Datenbanken

10 Lernziele

- Wie funktionieren verteilte Datenbanken?
- Welche Vorteile haben verteilte Datenbanken?
- Welche Probleme ergeben sich beim Betrieb verteilter Datenbanken und wie lassen sich diese lösen?

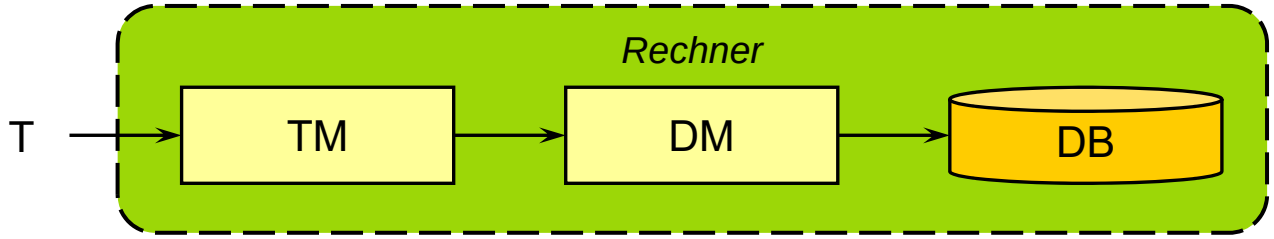
10 Literaturempfehlung

Lackes, R., Siepermann, M.: Verteilte Datenbanken. In: wisu – das wirtschaftsstudium, Zeitschrift für Ausbildung, Examen, Berufseinstieg und Fortbildung, 37. Jahrgang, Heft 4, Düsseldorf 2008, S. 572-578.

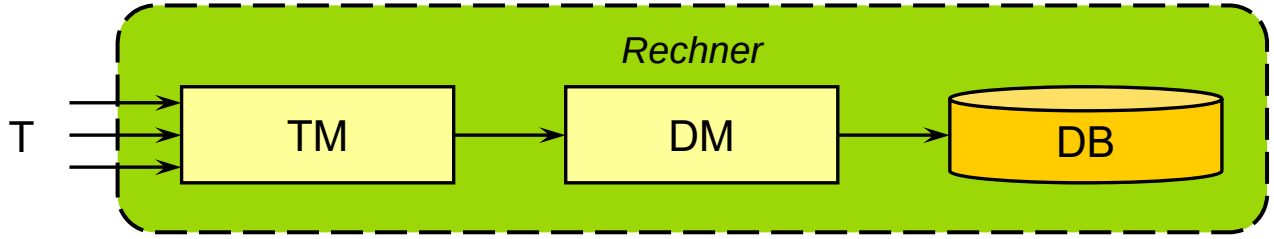
Lackes, R., Siepermann, M.: Konzeption und Funktionsweise verteilter Datenbanken. In: wisu – das wirtschaftsstudium, Zeitschrift für Ausbildung, Examen, Berufseinstieg und Fortbildung, 37. Jahrgang, Heft 8-9, Düsseldorf 2008, S. 1174 - 1180

10 Single-Site Processing

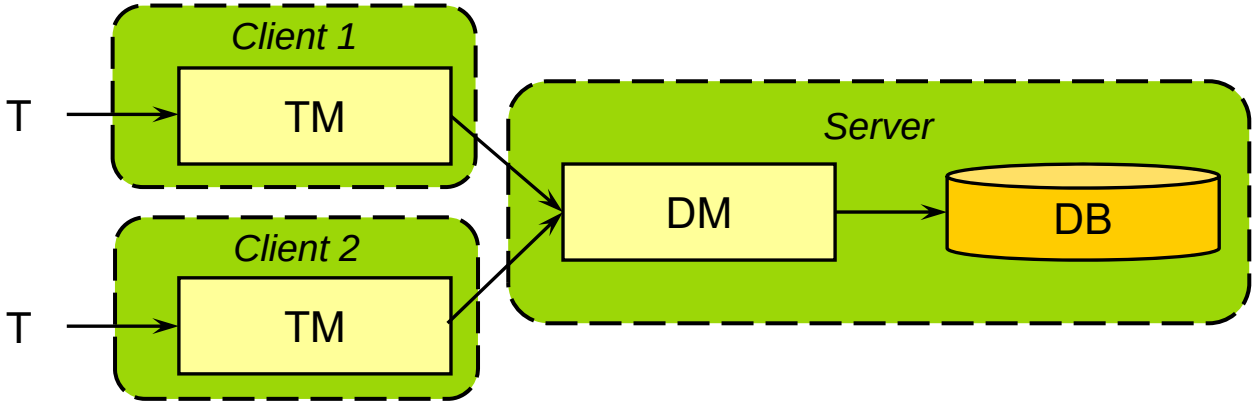
Single-User Architektur (Ein-Rechner-Architektur)



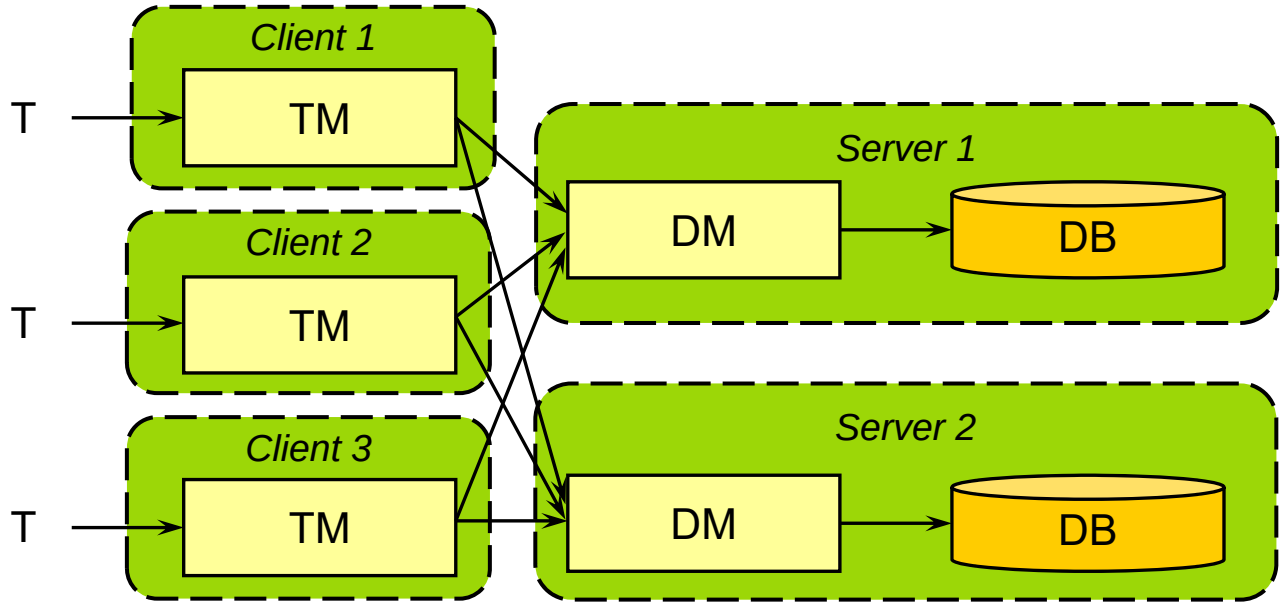
Multi-User Architektur (Mainframe mit Terminals)



Single-Site Data (Client-Server-Architektur)



Multiple-Site Data (Verteilte Architektur)



10 Verteiltes Datenbanksystem

Ein **verteiltes Datenbanksystem** besteht aus mehreren über ein Datennetz verbundenen Knoten, für die gilt:

- Jeder Knoten stellt eine eigenständige Datenbank dar, aber
- die Knoten arbeiten bei Bedarf zusammen, so dass jeder Benutzer an jedem Knoten auf jedes Datum, das im Netzwerk existiert, so Zugriff hat, als ob das Datum am Ort des Benutzers gespeichert wäre.

10 Verteilte Datenbasis

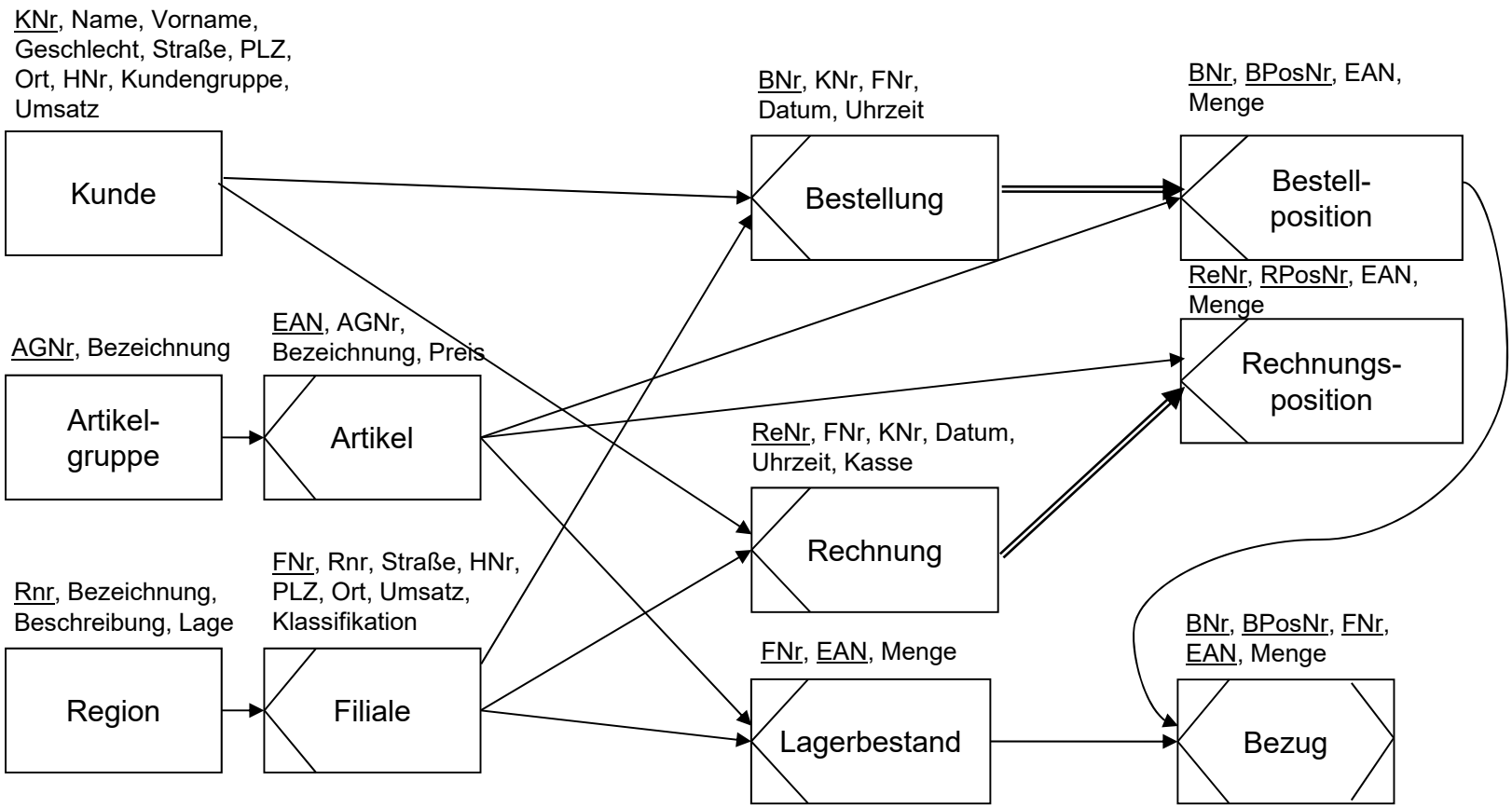
Eine **verteilte Datenbasis** ist eine Sammlung von Daten,

- die physisch über mehrere, autonom agierende, über ein Netzwerk verbundene Rechner verteilt sind;
- die durch die überwiegende Zahl von Anwendungen rechnerlokal bearbeitet werden;
- deren Zusammenschluss eine einheitliche, logische Gesamtsicht auf die Daten darstellt;
- die so verwaltet werden, dass jedes beliebige Anwendungsprogramm den Eindruck hat, die Daten würden auf dem Rechner gehalten, auf dem das jeweilige Anwendungsprogramm abläuft.
- auf die in ihrer Gesamtheit von jedem Rechner des Netzwerks aus zugegriffen werden kann.

10 Beispiel

Ein Unternehmen verfügt über mehrere Filialen in verschiedenen Regionen, deren aktueller Artikelbestand registriert wird. Verwenden Kunden beim Einkauf eine Kundenkarte, wird erfasst, wann sie in welcher Filiale welchen Artikel in welcher Menge bezogen haben. Darüber hinaus können auch Artikel bestellt werden, die nicht mehr in der Filiale vorhanden, jedoch noch in anderen Filialen in ausreichender Menge vorrätig sind.

10 Beispiel


KNr, Name, Vorname, Geschlecht, Straße, PLZ, Ort, HNr, Kundengruppe, Umsatz
Kunde
BNr, KNr, FNr, Datum, Uhrzeit
Bestellung
BNr, BPosNr, EAN, Menge
Bestell-position
ReNr, RPosNr, EAN, Menge
Rechnungs-position
AGNr, Bezeichnung
Artikel-gruppe
EAN, AGNr, Bezeichnung, Preis
Artikel
ReNr, FNr, KNr, Datum, Uhrzeit, Kasse
Rechnung
Rnr, Bezeichnung, Beschreibung, Lage
Region
FNr, Rnr, Straße, HNr, PLZ, Ort, Umsatz, Klassifikation
Filiale
FNr, EAN, Menge
Lagerbestand
BNr, BPosNr, FNr, EAN, Menge
Bezug

10 Beispiel

Kunde									
KNr	Name	Vorname	Geschlecht	Strasse	PLZ	Ort	HNr	Kundengruppe	Umsatz

Region			
RNr	Bezeichnung	Beschreibung	Lage

Artikelgruppe	
AGNr	Bezeichnung

Filiale							
FNr	RNr	Strasse	HNr	PLZ	Ort	Umsatz	Klassifikation

Artikel			
EAN	AGNr	Bezeichnung	Preis

Bestand		
FNr	EAN	Menge

Rechnung					
ReNr	FNr	KNr	Datum	Uhrzeit	Kasse

10 Beispiel

Rechnungsposition			
ReNr	RPosNr	EAN	Menge

Bestellung				
BNr	KNr	FNr	Datum	Uhrzeit

Bezug				
BNr	BPosNr	FNr	EAN	Menge

Bestellposition			
BNr	BPosNr	EAN	Menge

10 Vorteile verteilter Datenbanken

- Datenlokalität
  - Die Daten liegen an dem Ort, an dem sie am meisten benötigt werden
- Zugriffsgeschwindigkeit
  - Schnellerer Zugriff auf die Daten, da meist nur eine Teilmeinge der Gesamtdaten benötigt werden
- Bearbeitungseffizienz
  - Effizientere Datenbearbeitung, da dezentral

10 Vorteile verteilter Datenbanken

- Ausbaukosten
  - Hinzufügen neuer Workstations ist günstiger als Ausbau des Mainframe-Rechners
- Ausfallsicherheit
  - Durch Verteilung der Daten höhere Ausfallsicherheit
- Erweiterbarkeit
  - Neue Standorte können in das Netzwerk integriert werden, ohne dass andere Standorte davon beeinträchtigt werden

10 Nachteile verteilter Datenbanken

- Management und Kontrolle
  - Weitaus komplexer als bei einer zentralen Datenbasis
- Sicherheit
  - Mehr „Angriffsfläche“ führt zu erhöhten Sicherheitsmaßnahmen
- Unterstützung heterogener Systeme
  - Mangelnde Einhaltung von Standards und/oder unterschiedliche Versionen erschweren den Zusammenschluss vorhandener DBMS zu einem verteilten DBMS

10 Anforderungen an eine verteilte Datenbank

- Generell: Logische Transparenz aus Benutzersicht
  - „Fundamentalprinzip“
- Spezielle Regeln für verteilte Datenbanken:
  - Lokale Autonomie
  - Unabhängigkeit lokaler Komponenten
  - Dauerbetrieb / hohe Verfügbarkeit
  - Lokale Transparenz
  - Fragmentierungstransparenz
  - Replizierungstransparenz
  - Verteilte Bearbeitung von Datenbankoperationen
  - Verteiltes Transaktionsmanagement
  - Hardware-, Betriebssystem-, Netzwerk- und DBMS-Transparenz

10 Anforderungen an eine verteilte Datenbank

- Lokale Autonomie:
  - Jeder Standort kontrolliert und verwaltet autonom die am Standort gespeicherten Daten
  - Kein Standort A ist in dem Maße von einem anderen Standort B abhängig, dass A nicht funktionieren würde, wenn ein Zugriff auf B nicht möglich ist
- Unabhängigkeit lokaler Komponenten
  - Alle Standorte sind gleichberechtigt
  - keine zentrale Instanz

10 Anforderungen an eine verteilte Datenbank

- Dauerbetrieb / hohe Verfügbarkeit
  - Für Updateprozesse oder Hinzufügen neuer Standorte sollte der Betrieb an den vorhandenen Standorten nicht eingeschränkt werden
  - Unanfälligkeit gegenüber Fehlern wie z.B. Ausfall eines Knotens
- Lokale Transparenz
  - Ein Benutzer muss nicht wissen, an welchem Standort die Daten gespeichert sind
  - Die Daten müssen so erscheinen, als seien sie alle am lokalen Ort gespeichert

10 Anforderungen an eine verteilte Datenbank

- Fragmentierungstransparenz
  - Daten sollen an dem Ort gespeichert werden, wo sie am häufigsten benötigt werden
  - Dies erfordert eine Fragmentierung (Aufteilung) bestehender Relationen und Allokation auf die jeweiligen Standorte
  - Für den Benutzer einer verteilten DB soll die Fragmentierung transparent sein
- Replizierungstransparenz
  - Wenn Daten häufig an mehreren Stellen benötigt werden, ist eine Replizierung (kontrollierte Redundanz) sinnvoll
  - Vorteil: Arbeiten auf lokale Kopie; Verringerung der Netzlast
  - Nachteil: Aktualisierung/Pflege von Daten an mehreren Orten notwendig

10 Anforderungen an eine verteilte Datenbank

- Verteilte Bearbeitung von Datenbankoperationen
  - Benötigte Daten können an unterschiedlichen Orten liegen
  - Es sind Techniken für die Optimierung der Datenabfragen notwendig, um die Netzlast zu verringern (sog. Query Optimizer)
- Hardwaretransparenz
  - Das verteilte Datenbankmanagementsystem (VDBMS) soll unabhängig von der zugrundeliegenden Hardware arbeiten
  - Das VDBMS soll sich auf jeder Hardware gleich verhalten
- Betriebssystemtransparenz
  - Das VDBMS soll unabhängig vom verwendeten Betriebssystem sein und sich auf verschiedenen Systemen gleich verhalten

10 Anforderungen an eine verteilte Datenbank

- Netzwerktransparenz
  - Das VDBMS soll unterschiedliche Netzwerkttypen unterstützen
- DBMS-Transparenz
  - Unterschiedliche SQL-Datenbankimplementierungen sollen in einem Netzwerk aufgrund des SQL-Standards zusammenarbeiten können

Vorgehen in **drei Phasen**

1. Entwurf des globalen Modells
2. Entwurf der Fragmentierung
3. Replizieren und Ortzuweisung

10 Horizontale Fragmentierung

- Horizontale Fragmentierung:

Mitarbeiter				
MNr	Name	Vorname	Ausbildung	Gehalt
⋮				

Mitarbeiter_Applikation_1				
MNr	Name	Vorname	Ausbildung	Gehalt

...

Mitarbeiter_Applikation_X				
MNr	Name	Vorname	Ausbildung	Gehalt

10 Vertikale Fragmentierung

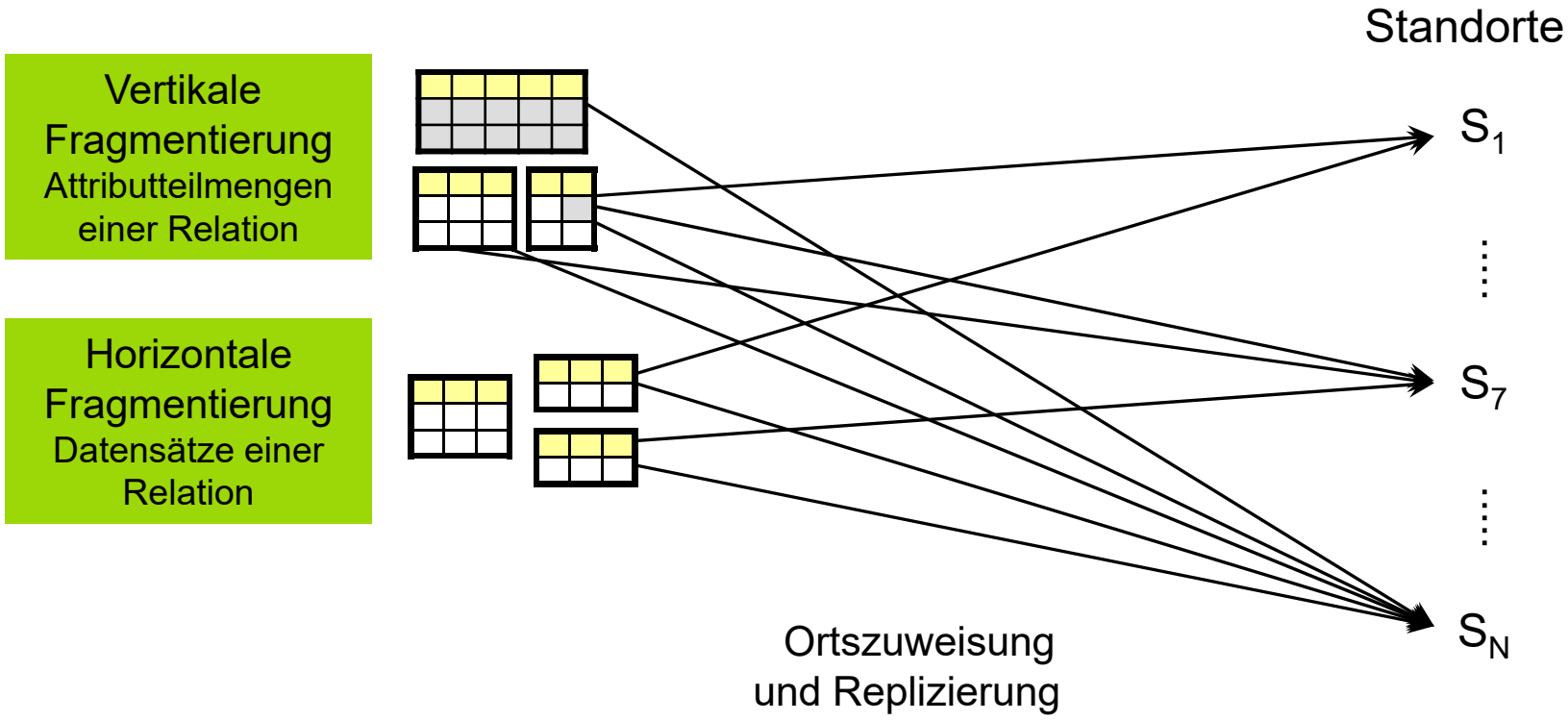
- Vertikale Fragmentierung:

Kunde									
KNr	Name	Vorname	PLZ	Ort	Straße	HNr	Geb.Dat.	Geschl.	Familienstand

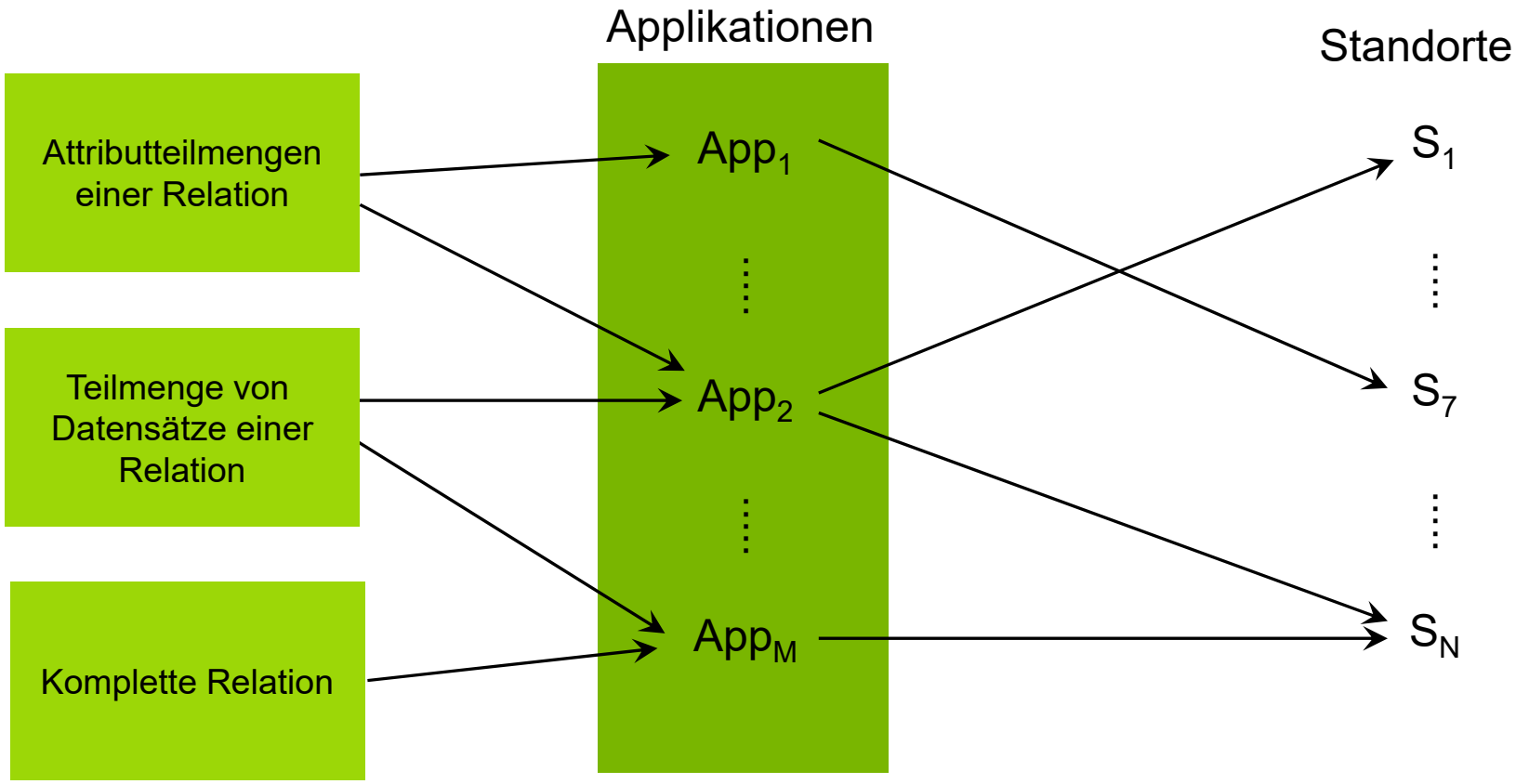
Kunde_Versand						
KNr	Name	Vorname	PLZ	Ort	Straße	HNr

Kunde_Marketing			
KNr	Geb.Dat.	Geschl.	Familienstand

10 Fragmentierung, Ortzuweisung und Replizierung



Fragmentierung, Ortzuweisung und Replizierung



10 Betrieb verteilter Datenbanken

Betriebsaufgaben:

1. Finden der relevanten Daten:
   Katalogmanagement
2. Sicherstellung der Konsistenz:
   Verteiltes Transaktionsmanagement

Aufgabe: Finden des Standortes von Relationen

Lösungsmöglichkeiten:

1. Zentralspeicherung des kompletten Katalogs
2. Komplettkatalog an jedem Standort
3. Jeder Standort hat eigenen lokalen Einzelkatalog
4. Verteilte Katalog- und Katalogverweisverwaltung

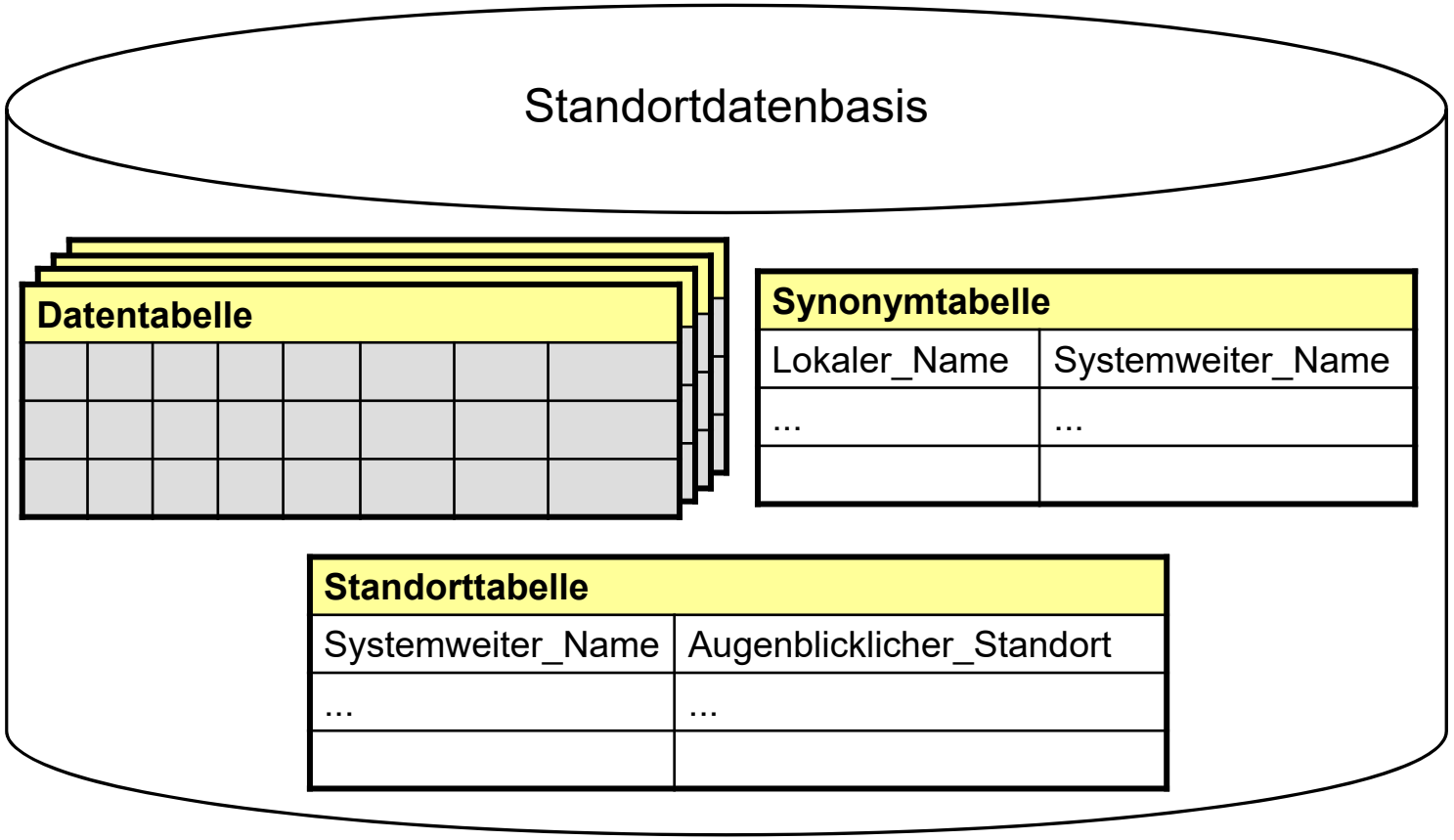
→ Systemweiter Name:

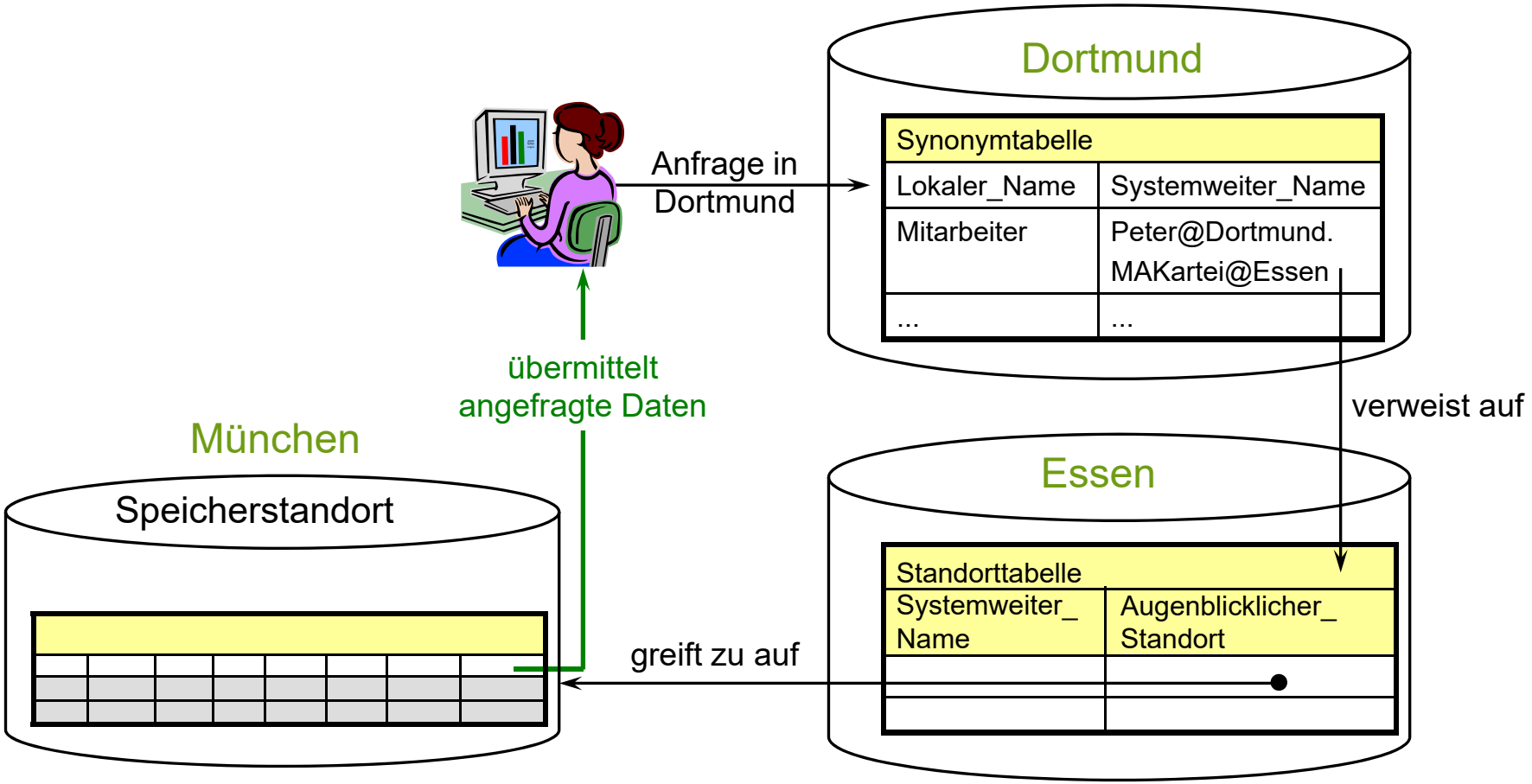
Peter@Dortmund.MAKartei@Essen

ErstellerID | *OrtsID des Erstellers* | *Gerburtsname der Tabelle* | *Geburtsort der Tabelle*

→ Synonym erstellen:

```
CREATE SYNONYM Mitarbeiter
FOR Peter@Dortmund.MAKartei@Essen
```


Standortdatenbasis
Datentabelle
Synonymtabelle
Lokaler_Name
Systemweiter_Name
...
...
Standorttabelle
Systemweiter_Name
Augenblicklicher_Standort
...
...


Dortmund
Synonymtabelle
Lokaler_Name
Systemweiter_Name
Mitarbeiter
Peter@Dortmund.
MAKartei@Essen
...
...
Anfrage in
Dortmund
übermittelt
angefragte Daten
verweist auf
München
Speicherstandort
Essen
Standorttabelle
Systemweiter_
Name
Augenblicklicher_
Standort
greift zu auf

Aufgabe: Sicherstellung der Datenintegrität
Lösungsmöglichkeiten für Sperroperationen:

- Zentraler Knoten
  - Unabhängigkeit lokaler Komponenten nicht mehr gegeben
- Schreiben sperrt alles – Lesen sperrt eins
  Write locks all – Read locks one
  - Lokale Leseoperationen möglich, falls Relation vorhanden
  - Keine Schreibsperren bei ausgefallenem Knoten möglich
- Majoritätssperren
  - Lese-/Schreibsperre gilt als gesetzt, wenn Mehrheit der Relationen gesperrt werden kann
- Verwendung von Primär- und Sekundärkopien
  - Schreiben in Primärkopie, Lesen aus Sekundärkopie

Lösungsmöglichkeiten für Deadlockbehandlung:

- **Vermeidung** durch Reihenfolgespezifikation der Datenobjekte
  - T1 und T2 möchten gleichzeitig **A_Tabelle** und **B_Tabelle** sperren
  - T1 sperrt zunächst **A_Tabelle**
  - Es würde zu einem Deadlock kommen, wenn T2 nun zuerst **B_Tabelle** sperrt
    - Dies ist jedoch aufgrund der geforderten (alphabetischen) Reihenfolge nicht gestattet!

Lösungsmöglichkeiten für Deadlockbehandlung:

- **Entdeckung und Behebung**:
  - **Timeout**
  - Analyse von **Abhängigkeitsgraphen**
    - Standorte tauschen Informationen bzgl. wartender Transaktionen aus
    - Zyklen weisen auf Deadlocks hin

Durchführung von Transaktionen

Vorbereitung:

- Transaktionskoordinator sendet PREPARE TO COMMIT an alle beteiligten Stationen
- Empfangen: Logfile schreiben
- Rückmeldung: PREPARED TO COMMIT oder NOT PREPARED
- Koordinator: Fortsetzung oder Abbruch

Finales COMMIT:

- COMMIT an alle Stationen
- Empfangen: Änderung in Datenbank
- Rückmeldung: COMMITTED oder NOT COMMITTED.
- Falls NOT COMMITTED: ABORT an alle Stationen