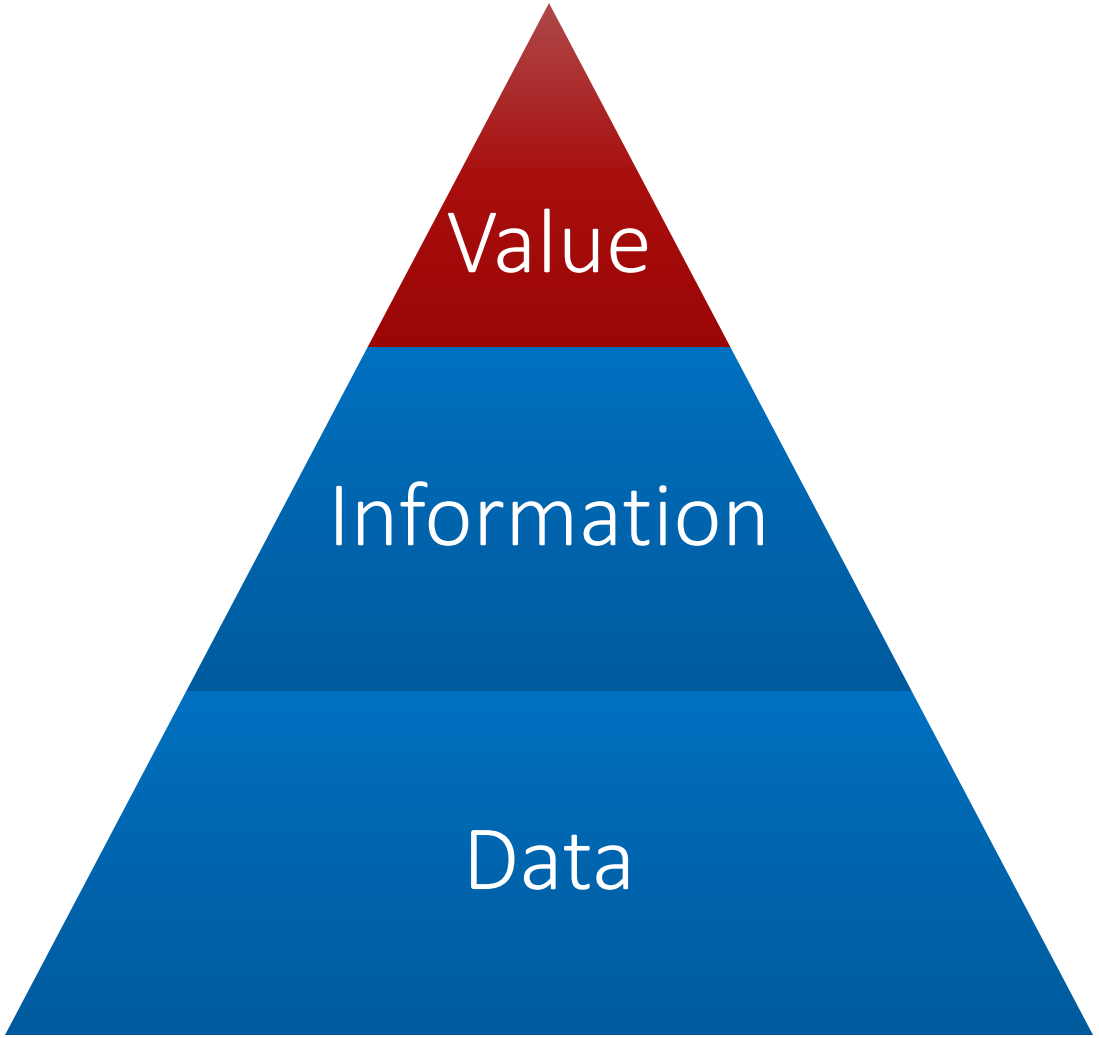
GIN Server

- Verknüpfung unterschiedlicher Daten auf Basis von
  - Text Mining Verfahren
  - Statistischen Analysen
  - NLP
  - ...
- Extraktion der Themen von Dokumenten
- Erstellung und Harmonisierung eines Metamodells

GIN Server

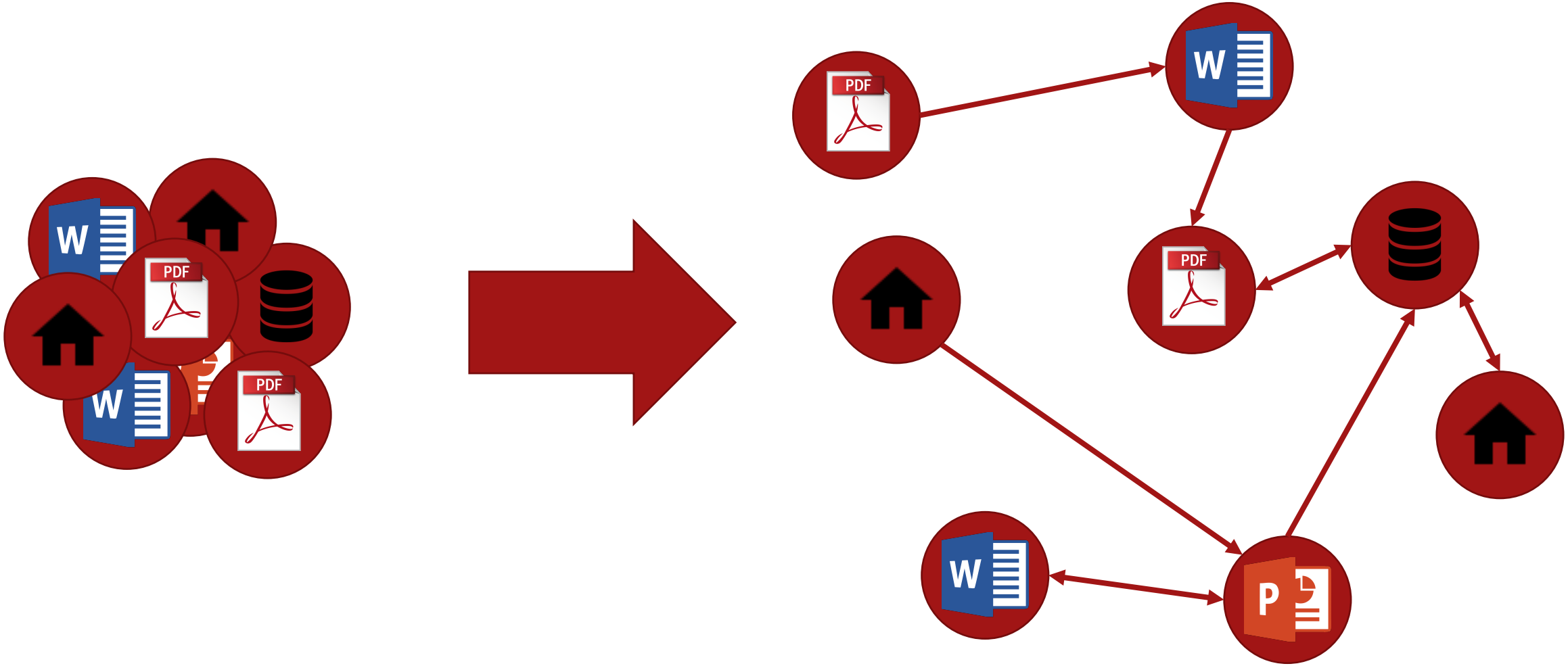

UIM Graph
Content-Graph
Konzeptgraph
Contract
- date
- title
Customer
- customerID
- name
PDF
W
X
W
PDF
Feinstaub
Diesel
Motor

GIN Server



- Von Daten zu Informationen
  - automatische Verknüpfung durch Algorithmen in GIN Server
  - vollautomatische Unterstützung
- Gewinn von Value (=Wissen?) aus Information
  - Queries und kontextbezogene Suchen in den verknüpften Daten

Drying up the data swamp



Herausforderungen

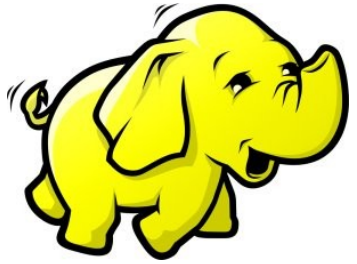
Big Data – Big Complexity

Entwicklung der Architektur

- Vor „Big Data“
  - Monolithische Architektur, relationale Datenbank
  - keine Skalierbarkeit
  - Unzählige Joins

- Der erste große Schritt
  - Microservice-Architektur
  - Batch-Processing (hadoop)
  - Graph Datenbank (Titan DB)
  - => Sehr viel größere Datenmengen beherrschbar

Entwicklung der Architektur: Erkannte Probleme



- **Streaming Problemstellung – Batch Lösung** ⛈️

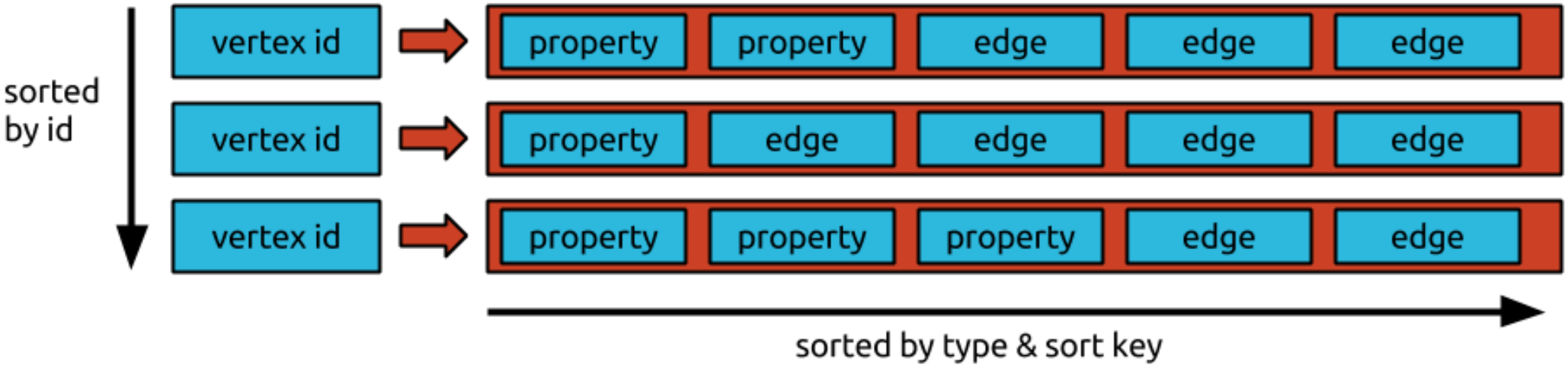
- Optimierungen überwiegend am Batch Import, Vernachlässigung des restlichen Systems
- Batches als Lösung für jedes Problem
- Lange Wartezeiten bis Datenänderungen im System sind

- Ausführungsdauer von Batches wird zunehmend länger
- Teilweise Systemstillstand notwendig für Batchläufe

- Hadoop als Technologie erfordert spezielle Kenntnisse von Dev und Ops

Entwicklung der Architektur: Erkannte Probleme

- Im Voraus berechnete Statements werden alle abgespeichert
  - Enorme Anzahl an Verknüpfungen verbraucht viel Festplattenspeicher
  - Datenbankabfragen werden zunehmend langsamer
  - Superknoten können das System lahmlegen



Entwicklung der Architektur: Erkannte Probleme

- Große Sünde: Gemeinsame Persistenz der Services
  - erzeugt dort immense Last
  - unser eigener kleiner Data Swamp
  - Services können sich nicht unabhängig entwickeln

- Komplexität für Deployment und Konfiguration wächst exponentiell
  - Komplexität verhindert ab einem gewissen Punkt weitere Skalierung
  - Wartung für das Deployment wird immer teurer

Lessons Learned

Entwicklung der Architektur: Lessons learned

Verarbeitung im Batch	▶	Verarbeitung im Stream
Verknüpfungen speichern	▶	Verknüpfungen berechnen
Gemeinsame Persistenz	▶	Unabhängige Services
Betrieb auf dem Host System	▶	Betrieb im Container

Entwicklung der Architektur – Status Quo

- **Eine Streaming-Lösung für ein Streaming-Problem** 💡

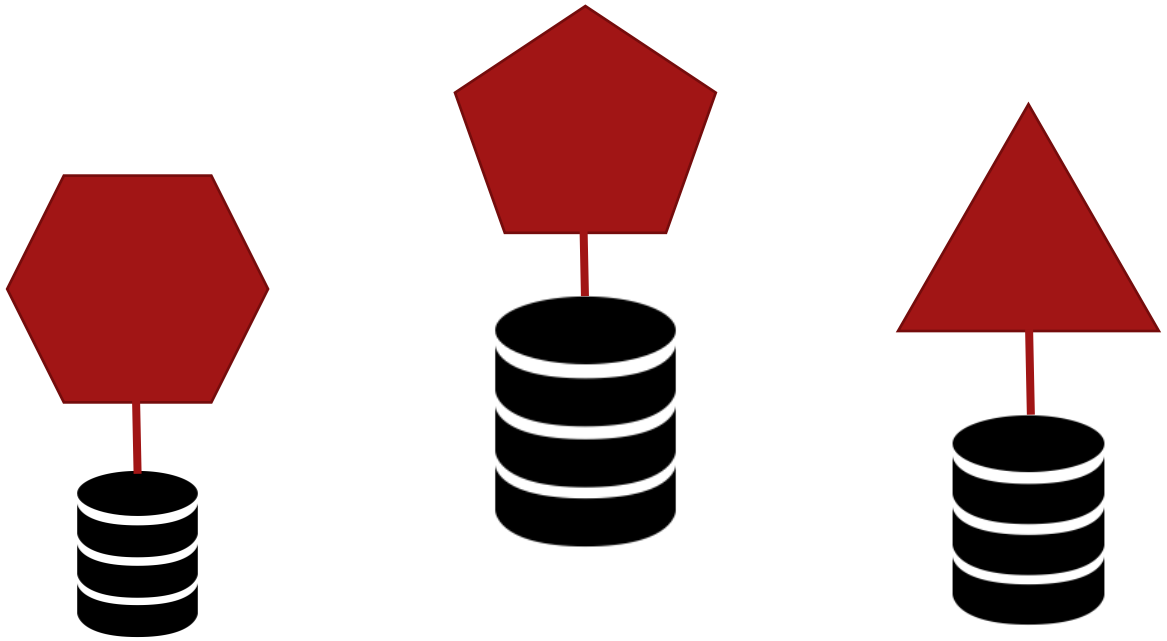
- (Vor-)verarbeitung der Daten im Stream
  - Umsetzung des Datenimports mit Apache Storm
  - Kontinuierliche Verarbeitung macht Ergebnisse schneller abfragbar

- Das System kann durchgehend laufen
- Bei Bedarf können zu Stoßzeiten leicht weitere Ressourcen hinzugenommen werden

- Storm als Technologie auch an anderen Stellen als nur beim Import

Entwicklung der Architektur – Status Quo

- Ad-hoc Berechnung der Verknüpfungen
  - Speicherplatzproblem: gelöst
  - Nur noch Speicherung von statistischen Informationen
  - Berechnungsdauer für Verknüpfungen geringer als DB-Abfragezeiten
- Keine Angst vor Redundanz!



Entwicklung der Architektur – Status Quo

- Weiterer Ausbau der Microservice Archtitektur
  - ermöglicht bessere Skalierung
  - unabhängige Weiterentwicklung der Services wird möglich
  - Last auf die Persistenz(en!) verteilt sich sehr viel besser

- Containerisierung
  - Deployment um ein vielfaches einfacher und beherrschbarer
  - Zentrales Management macht Konfiguration wesentlich simpler
  - Große Hilfe für Skalierbarkeit

Entwicklung der Architektur – Fazit

- Technologien müssen passend zu den Problemen ausgewählt werden
- Viele Probleme sind Streaming Probleme

- Viele neue Technologien im Markt, die vieles erleichtern

- **Aber**: Diese Lösungen kommen mit eigenen Fallstricken

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT

Franz-Schubert Straße 75
70195 Stuttgart
+49 711 699 475 60
info@dibuco.de
www.dibuco.de

dibuco
solutions for the digital age

Quellen

- Microsoft Office Produkt-Logos: Wikimedia Commons (Rezonansowy, ©Microsoft)
- Adobe PDF Logo: Adobe Website (©Adobe)
- Hadoop Logo: @hadoop on Twitter (https://twitter.com/hadoop)
- Storm Logo: @ApacheStorm on Twitter (https://twitter.com/ApacheStorm)
- Titan Data Model: Titan Dokumentation (http://s3.thinkaurelius.com/docs/titan/current/data-model.html)