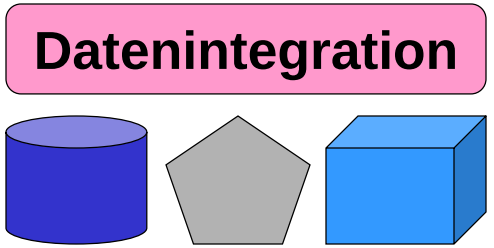
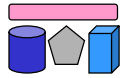
Datenintegration



Kapitel 3: Eigenschaften von Integrationssystemen

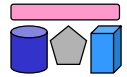
Dr. Anika Groß
Sommersemester 2016

Universität Leipzig
Institut für Informatik
http://dbs.uni-leipzig.de



Inhalt

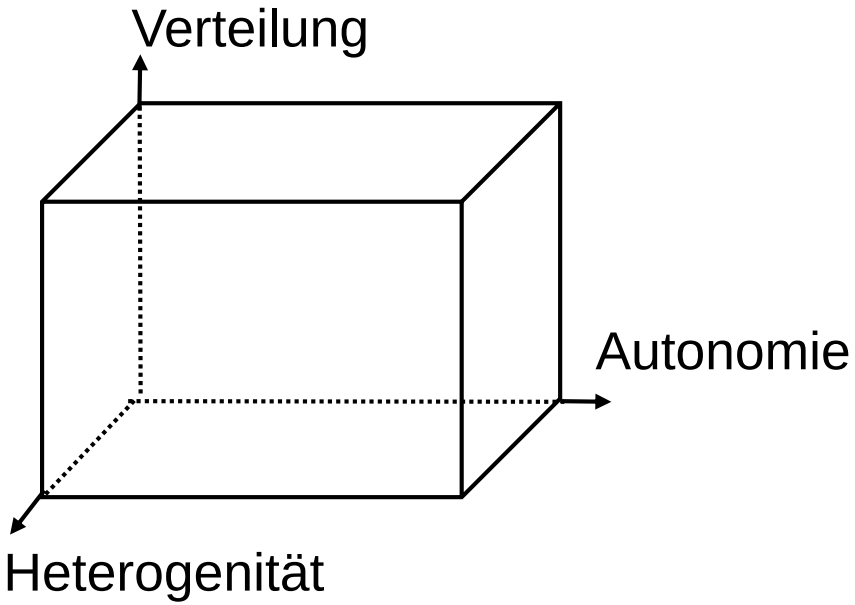
- Einordnung von Integrationssystemen bzgl.
  - Verteilung
  - Autonomie
  - Heterogenität
- Kriterien zur Beschreibung von Integrationssystemen
  - Enge und lose Kopplung
  - Art der semantischen Integration
  - Bottom-up- oder Top-down-Entwurf
  - …
- Vergleich materialisierte vs. virtuelle Integration
  - Flexibilität
  - Aktualität
  - Antwortzeit
  - …



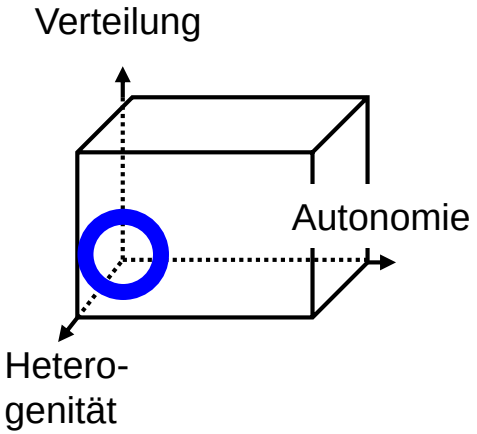
Verteilung, Autonomie und Heterogenität

- Drei orthogonale Kriterien
  → „Raum“ von Architekturen

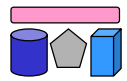
- Nicht alle Kombinationen in der Praxis relevant
  - Z.B. Autonom, aber weder verteilt noch heterogen



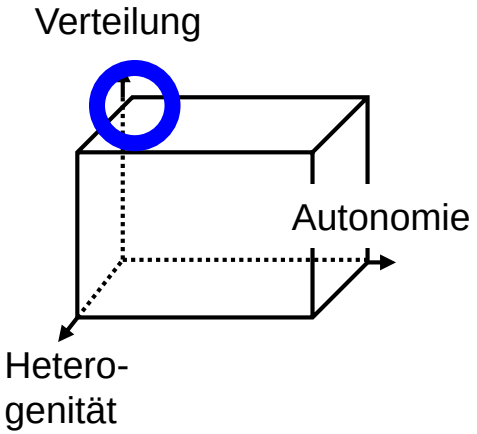
1. Zentrale Datenbank [---]



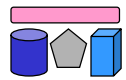
- „Normalfall“ – homogene, zentrale Datenbank
  - Daten/Berechnung können trotzdem begrenzt verteilt sein
    - Filesystem: Partitionierung, RAID, SAN
    - Berechnung: Cluster
    - Parallele Datenbanken
- Datenbank entsteht aus homogenem Entwurf
  - Wenn Redundanz / Heterogenität, dann mit Absicht und kontrolliert
  - Problem: Weiterentwicklung (Evolution)
- Zentrale Kontrolle und Administration



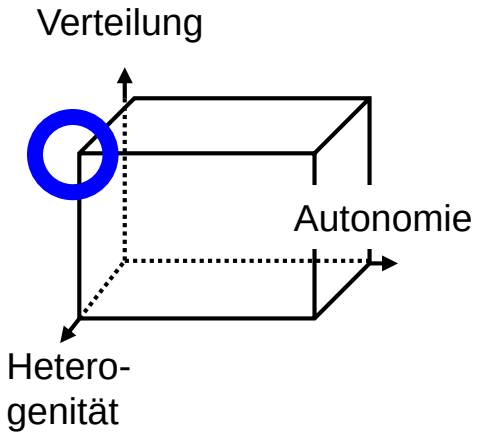
2. Verteilte Datenbanken [V--]



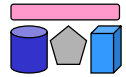
- Daten liegen physisch verteilt
  - Beabsichtigte, kontrollierte, a-priori Verteilung
  - Existenz eines konzeptionell homogenen, verteilt realisierten Schemas
  - Knoten sind nicht autonom
- Ziele
  - Höhere Performanz durch Parallelisierung
  - Höhere Sicherheit vor Katastrophen
  - Höhere Ausfallsicherheit durch redundant ausgelegte Systeme (Replikation)



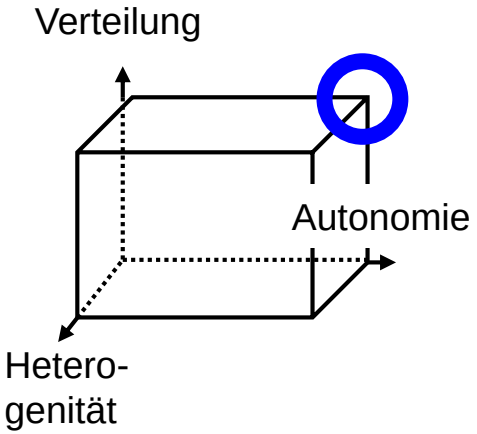
3. Verteilte & heterogene DB [V-H]



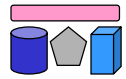
- Daten liegen physisch verteilt vor
- Daten sind heterogen
  - Z.B. aufgrund der Historie der Entstehung
- Keine Autonomie
  - Heterogenität sollte also nicht schlimmer werden
- Typischer Zwischenzustand



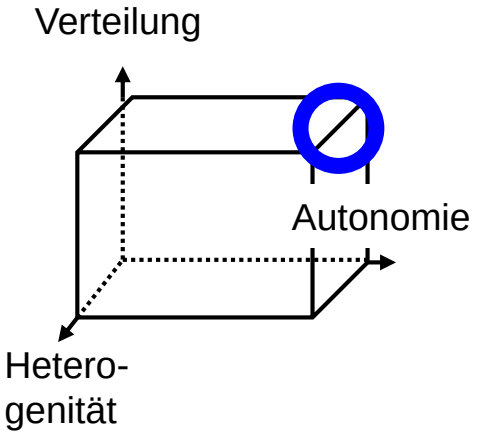
4. Verteilte & autonome DB [VA-]



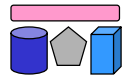
- Verteilte, aber homogene Datenbestände
- Entsteht durch freiwillige Übernahme von Regeln
  - Standards, Verträge, …
- Autonomie wird teilweise aufgegeben
  - Z.B. Aufgabe von Designautonomie
  - Z.B. nicht Kommunikationsautonomie
  - Z.B. nicht juristische Autonomie



5. Multidatenbanken [VAh]

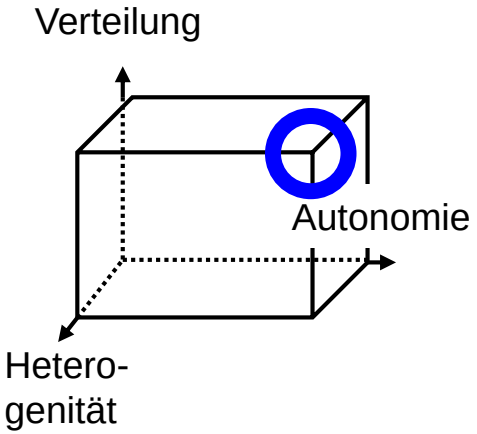


- Verteilt, autonom, und „etwas“ heterogen
  - Keine technische Heterogenität
  - Keine Datenmodellheterogenität
  - Schemata können strukturell und semantisch heterogen sein
  - Verteilte Systeme benutzen gleiche Techniken (RDBMS)
    - Zugriff über einheitliche Sprache (oder Simulation durch Wrapper)
- Autonomie bleibt bewahrt
  - Aber Zugriff muss möglich sein (Kommunikationsautonomie)
- Zugriff über Multidatenbanksprachen
  - Qualifizierung von Tabellennamen mit Datenbanknamen
  - Beispiel: SchemaSQL



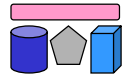
6. “Diese Vorlesung” [VHA]

- Quellen behalten volle Autonomie
  - Wissen u.U. nichts von ihrer Integration
  - Nehmen u.U. keine Rücksicht bzgl. Änderungen
- Quellen sind heterogen
- Quellen sind verteilt



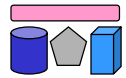
Weitere Kriterien

- Strukturierungsgrad der Daten
- Enge und lose Kopplung
- Datenmodell
- Art der semantischen Integration
- Transparenz
- Anfrage-Paradigma
- Bottom-up oder Top-down Entwurf
- Read-only oder read-&-write



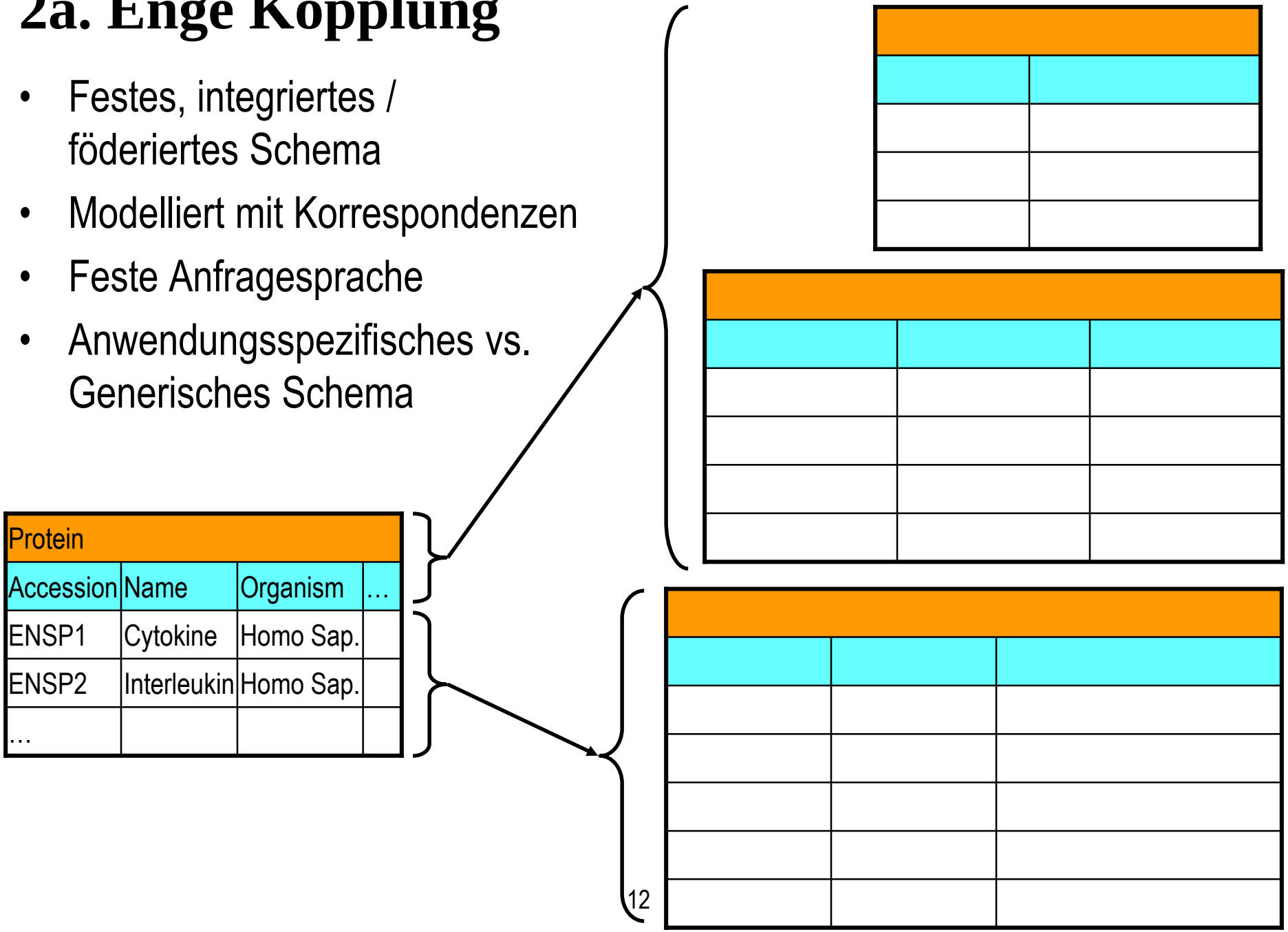
1. Strukturierungsgrad der Daten

- Strukturiert
  - Festes Schema, festes Format
  - Beispiel: Datenbanken
- Semi-strukturiert
  - Struktur vorhanden, aber nur teilweise bekannt
  - Daten sind mit Semantik gelabelt
  - Beispiel: XML Dokumente, Property-Dateien (Objekt-Attribut-Wert)
- Unstrukturiert
  - Keine Struktur
  - Beispiel: Textuelle Daten, Abstracts



2a. Enge Kopplung

- Festes, integriertes / föderiertes Schema
- Modelliert mit Korrespondenzen
- Feste Anfragesprache
- Anwendungsspezifisches vs. Generisches Schema



2b. Lose Kopplung

- Kein festes Schema
  - Nutzer müssen Semantik der Quellen kennen
  - Integrierte Sichten helfen
- Feste Anfragesprache
- Multidatabase query language
- SchemaSQL
  - Beispiel rechts: “Alle Abteilungen in univ-A, die Technikern mehr zahlen als in gleichen Abteilungen von univ-B”

univ-A

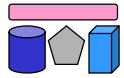
salInfo

category	dept	salFloor
Prof	CS	65,000
AssocProf	CS	50,000
Technician	CS	45,000
Prof	Math	60,000
AssocProf	Math	55,000
Technician	Math	45,000

univ-B

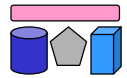
salInfo

category	CS	Math
Prof	55,000	65,000
AssocProf	50,000	55,000
Technician	43,000	44,000



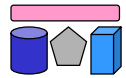
3. Datenmodell

- Kanonisches Datenmodell (im integrierten System)
  - Objektorientiertes Modell
  - Relationales Modell
  - Hierarchisches Modell
  - Semistrukturiertes Modell
  - XML Datenmodell
- Verlustfreie Integration ist schwierig
  - Beispiel: OO to Relational Mapping
    - Datenquelle: OO, kanonisches Datenmodell: Relational
    - Schlüssel erfinden
    - Nicht-Atomare Attribute müssen untergebracht werden: struct, set, bag, list, array
    - Semantische Beziehungen gehen verloren (Schlüssel/Fremdschlüssel)



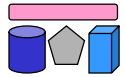
4. Art der semantischen Integration

- Vereinigung
  - Simple „Konkatenation“ von Objekten
  - Beispiel: “Merge” von mehreren RSS-Feeds
  - Problem: Redundanz
- Anreicherung
  - Anreicherung um Metadaten; keine Konfliktauflösung
  - Erzeugt mehr Daten $\rightarrow$ Qualität?
- Fusion
  - Objektidentifizierung
  - Re-Strukturierung
  - Komplementierung $\rightarrow$ Mehrere Objekte werden zu einem integriert
  - Aggregation $\rightarrow$ Konfliktlösung
  - Beispiel: Integration zweier Publikations-Web-Services (Kap. 1)



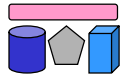
5. Transparenz

- Speicherorttransparenz
  - Physischer Ort der Daten unbekannt
  - IP, Quellenname, DB Name
- Schematransparenz
  - Nutzer sehen nur integriertes Schema
  - Strukturelle Konflikte werden verborgen
- Sprachtransparenz
  - Nutzer muss nur eine Sprache beherrschen
  - Integriertes System übersetzt



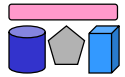
6. Anfrage-Paradigma

- Strukturierte Anfragen
  - Struktur ist Nutzern bekannt
  - Struktur kann in Anfrage verwendet werden
  - Beispiel: DBMS + SQL
- „Canned queries“
  - Vordefinierte Anfragen (parametrisiert)
  - Beispiel: Web-Formular
- Such-Anfragen
  - Struktur unbekannt
  - Information Retrieval
  - Beispiel: Suchmaschinen auf Texten
- Browsing
  - Kein Such-Interface
  - Beispiel: Web-Portal



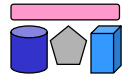
7. Bottom-up oder Top-down Entwurf

- Beim Entwurf des integrierten Systems
- Bottom-up:
  - Ausgelöst durch den Bedarf mehrere Quellen integriert anzufragen
  - Ein föderiertes (globales) Schema wird als “Vereinigung” der Quellenschemata gebildet (Schema Integration → Kap. 6)
  - Änderungen schwierig, da neu integriert werden muss
  - Typisches Szenario: Data Warehouse
- Top-down
  - Ausgelöst durch globalen Informationsbedarf
  - Einem definierten föderierten Schema müssen die Elemente der jeweiligen Quellenschemata zugeordnet werden (Schema Matching → Kap. 6)
  - Vorteilhaft bei labilen Quellen
  - Typisches Szenario: Virtuelle Integration



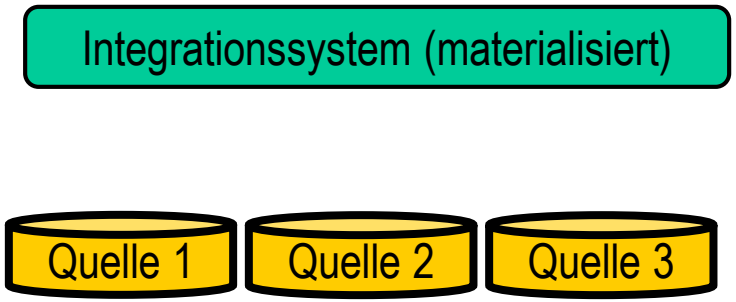
8. Read-only oder read-&-write

- Read-only die beliebtere Variante
- Write (insert & update) schwierig
  - Viele Interfaces erlauben kein Schreiben
  - Update durch Sichten ist schwierig
  - Bei Komplementierung: Welche Quelle?
  - Globale Transaktionen (komplexe Protokolle)
  - Autonomie!

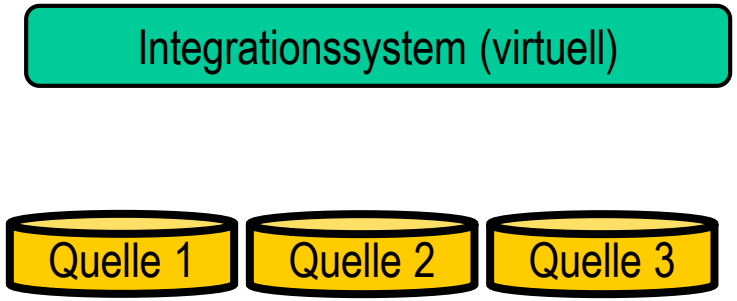


Integration der Instanzdaten

- Materialisiert (“Push”)
  - Daten werden transformiert und lokal gespeichert
  - Anfragen werden direkt gegen die materialisierten Daten gestellt
  - Beispiel: Data Warehouse



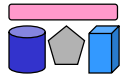
- Virtuell (“Pull”)
  - Anfragen werden in Teilanfragen übersetzt
  - Daten werden nur bei Bedarf übertragen und nur temporär gespeichert
  - Nachbearbeitung (Data Cleaning)
  - Beispiel: Query-Mediator



→ Großer Einfluss auf Vielzahl von Kriterien

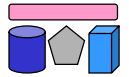
Materialisierte vs. Virtuelle Integration (1)

	Materialisiert	Virtuell
Aktualität der Daten	– je nach Update-Frequenz (z.B. tägl.)	+ Immer aktuelle Quellendaten (evtl. Cache)
Antwortzeit	+ Lokale Bearbeitung; DBMS-Optimierung (material. Sichten, Indizes, …)	– Datenübertragung, Antwortzeit der Quellen, schwierige Optimierung, Data Cleaning zur Laufzeit
Flexibiltät	– Entfernen/Ändern/Hinzufügen einer Quelle kann gesamte Integr. Verändern	+ Entfernen/Ändern/Hinzufügen einer Quelle wirkt sich nur auf Mapping dieser Quelle aus
Wartbarkeit	– Lokale Wartung (wachsender Bestand)	+ Quellen selbst für Wartung verantwortlich
Komplexität	Komplexe Anfragen (wie bei DBMS), Anfrageplanung leicht (GaV), Quellen oft ähnlich (z.B. DBMS)	Komplexe Modellierung der Quellen, Anfrageplanung schwierig (LaV), oft verschiedene Quellen (HTML, WS, …)
Autonomie	– Keine Kommunikations-A., geringe Ausführung- und Design-A., “Bulk-Read” muss möglich sein	+ Volle Design-Autonomie, (fast) volle Kommunikations- und Ausführungs-A.
Anfrage-bearbeitung	+ Wie bei DBMS	– Komplex wegen Autonomie, Verteilung und Heterogenität



Materialisierte vs. Virtuelle Integration (2)

	Materialisiert		Virtuell	
Anfrage-mächtigkeit	+	Volle SQL-Mächtigkeit	–	abhängig von Fähigkeiten der Quellen (z.T. global ausgleichbar); Spezialfähigkeiten (Text Index) nutzbar
Lesen / Schreiben	+/+	Lesen immer möglich; Schreiben unerwünscht (Inkonsistenzen!)	+/ –	Lesen bei Quellenverfügbarkeit mög-lich; Schreiben meist nicht (Autonomie)
Speicher-bedarf	–	Hoher Bedarf da Redundanz; meist stetiges Wachstum durch historische Daten	+	Gering; nur Metadaten und Zwischenergebnisse, evtl. Cache
Ressourcen-bedarf	?	Planbare Netzwerklast; abhängig von Workload	?	Potentiell hohe Netzwerklast; Mehrfachübertragung (falls kein Cache); abhängig von Workload
Vollständigkeit	+	Annahme: Materialisierung vollständig	?	Nur bei Verfügbarkeit aller Quellen, ggf. Anfrage unbeantwortbar / nur unvollständig beantwortbar
Data Cleaning	+	Offline; viele Methoden (auch manuelle Unterstützung)	–	Nur online; sehr schwierig
Datenqualität	+	Hoch; kontrolliert / kontrollierbar	–	Quellenabhänig; meist zweifelhaft



Zusammenfassung

- Einordnung von Integrationssystemen bzgl. Verteilung, Autonomie, Heterogenität
  - “Diese Vorlesung = verteilt, autonom & heterogen”
- Kriterien zur Beschreibung von Integrationssystemen
  - Vielzahl von Kombinationen möglich
    → Vielzahl von Integrationssystemen möglich
    → Typische Vertreter in Kap. 4
- Vergleich materialisierte vs. virtuelle Integration
  - Materialisiert: Data Warehouse → Kap 4.
  - Virtuell: Query-Mediator-Wrapper-Ansatz → Kap 4.