2. Klassifikation von Mehrrechner-DBS

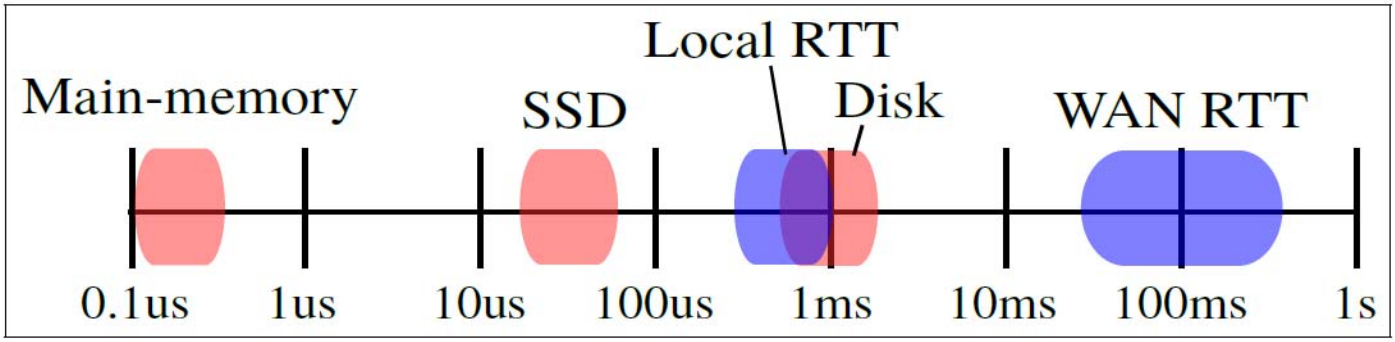
- Merkmale für PDBS/VDBS
  - ***räumliche Verteilung***: ortsverteilt oder lokal
  - ***Rechnerkopplung***: enge, lose oder nahe Kopplung
  - ***Externspeicheranbindung***: partitioniert oder gemeinsam ('shared')
- PDBS-Architekturen
  - *Scale Up* vs. *Scale Out*
  - *Shared Nothing* vs. *Shared Storage*
- Weitere Klassifikationsmerkmale
  - funktionale Spezialisierung vs. funktionale Gleichstellung
  - integrierte vs. heterogene/föderierte MRDBS
- Systemansätze zur Datenintegration
- Grobbewertung von MRDBS-Alternativen

Räumliche Verteilung

- ortsverteilt:
  - unterstützt dezentrale Organisationsstrukturen
  - unterstützt Katastrophen-Recovery (replizierte DB an entfernten Knoten)
  - relativ langsame Kommunikation
    - Signallaufzeiten > 100 ms
    - aufwändige Protokolle (> 10.000 Instruktionen pro Send/Receive)
- lokal:
  - schnelle Rechnerkopplung (gemeinsame Speicher bzw. Hochgeschwindigkeitsnetz)
  - effektive dynamische Lastverteilung möglich
  - bessere Voraussetzungen für Intra-Query-Parallelität
  - einfachere Administration

	Durchmesser	Latenzzeit	Bandbreite (Mbit/s) 1990	2020	Übertragung 10 KB 1990	2020
Cluster	20 m	1 µs	1000	10000	0,1 ms	0,01 ms
LAN	1 km	10 µs	10	1000	10 ms	0,1 ms
WAN	10.000 km	100 ms	0,05	150	1.700 ms	101 ms

Vergleich Latenzzeiten

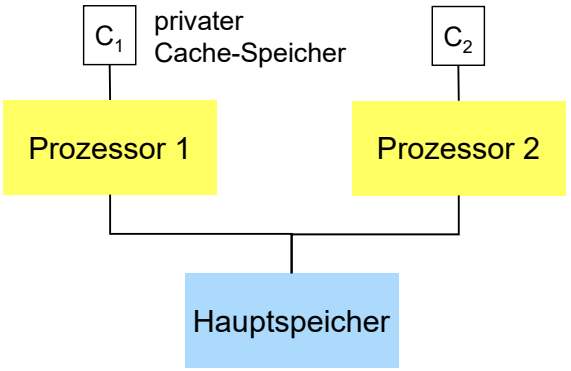


F. Nawab et al: A system infrastructure for strongly consistent transactions on globally-replicated data. IEEE Bull. TCDE, 2017

RTT: Round-trip time

Enge Rechnerkopplung (tightly coupled systems)

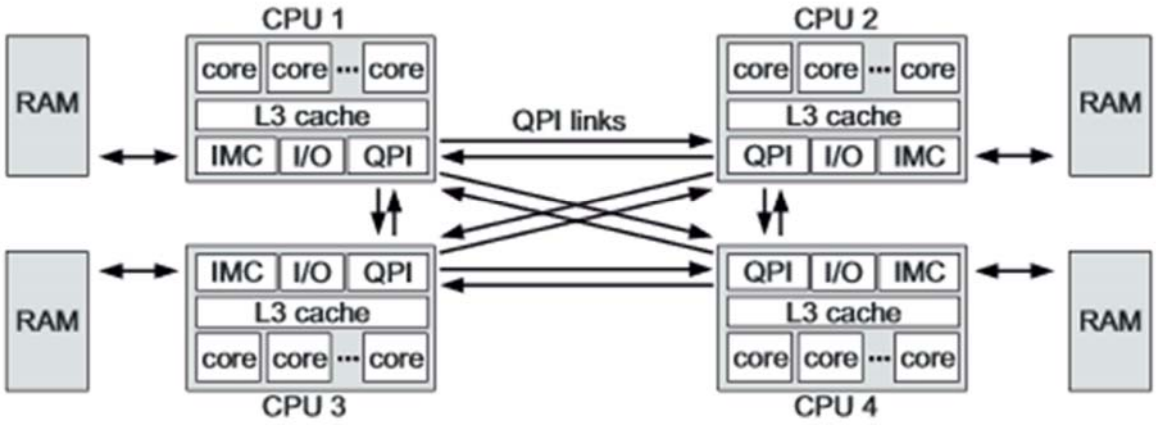
- Eigenschaften
  - gemeinsamer Hauptspeicher
  - 1 Kopie von Software-Komponenten (Betriebssystem, DBMS)
  - HW-Cache pro Prozessor



- Vorteile
  - weit verbreitet
  - effiziente Kommunikation über Hauptspeicher
  - Lastbalancierung durch Betriebssystem
  - Single System Image
- Nachteile
  - mangelnde Fehlerisolation
  - oft begrenzte Erweiterbarkeit und Skalierbarkeit
  - Cache-Kohärenz

NUMA-Architekturen

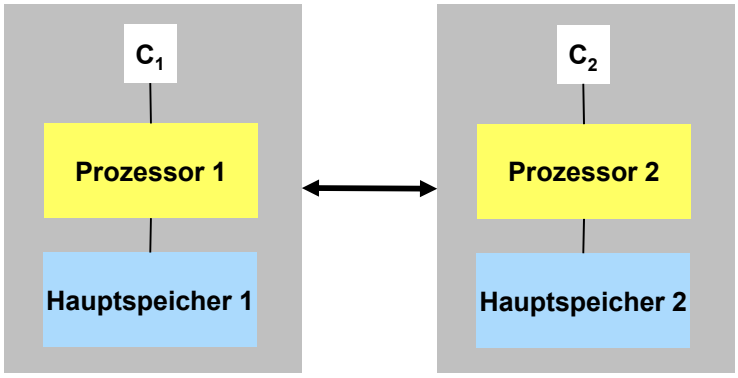
- Non-Uniform Memory Access (NUMA) bei hoher Prozessorzahl
- Beispiel Intel



- Memory-Zugriff über Links wesentlich langsamer
  - Datenpartitionierung und -lokalität innerhalb eines NUMA-Servers verbessert Leistung für DB-Anwendungen

Lose Rechnerkopplung (loosely coupled systems)

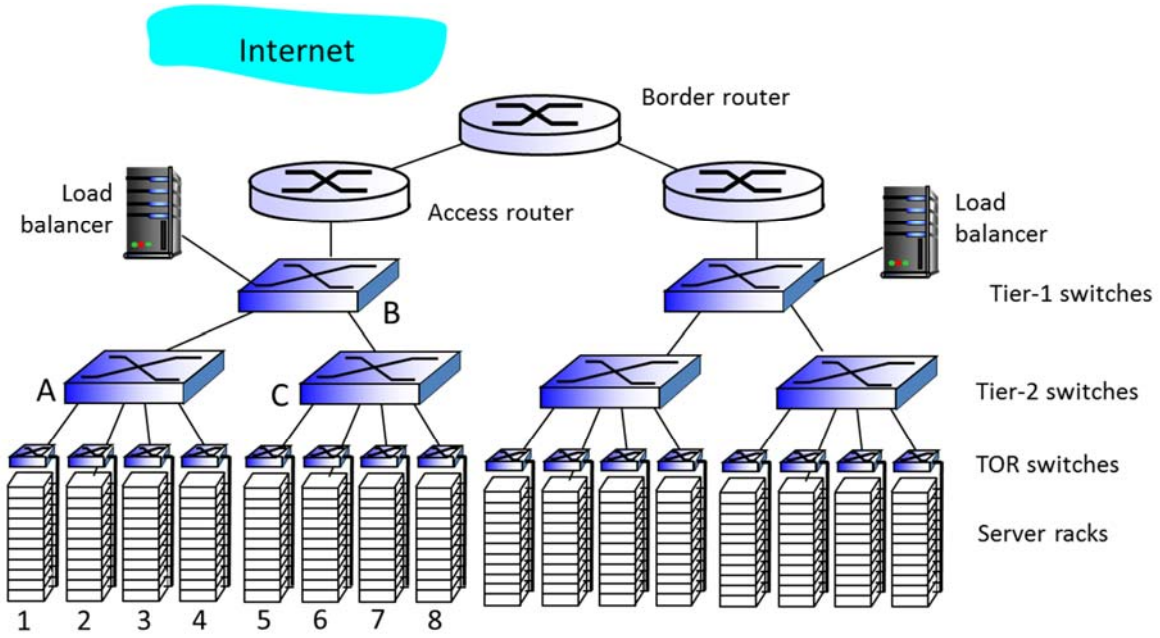
- Eigenschaften
  - n selbständige Rechner (pro Knoten eigener Hauptspeicher, eigene Software-Kopien)
  - Kommunikation über Nachrichtenaustausch



- Vorteile:
  - höhere Fehlerisolation/Verfügbarkeit
  - bessere Erweiterbarkeit
- Nachteile:
  - Nachrichtenaustausch aufwendig (Kommunikations-Overhead)
  - kein 'single system image'

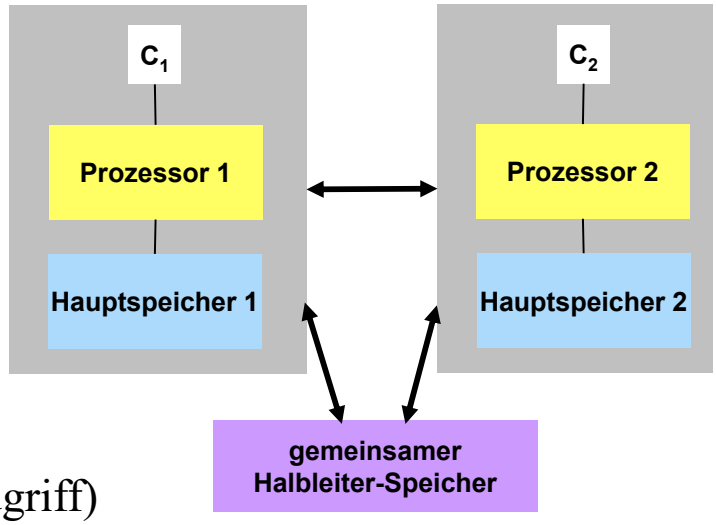
Lose Kopplung im Data Center

- mehrstufige Kopplung mit Server-Racks
  - Tausende von Rechnern
  - schnellere Kommunikation innerhalb Rack



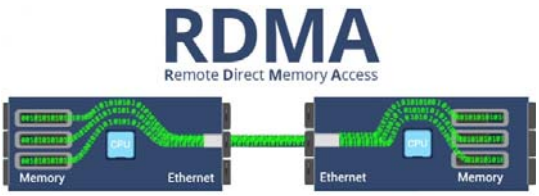
Nahe Rechnerkopplung (closely coupled systems)

- Kompromiß zwischen enger und loser Kopplung
  - effizientere Kommunikation als mit loser Kopplung unter Beibehaltung einer ausreichenden Fehlerisolation und Erweiterbarkeit
- Merkmale
  - n selbständige Rechnerknoten
  - gemeinsame Halbleiter-Speicherbereiche
  - lokale Rechneranordnung
- Speichereigenschaften
  - schneller, synchroner Zugriff in wenigen Mikrosekunden (kein Prozesswechsel während Zugriff)
  - Disaggregated memory oder Spezialrechner (z.B. Coupling Facility in IBM Sysplex)



Remote Direct Memory Access (RDMA)

- direkter Zugriff auf Hauptspeicher anderer Rechner im Cluster
  - Abwicklung über Netzwerkkomponenten, ohne Involvierung des Betriebssystems und von CPUs
  - „Zero-Copy-Networking“
  - Unterstützung für InfiniBand, Ethernet (RDMA over Converged Ethernet)

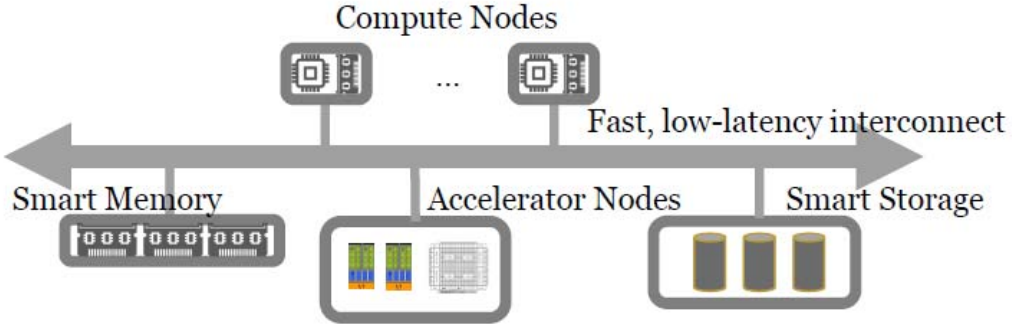


- wesentlich schnellerer Datenaustausch als mit loser Kopplung
  - nur noch ca 10-mal langsamer als Hauptspeicherzugriff

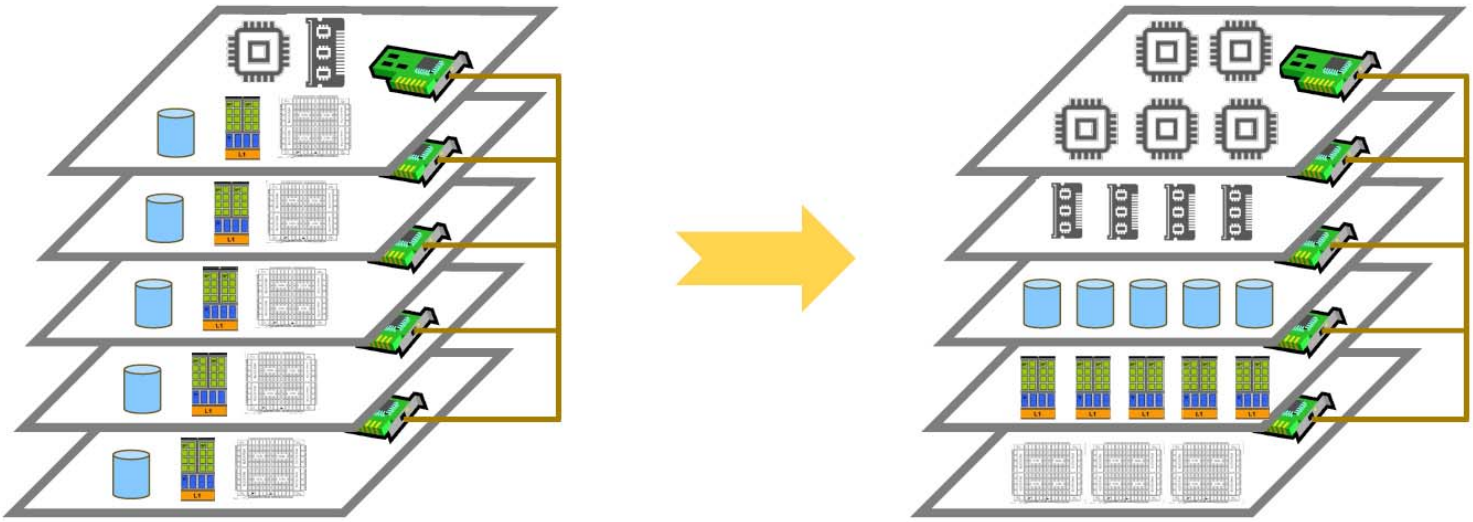
- Alternative zu naher Kopplung
- Nutzung u.a. in Hadoop, Spark, …

Disaggregated Computing

- Trend in **Cloud Data Centers** zur Disaggregation von Ressourcen (CPUs, Memory, storage, GPUs …)
  - bessere Nutzung der Ressourcen
  - größere Elastizität bzgl Wachstum
    - Compute (Storage) kann unabhängig von Storage (Compute) skaliert werden
  - hohe Flexibilität und Kosteneffektivität v.a. bei unterschiedlichen Tenants
- Bsp.: Shared Storage, Disaggregated memory …

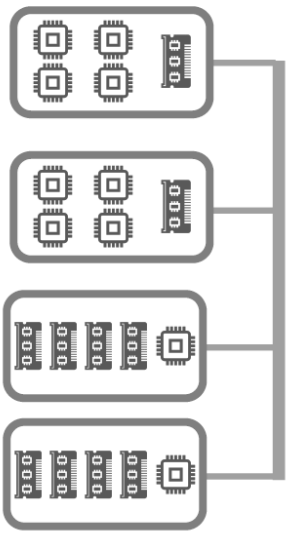


© Özsu 2023 https://www.dropbox.com/scl/fi/326cseumioe1tj5jzyqkp/Disaggregation-2023-12.pdf?rlkey=n450vciaq7yvlvn5f0sexx39e



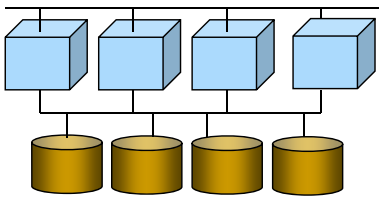
Disaggegrated Memory

- separate Hauptspeicher-Module mit schneller Netz-Anbindung
  - RDMA (Kopieren der Daten über Netzwerk)
  - CXL (Compute Express Link) – erlaubt direkten entfernten Speicherzugriff ohne Belastung der CPU und ohne Datenkopien
- entspricht neuer Variante einer nahen Kopplung



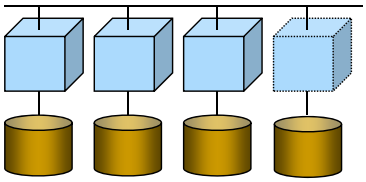
Externspeicheranbindung (Storage)

gemeinsam: jeder Prozessor kann alle Externspeicher / Daten direkt erreichen



- Rechneranordnung: lokal
- Rechnerkopplung: eng, lose oder nahe
- hohes Potenzial zur Lastbalancierung
- ggf. Separierung von Rechnern und Storage (Cloud)

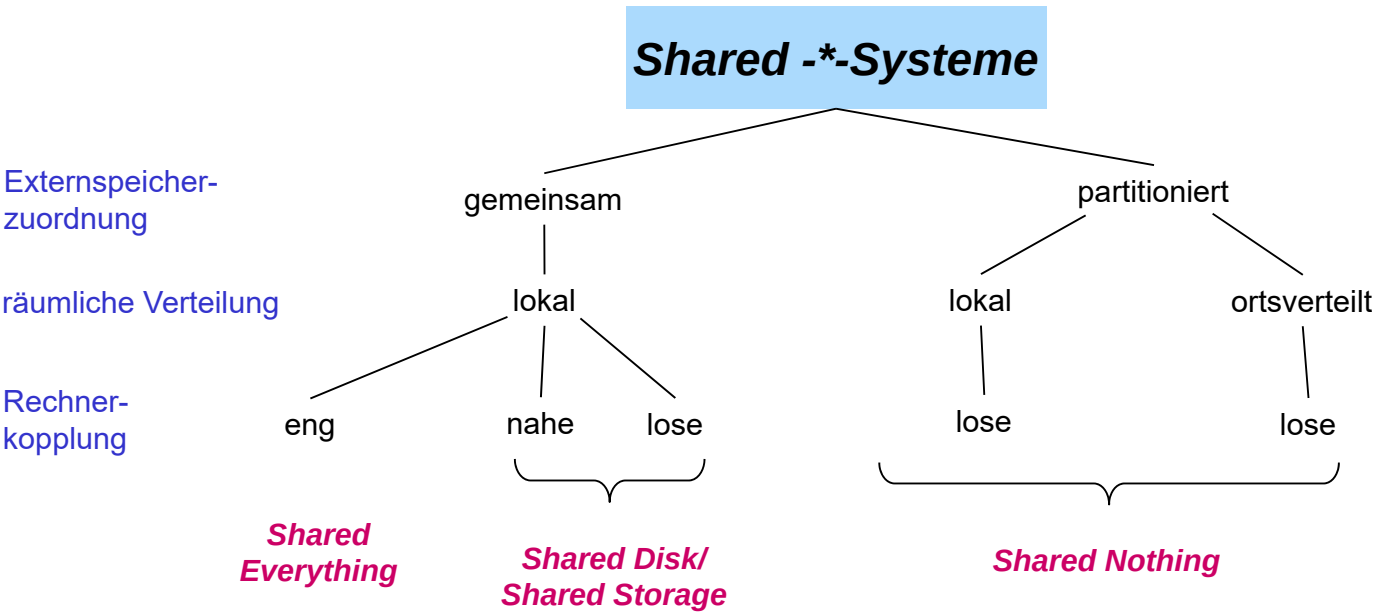
partitioniert: Externspeicher sind primär nur je einem Knoten zugeordnet



- Rechneranordnung: lokal oder ortsverteilt
- Rechnerkopplung: i. a. lose
- verteilte Transaktionsausführung, um auf entfernte Daten zuzugreifen

verteilte/parallele **In-Memory-Datenbanksysteme** ?

- enge Kopplung: gemeinsam genutzte In-Memory-DB
- lose Kopplung: partitionierte In-Memory-DB
- nahe Kopplung mit disaggregated memory



- Parallele DBS: lokal verteilte Shared -*-Systeme
- Verteilte DBS: ortsverteilte DBS
- hybride Ansätze
  - eng gekoppelte Knoten in lose gekoppelten Systemen
  - ortsverteilte Data Center mit PDBS pro Center

3 Stufen der Verteilung

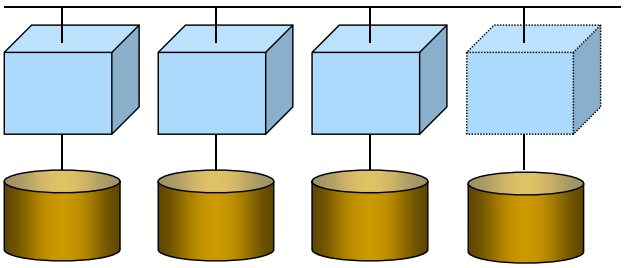
1. Scale Up: mehrere Prozessoren innerhalb von 1 Knoten (Shared Everything)
   - sehr effiziente Kommunikation (über Hauptspeicher)
   - direkter Zugriff auf gesamte Datenbank für alle DBMS-Instanzen; zentrale Datenstrukturen (Sperrtabelle, DB-Puffer, etc.)
   - einfache DB-Administration
   - von allen DBS-Herstellern unterstützt (Microsoft, Oracle, IBM ...)
   - Leistungsfähigkeit reicht für Mehrzahl von Datenbanken
   - begrenzte Erweiterbarkeit und Verfügbarkeit
   - relativ hohe Kosten im High-End-Bereich
2. Scale Out: SN/SD/Hybrid-Cluster
   - hohe Skalierbarkeit durch unabhängige Rechnerknoten (kein gemeinsamer Hauptspeicher, lokale Software)
3. (geo-)verteilte Replikation (für SE, SD/SS oder SN)

Skalierbarkeit: Scale-Up vs. Scale-Out

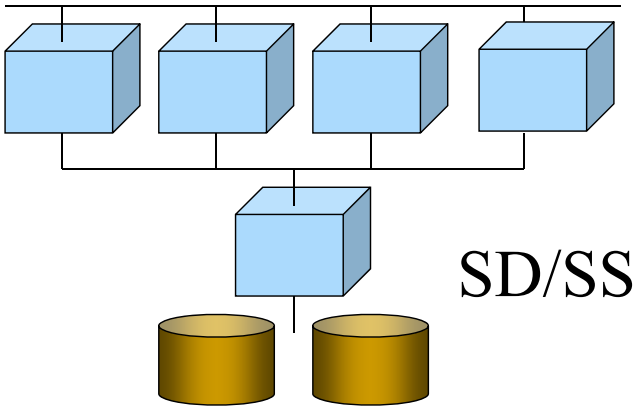
"Scale Up"
- schnellere Mehrprozessorknoten
- Shared Everything

"Scale Out"
- N unabhängige Rechner (z.B. Commodity-Server)
- Hinzufügen neuer Server nach Bedarf
- Shared Nothing oder Shared Disk (Cluster)

Shared Nothing (SN) vs. Shared Disk/Storage (SD/SS)



SN

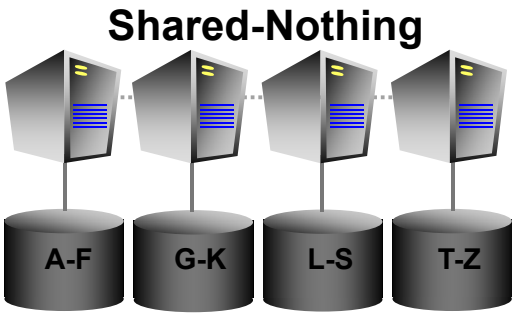


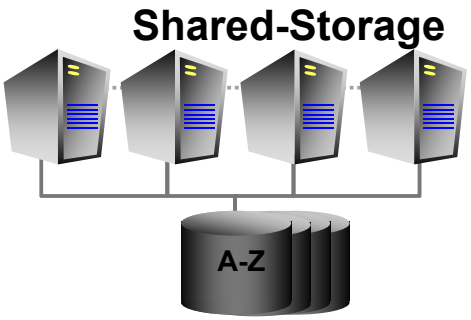
SD/SS

- Teradata
- IBM DB2 (Windows, Unix)
- NewSQL-Systeme wie Google Spanner, Greenplum, Exasol
- viele NoSQL-Systeme (AWS DynamoDB, Azure CosmosDB …)

- Oracle (RAC, Exadata)
- IBM DB2 z/OS, DB2 PureScale
- Shared-Storage Cloud DBS (AWS Aurora, Azure SQL Hyperscale, Alibaba PolarDB, Huawei GaussDB …)

SN vs. SD/SS: Leistungsfähigkeit





statische Datenpartitionierung bestimmt Ausführungsort von DB-Operationen und damit Kommunikationsaufwand

geringe Möglichkeiten zur dynamischen Lastbalancierung

hohe Flexibilität zur Parallelisierung und Lastbalancierung

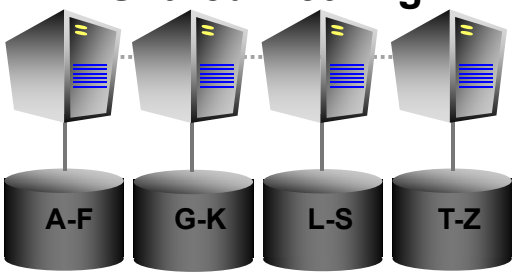
- Erreichbarkeit aller Daten von jedem Knoten

hoher Aufwand für Synchronisation und Kohärenzkontrolle

- Lokalität ermöglicht Einsparungen
- nahe Kopplung reduziert Overhead

SN vs. SD/SS: *Erweiterbarkeit*

Shared-Nothing

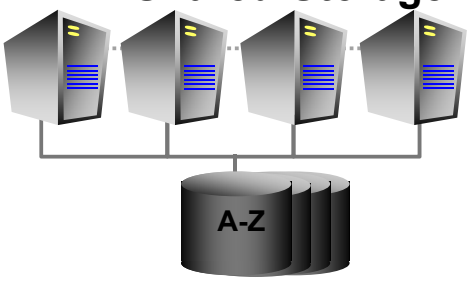


Hinzufügen neuer Knoten hardwareseitig einfach

- viele Knoten sind möglich

neuer Rechner erfordert physische Neuaufteilung der Datenbank (N -> N+1)

Shared-Storage

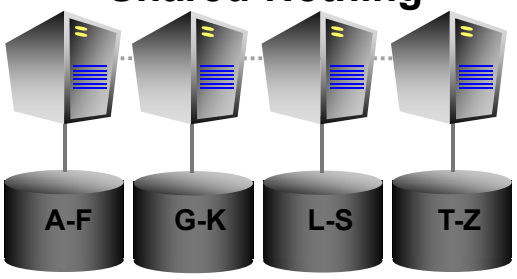


keine physische (Neu-) Aufteilung der DB bei neuem Rechner

direkte Plattenanbindung begrenzt Rechneranzahl

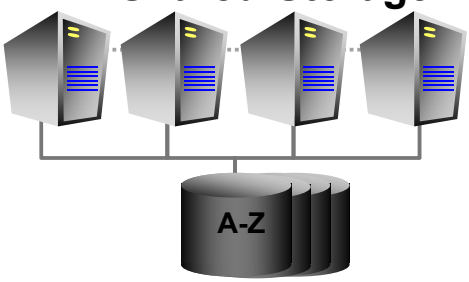
SN vs. SD/SS: Recovery

Shared-Nothing



Übernahme/Recovery der betroffenen Partition durch anderen Rechner vorzusehen (ggf. Überlastungsgefahr)

Shared-Storage



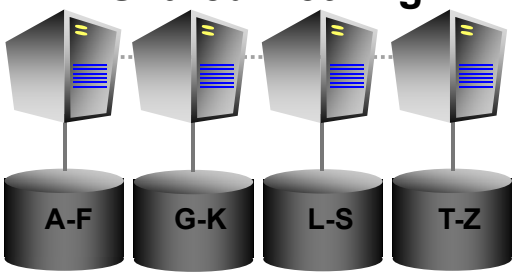
gesamte DB bleibt nach Rechnerausfall erreichbar

komplexe Crash-Recovery

Erstellung einer globalen Log-Datei

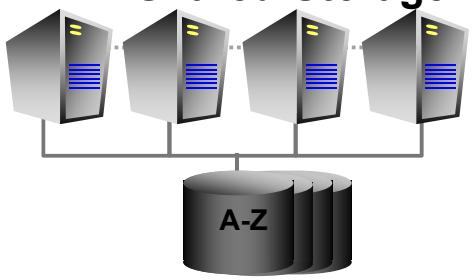
SN vs. SD/SS: *Technische Probleme*

Shared-Nothing



- DB-Partitionierung bzgl. Rechner
- verteilte und parallele Anfrageverarbeitung
- verteilte Commit-Protokoll
- globale Deadlock-Behandlung
- Replikationskontrolle
- Katastrophen-Recovery ...

Shared-Storage



- DB-Partitionierung bzgl. Externspeicher
- parallele Anfragevearbeitung
- globale Synchronisation
- Kohärenzkontrolle
- globaler Log, Crash-Recovery
- Lastbalancierung
- Katastrophen-Recovery ...

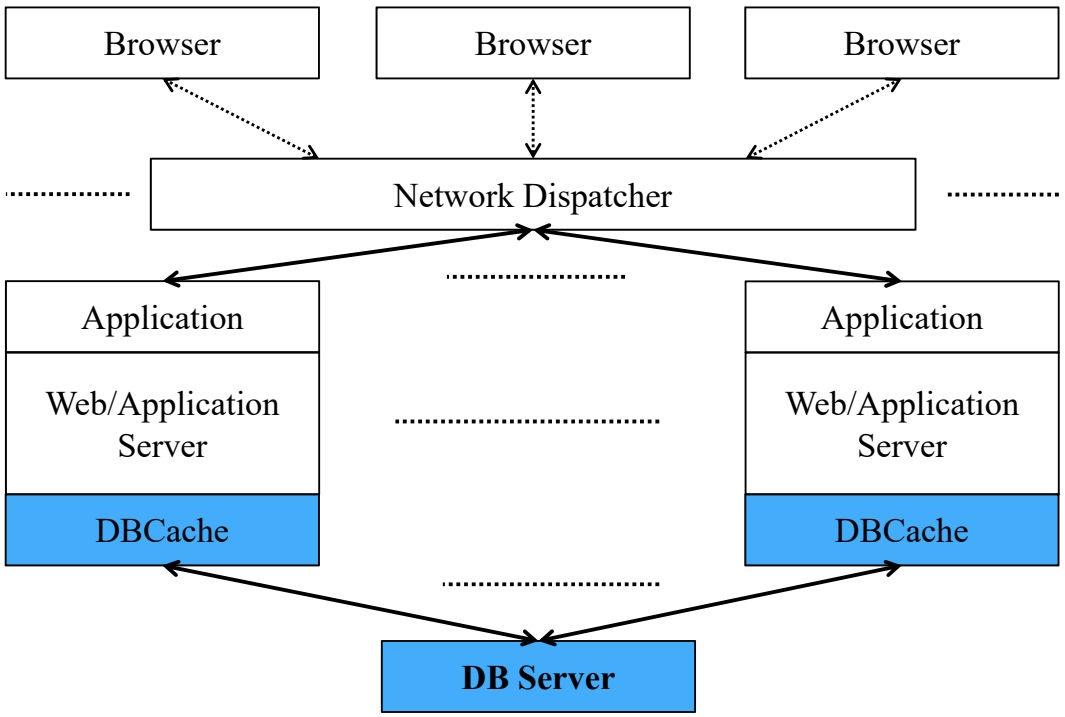
2. Klassifikation von Mehrrechner-DBS

- Merkmale für PDBS/VDBS
  - ***räumliche Verteilung***: ortsverteilt oder lokal
  - ***Rechnerkopplung***: enge, lose oder nahe Kopplung
  - ***Externspeicheranbindung***: partitioniert oder gemeinsam ('shared')
- PDBS-Architekturen
  - *Scale-Up* vs. *Scale-Out*
  - *Shared Nothing* vs. *Shared Storage*
- Weitere Klassifikationsmerkmale
  - funktionale Spezialisierung vs. funktionale Gleichstellung
  - integrierte vs. heterogene/föderierte MRDBS
- Systemansätze zur Datenintegration
- Grobbewertung von MRDBS-Alternativen

Verteilung der Funktionalität

- funktionale Gleichstellung („horizontale Verteilung“)
  - jeder Knoten besitzt gleiche DBS-Funktionalität (z.B. vollständiges DBMS pro Knoten)
  - Replikation der Funktionen
- funktionale Spezialisierung: Partitionierung von Funktionen
  - Beispiele:
    - *DB-Maschinen* mit Spezialprozessoren für bestimmte DB-Funktionen (Join-Prozessor, Sortier-Prozessor, etc.)
    - intelligente Platten-Kontroller z.B. für Selektion/Projektion (**Oracle Exadata**)
    - Optimierung von DB-Funktionen durch programmierbare/konfigurierbare GPUs und FPGA (Bsp. IBM Netezza)
    - Web-Informationssysteme (Multi-Tier-Architekturen) mit DB-Server und DB-Verarbeitung auf Applikations-Server
  - Spezialisierung erschwert Lastbalancierung, Erweiterbarkeit und Fehlertoleranz
- hybride Partitionierung + Replikation von DBS-Funktionen

Web-Informationssysteme mit Mid-Tier Caching

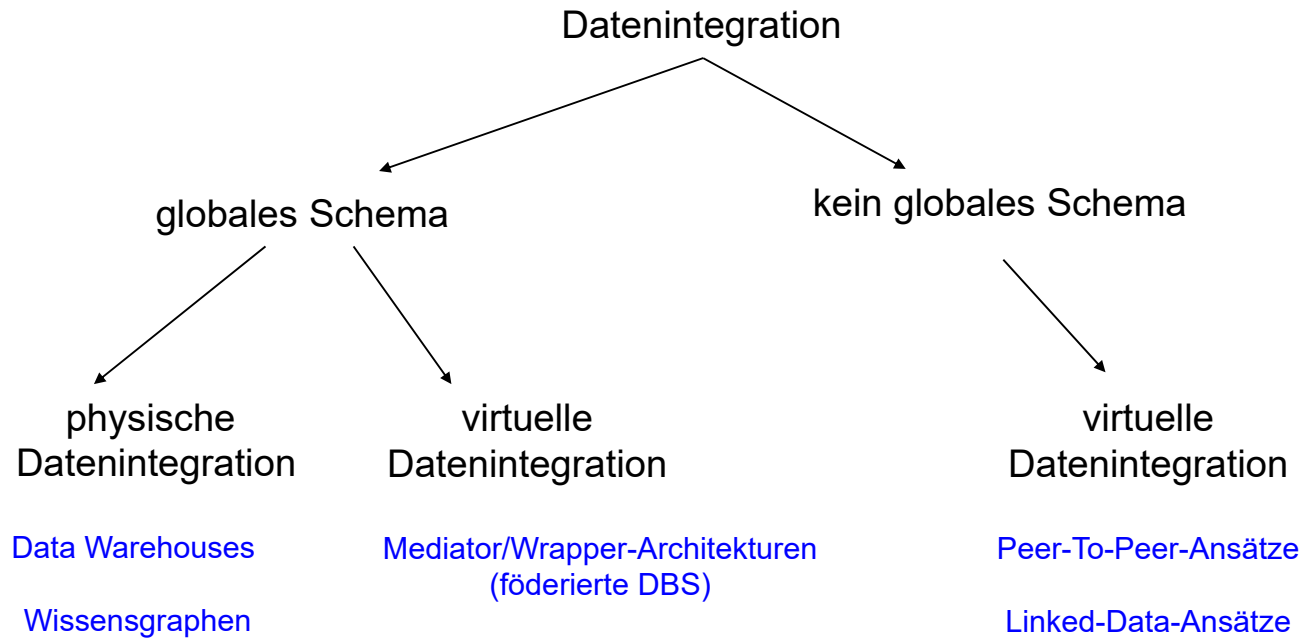


Integrierte vs. heterogene/föderierte MRDBS

- integrierte Mehrrechner-DBS
  - 1 logische Datenbank: DB-Zugriff wie im zentralen Fall (Verteiltransparenz für AP)
  - Top-Down-Ansatz zur Verteilung einer DB
  - homogenes MRDBS (z. B. identische DBMS-Instanzen)
  - geringe Autonomie für beteiligte DBMS
  - Beispiele: Verteilte DBS, Parallele DBS

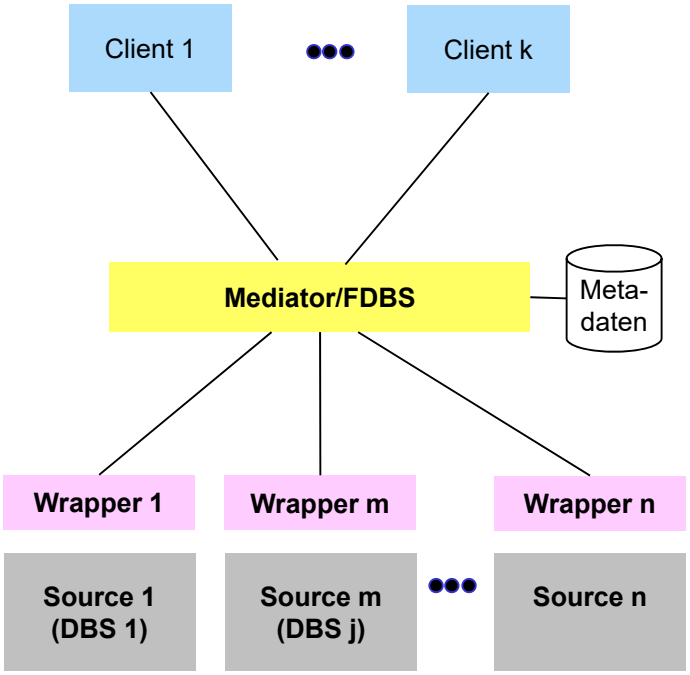
- heterogene / föderierte Mehrrechner-DBS
  - Bottom-Up-artige Kopplung existierender Datenbanken
  - weitgehend unabhängige DBMS mit eigenen DB-Schemata
  - partielle Zulassung externer Zugriffe (Kooperation)
  - Heterogenität bei Datenmodellen und Transaktionsverwaltung möglich
  - große Probleme mit semantischer Heterogenität
  - Verteilungstransparenz i.a. nur bedingt erreichbar
  - verschiedene Alternativen zur Datenintegration

Heterogene DBS

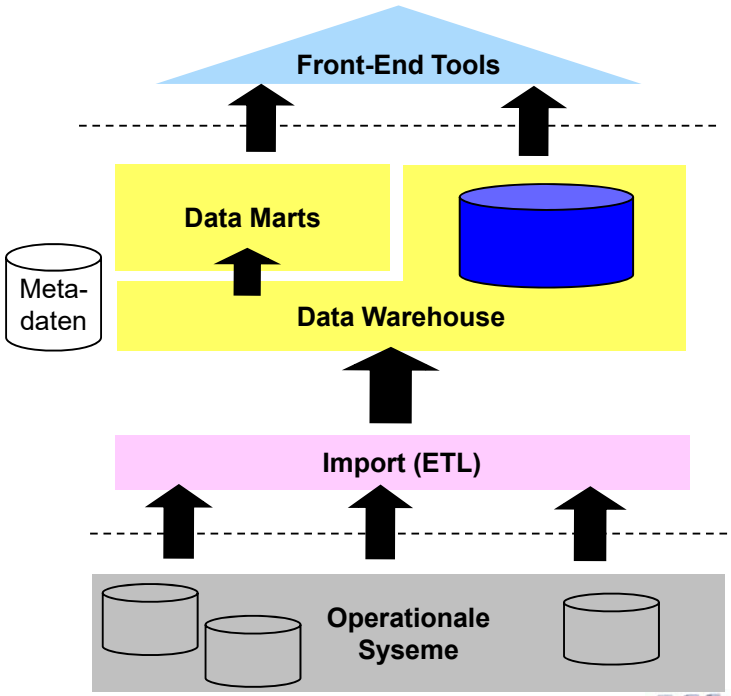


Grundlegende Alternativen zur Datenintegration

Virtuelle Integration
(Mediator/Wrapper-Architekturen, föderierte DBS)

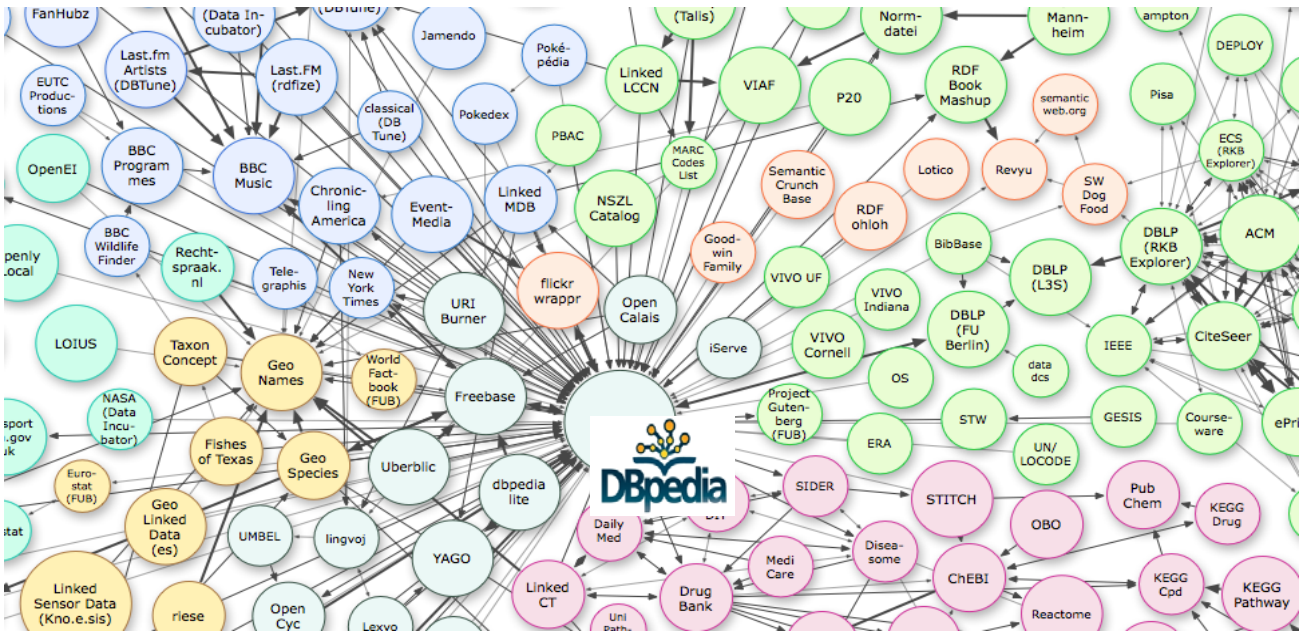


Physische (Vor-) Integration
(Data Warehousing)



Linked Open Data

- vernetzte RDF-basierte Datenquellen
- Links (Mappings) zwischen Instanzen und Ontologie-Konzepten ermöglichen Datenintegration



Wissensgraphen

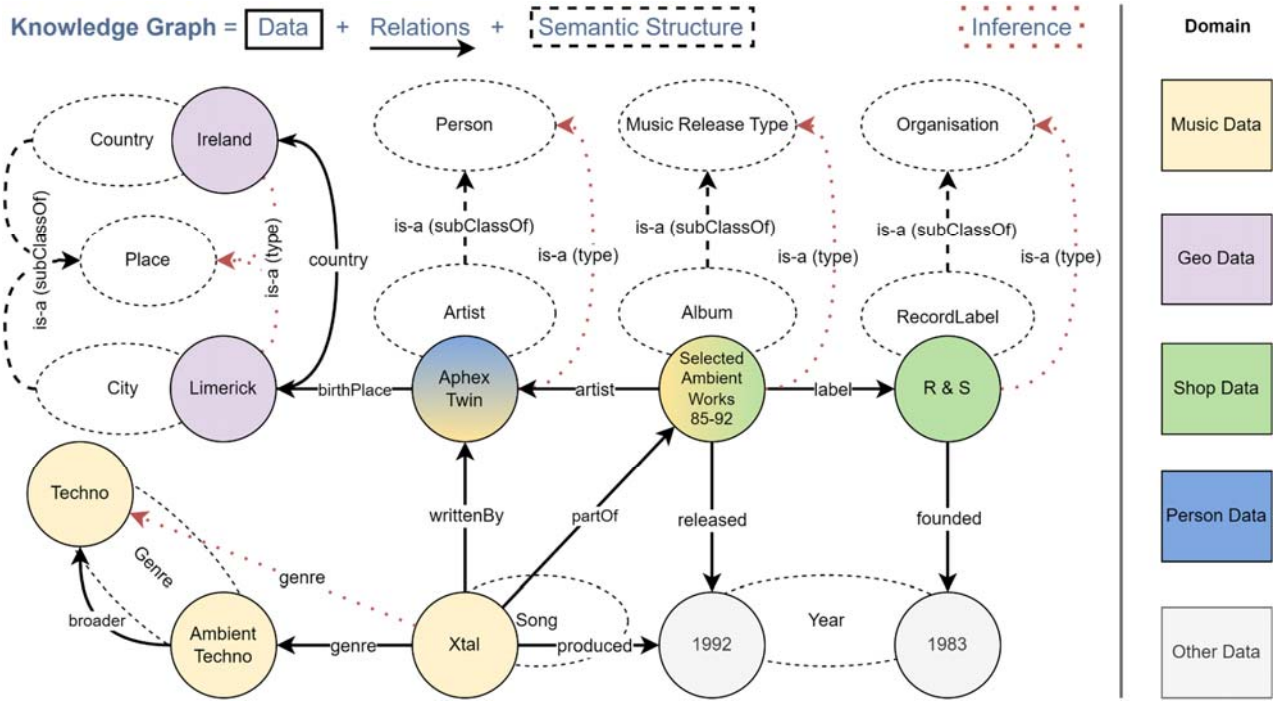
- einheitliche Verwaltung und semantische Kategorisierung von Entitäten und ihren Beziehungen
  - Beispiele: DBPedia, Yago, Wikidata
  - Google KG, MS Satori
  - Amazon Product KG …

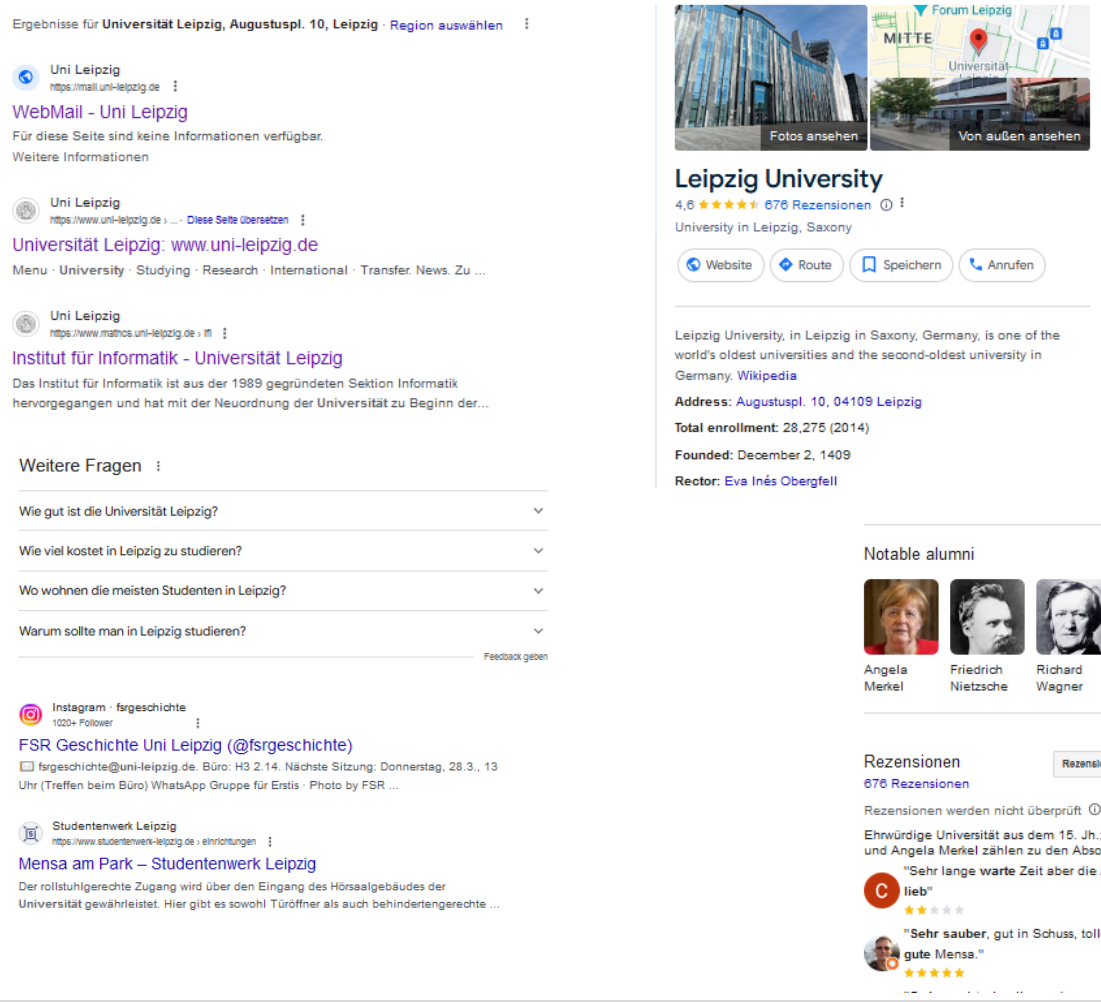
DBpedia

yago
select knowledge

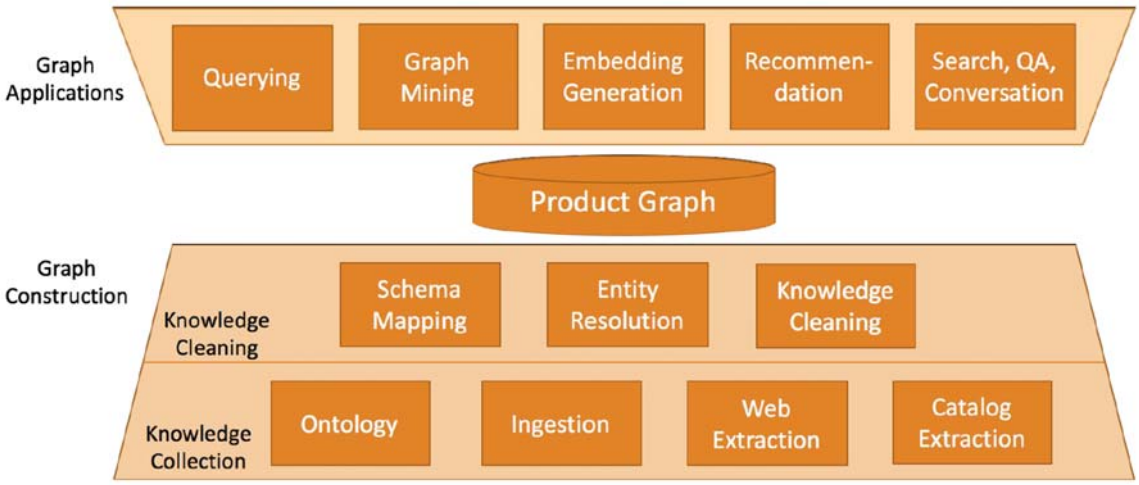
- Informationen stammen oft aus anderen Wissensquellen (Wikipedia, Wordnet etc.), Webseiten …
  - domänenspezifisch oder übergreifend / global
- viele Nutzungmöglichkeiten
  - Anreicherung von Daten
  - Verbesserung von Suchergebnissen
  - Generierung von (Produkt-) Empfehlungen
  - Gewinnung von Trainingsdaten für Machine Leaning …

Beispiel-Wissensgraph



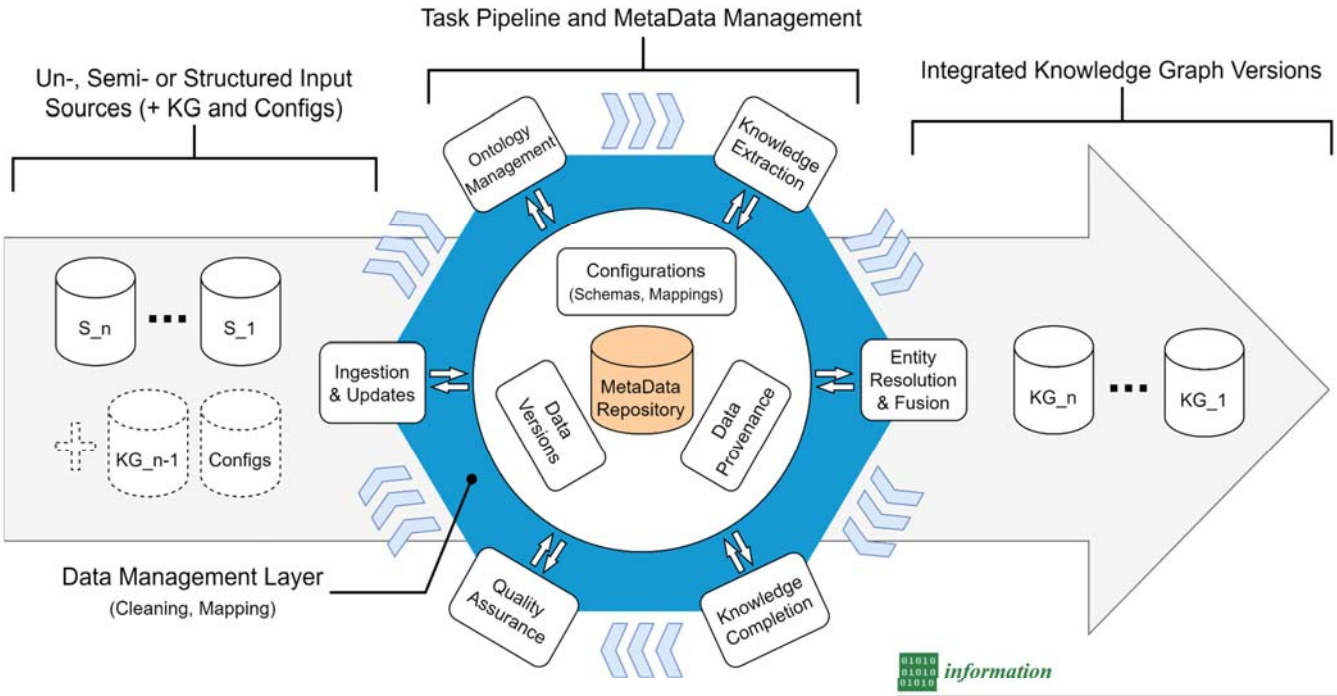


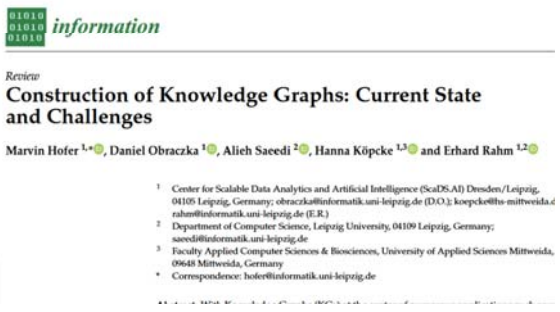
Beispiel: Product Knowledge Graph



from: Dong. KDD2018

Konstruktion von Wissensgraphen




information

Review

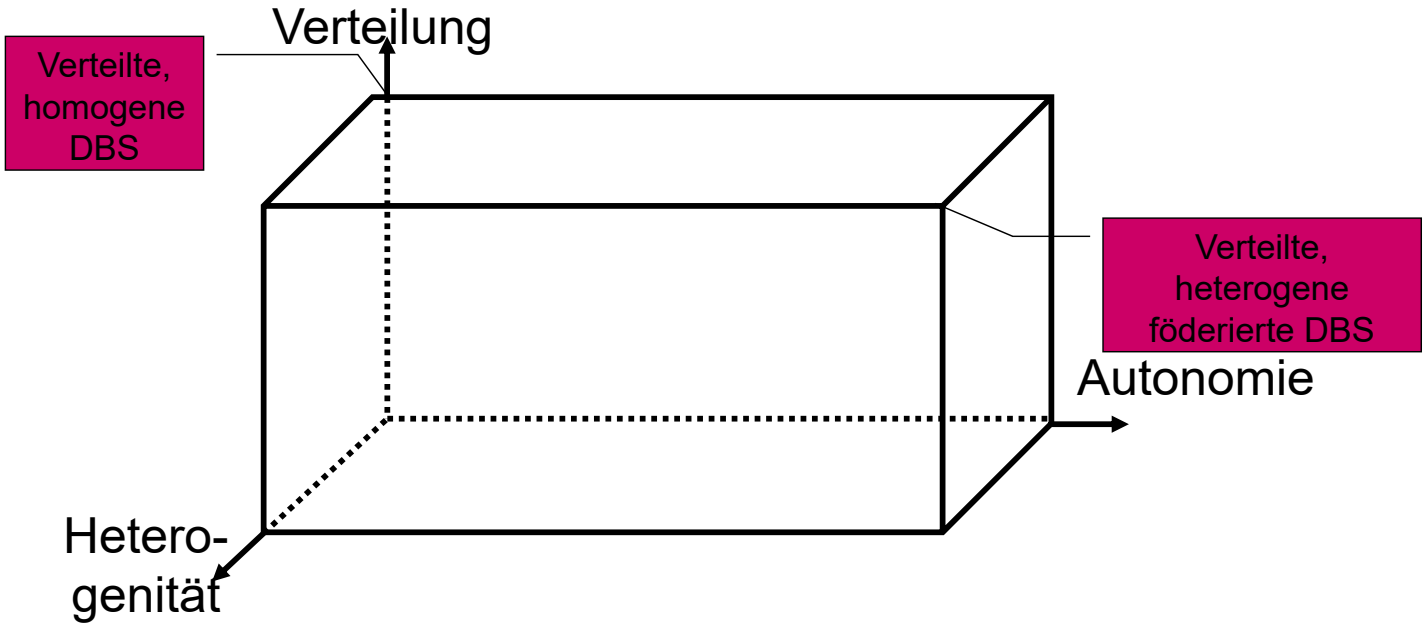
Construction of Knowledge Graphs: Current State and Challenges

Marvin Hofer [1,*], Daniel Obraczka [1], Alieh Saeedi [2], Hanna Köpcke [1,3] and Erhard Rahm [1,2]

[1] Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden/Leipzig, 04105 Leipzig, Germany; obraczka@informatik.uni-leipzig.de (D.O.); koepcke@hs-mittweida.de; rahm@informatik.uni-leipzig.de (E.R.)
[2] Department of Computer Science, Leipzig University, 04109 Leipzig, Germany; saeedi@informatik.uni-leipzig.de
[3] Faculty Applied Computer Sciences & Biosciences, University of Applied Sciences Mittweida, 09648 Mittweida, Germany
* Correspondence: hofer@informatik.uni-leipzig.de

https://doi.org/10.3390/info15080509

Klassifikation nach Özsu/Valduriez



Grobbewertung von Mehrrechner-DBS

	Parallele DBS (SD/SS, SN)	Verteilte DBS	Föderierte DBS
hohe Transaktionsraten	++	o/+	o
Intra-Query-Parallelität	++	o/+	-/o
Skalierbarkeit	+/++	o/+	o
Verfügbarkeit	+	+	-
Verteilungstransparenz	++	+	o
geographische Verteilung	-	+	+
Knotenautonomie	-	o	+
DBS-Heterogenität	-	-	+
Administration	o	-	-/--

- Big Data Architekturen nutzen v.a. Shared Nothing
  - NoSQL meist auf Shared-Nothing-Cluster
  - NewSQL z.B. mit In-Memory-Datenbanken / Column Stores
- Shared Storage zjnehmend verbreitet für Cloud DBS

Zusammenfassung

- vielfältige Anforderungen an Mehrrechner-DBS führen zu verschiedenen Architekturtypen:
  - Parallele DBS, Verteilte DBS, föderierte DBS, Data Warehouses …
- Klassifikationsmerkmale
  - räumliche Verteilung, Rechnerkopplung, Externspeicheranbindung, integrierte/homogene vs. föderierte / heterogene DBS, funktionale Spezialisierung vs. Gleichstellung
- Parallele DBS:
  - Ziele: hohe Leistung/Parallelisierung, hohe Verfügbarkeit, Skalierbarkeit
  - lokale Rechneranordnung
  - Hauptansätze: Shared Everything, Shared Storage, Shared Nothing
- Verteilte DBS: ortsverteilte, integrierte MRDBS (globales Schema)
- PDBS-Vergleich
  - Skalierbarkeit: Scale Up (SE) einfacher umsetzbar als Scale-Out
  - Scale Out: Shared-Disk/Storage vs. Shared-Nothing
  - Geo-Replikation für Hochverfügbarkeit