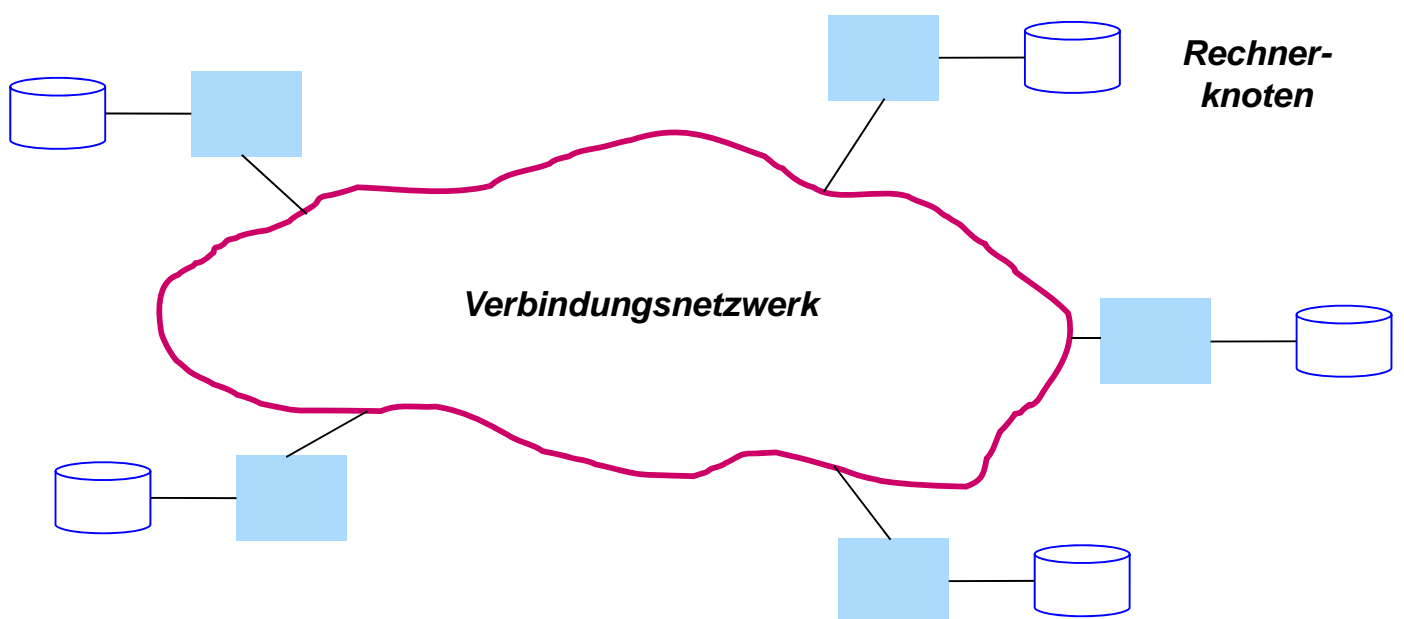


# 3. Verteilte Datenbanksysteme: Schemaarchitektur und Katalogverwaltung

- Einführung Verteilte DBS
- Schemaarchitektur
- Katalogverwaltung
- Namensverwaltung



## Grobaufbau eines Verteilten DBS



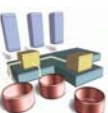
# Wünschenswerte Eigenschaften von VDBS

- traditionelle Zielsetzung (z.B. nach Chris Date):  
für den Benutzer sollen alle Aspekte der Verteilung verborgen bleiben (**Verteilungstransparenz**)
- Ortsunabhängigkeit (Ortstransparenz)
  - physische Lokation von Daten muss verborgen bleiben
  - Datenumverteilungen sollen keine Auswirkungen auf Programme haben
- Fragmentierungsunabhängigkeit
- Replikationsunabhängigkeit
- Hardware-, Betriebssystem-, Netzwerkunabhängigkeit
- *DBS-Unabhängigkeit* ( $\Rightarrow$  *heterogene DBS*)

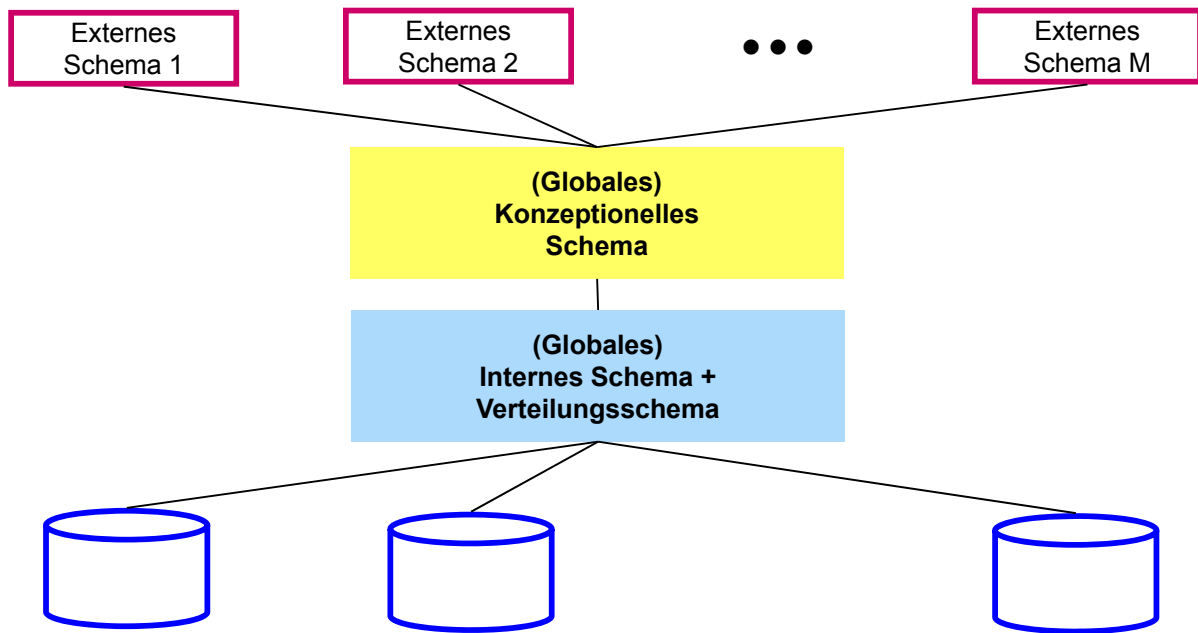


## VDBS Eigenschaften (2)

- größtmögliche lokale Autonomie
  - lokale Verwaltung von lokalen Daten
  - keine Abhängigkeit zu zentralen Knoten
- permanenter Betrieb
- verteilte Query-Bearbeitung
  - erforderlich für Zugriff auf externe Daten
  - Optimierung verteilter Anfragen
- verteilte Transaktionsverwaltung
  - Synchronisation
  - Recovery (verteilttes Commit-Protokoll)



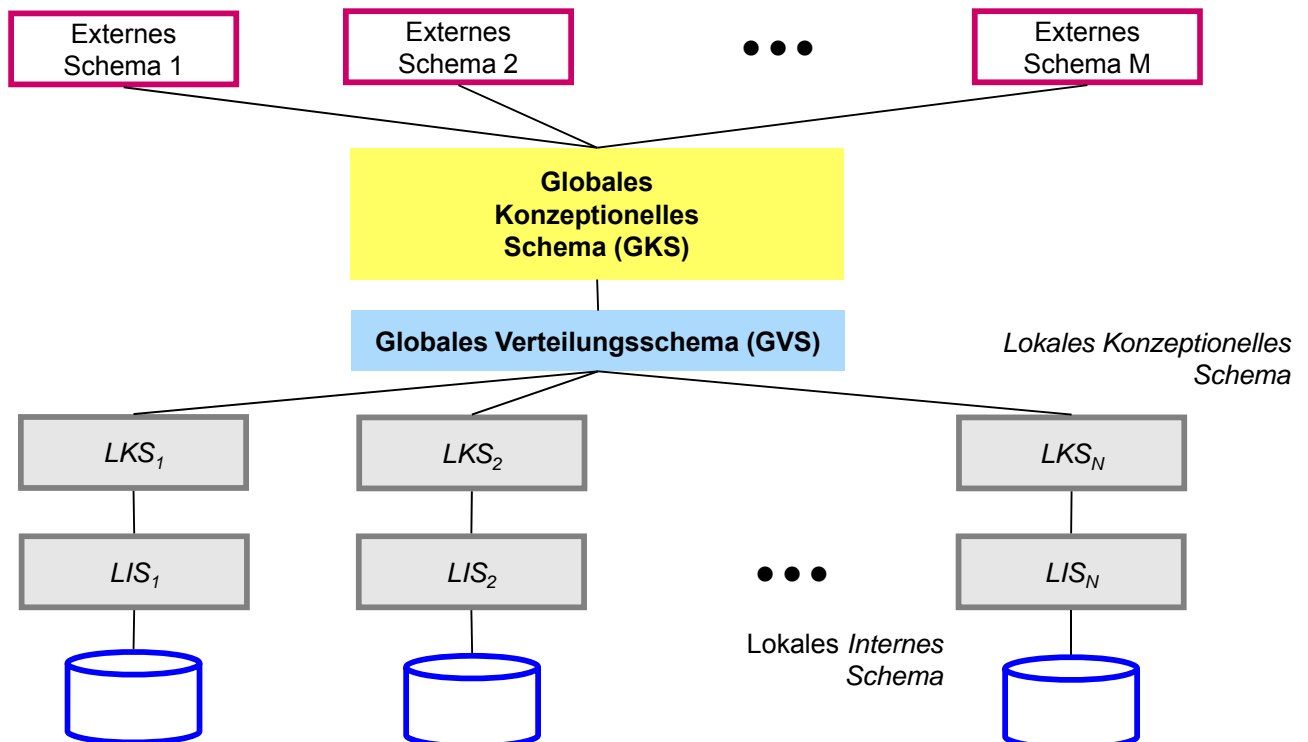
# VDBS Schemaarchitektur ?



- gemeinsames konzeptionelles und internes Schema
  - unterstützt Verteilungstransparenz aber keine Knotenautonomie
  - für geographisch verteilte Systeme ungeeignet



# VDBS Schemaarchitektur (2)

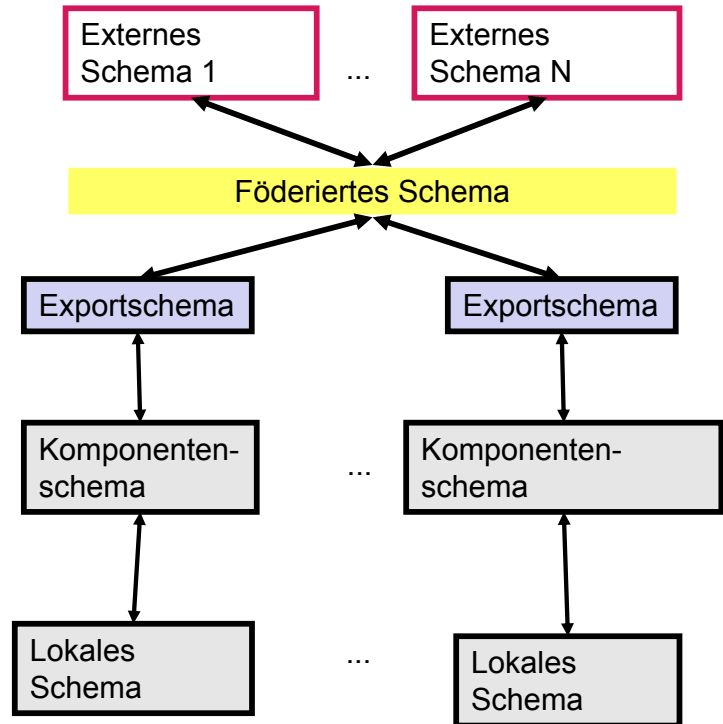


- Unterstützung von Verteilungstransparenz durch GKS
- Unterstützung von Knotenautonomie durch LKS und LIS



# Schemaarchitektur für Föderierte DBS (Sheth/Larson 1990)

- neu: Exportschemas
- Terminologie
  - Lokales Schema = lokales konzept. Schema
  - Föderiertes Schema = globales konzept. Schema
- Komponentenschema:
  - Kanonisches Datenmodell
- Exportschema
  - Teilmenge des Komponentenschemas



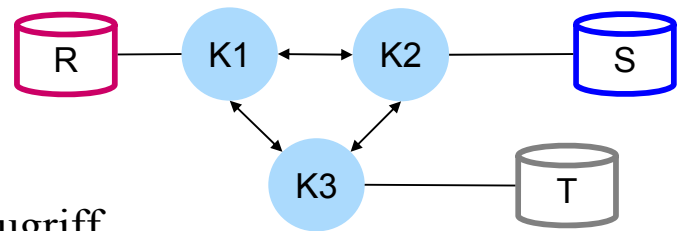
## Katalogverwaltung

- Katalog führt Metadaten für DB-Verarbeitung
  - Namen u. Adressen externer Knoten (= DBS-Instanzen)
  - Angaben zur Datenverteilung
  - Angaben zu Relationen, Sichten, Attribute, Integritätsbedingungen, Benutzern, Zugriffsrechten, Indexstrukturen, Statistiken, ...
- jeder Knoten sollte für lokale Objekte Katalogdaten lokal führen
- Realisierung für globalen Katalog ?
  - Verteilungsinformationen
  - Angaben zu nicht-lokalen Objekten und Benutzern



# Globaler Katalog: Realisierungsalternativen

- zentralisierter Katalog
  - Nachteile: Kommunikationsaufwand, Autonomie
- vollständig replizierter Katalog
  - schneller Lesezugriff
  - Änderungs- und Autonomieprobleme
- Mehrfachkataloge
  - Kombination aus den beiden ersten Ansätzen
- partitionierter Katalog
  - Identifikation des Katalogknotens über Objektnamen
  - hohe Knotenautonomie
  - für nicht-lokale Objekte  
Kommunikation bereits für Katalogzugriff



## Globaler Katalog (2)

- Variante: Partitionierte Kataloge + Caching von entfernten Katalogdaten
- Problem: Behandlung veralteter Katalogangaben
- Lösung 1 (SDD-1-Prototyp):
  - Besitzerknoten vermerkt sich, wo Katalogdaten gepuffert sind
  - Katalogänderung führt zur Invalidierung aller Kopien
- Lösung 2 (IBM R\*):
  - Verwendung von Zeitstempeln
  - bei Übersetzung/Optimierung von DB-Operationen wird Zeitstempel der verwendeten Katalogdaten vermerkt
  - bei Ausführung einer Operation wird festgestellt, ob veraltete Katalogdaten verwendet wurden
  - ggf. Neuübersetzung und -ausführung mit aktualisierten Daten



# Namensvergabe

## ■ Anforderungen

- eindeutige Bezeichner für globale Objekte: Relationen, Sichten, usw.
- lokale Namensvergabe
- Unterstützung von Verteilungstransparenz
- Stabilität gegenüber Datenumverteilungen (Migration)

## ■ Hierarchische Struktur des Namensraums, z.B.

[ [ <node-id>@ ] <user-id> . ] <object-id>

- node-id kann Internet-Adresse /URI sein, z.B. dbs1.uni-leipzig.de
- gewährleistet Knotenautonomie
  - lokale Namenswahl durch Benutzer wie in zentralisierten Systemen
  - verschiedene Benutzer können die gleichen Objektnamen verwenden
  - Referenzierung lokaler Objekte wie im zentralen Fall
- toleriert Netzwerk-Partitionierung
- passt sich dem Wachstum an
- Problem: Verwendung von Speicherknoten als <node-id> für externe Objekte verletzt Ortstransparenz  
=> Änderung der Datenallokation erfordert Programmänderungen!



## Namensvergabe (2)

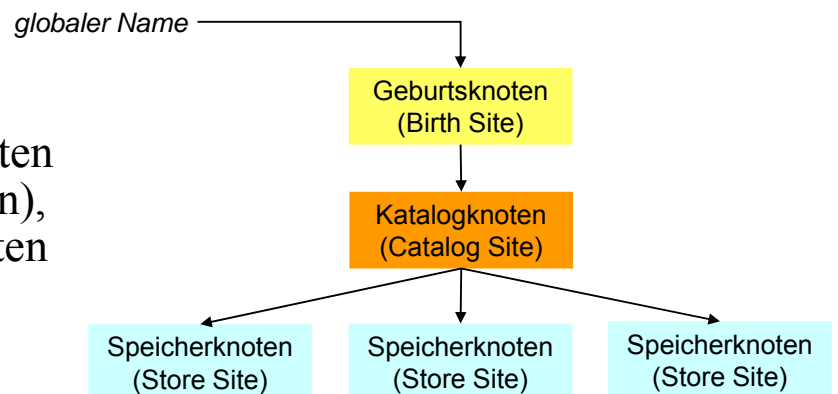
### ■ Verwendung des Geburtsknotens des Objekts

- realisiert im R\*-Prototyp
- Objektmigrationen bleiben ohne Auswirkungen

*Namensauflösung:*  
globaler Name -> physische Adresse

### ■ Unterscheidung von Geburtsknoten (sei im globalen Namen enthalten), Katalogknoten und Speicherknoten

- Unterstützung von Replikation (mehrere Speicherknoten)
- Trennung von Geburts- und Speicherknoten erlaubt Stabilität gegenüber Datenumverteilungen
- Katalogknoten kann mit Geburts- oder Speicherknoten übereinstimmen (=> Kommunikationseinsparungen)



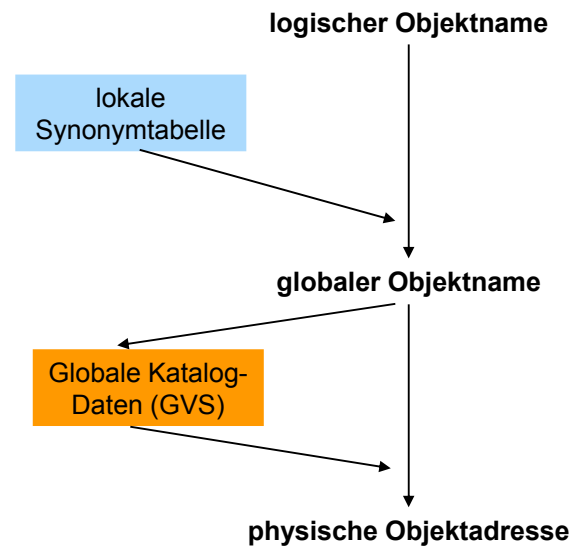
# Namensauflösung über Synonyme

## ■ Verwendung von Knoten-Namen weiterhin problematisch

- Default-Regelung zur Expansion des Geburtsknotens
- Nutzung von Synonymen (Aliases) reduziert Probleme

## ■ Synonyme (Alias-Namen): automatische Abbildung benutzerspezifischer logischer Namen in vollqualifizierte globale Namen

- Verwaltung von Synonymtabellen durch DBS im lokalen Katalog
- Unterstützung in DB2, Oracle, etc.
- Änderung der Knotennamen (Datenmigration, Knotenwegfall): nur Anpassung der Synonymtabellen



# Namensvergabe in Oracle

## ■ globaler DB-Name besteht aus zwei Komponenten

- lokaler DB-Name, z.B. HQ
- Domänenname (Internet-Konvention)

Bsp.: HQ.Division1.Example-Tools.COM

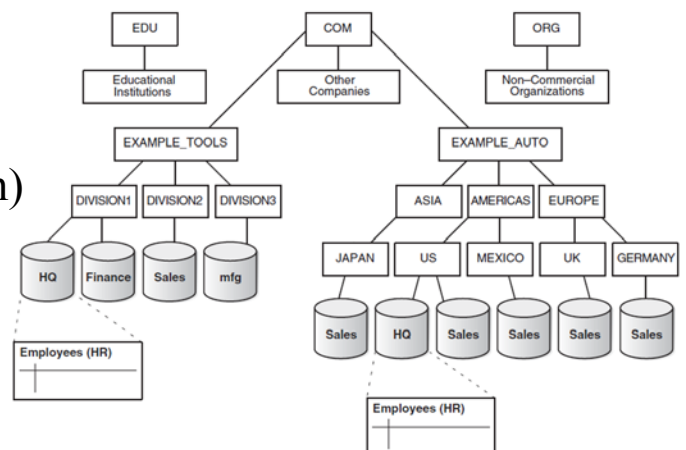
## ■ Festlegung durch Initialisierungsparameter

- DB\_NAME
- DB\_DOMAIN

## ■ Vergabe und Nutzung von Alias-Namen

```
CREATE PUBLIC SYNONYM PERS FOR  
Employees@HQ.Division1.Example-Tools.COM
```

```
SELECT * FROM PERS
```



## Zusammenfassung

- Zielkonflikte für VDBS: vollständige Transparenz vs. Knotenautonomie / Heterogenität
- Verteilungstransparenz:  
Orts-, Fragmentierungs-, Replikationstransparenz
- Schemaarchitektur
  - gemeinsames globales konzeptionelles Schema
  - separate lokale konzeptionelle und interne Schemata
- günstige Katalogarchitektur für VDBS:  
partitionierte Kataloge + Pufferung
  - in PDBS replizierter Katalog zweckmäßig
- globale Objektnamen
  - lokale Vergabemöglichkeit über hierarchische Namen
  - Ortstransparenz über Synonyme

