Datenbanken und Informationssysteme

Verteilte Datenbanken

Burkhardt Renz

Fachbereich MNI
TH Mittelhessen

Sommersemester 2019

Übersicht

- Architektur verteilter Datenbanken
  - Definition
  - Die ideale verteilte Datenbank
  - Architektur
- Speicherung in verteilten Datenbanken
- Verteilte Anfragen
- Replikation
- Verteilte Transaktionen

Gründe für verteilte Datenbanken

Beispiel 1
Großhändler mit Zentrale und mehreren Auslieferungslagern
Beispiel 2
Bestellung von Weihnachtsgeschenken bei Amazon, Bezahlung mit Kreditkarte

Charakteristik:
Beispiel 1
Gemeinsamer Datenbestand örtlich verteilt – logisch *eine* Datenbank mit verteilten Daten
Beispiel 2
Integration *zweier* Datenbanken in einem Geschäftsprozess

Uns geht es in erster Linie um verteilte Daten à la Beispiel 1.

Definition

- *ein* inhaltlich zusammengehöriger Datenbestand
- gespeichert an verschiedenen, geographisch *verteilten* Standorten

= *verteilte Datenbank*

Forderungen an eine verteilte Datenbank

Chris Date 12 + 1 Ziele

1. Lokale Autonomie
2. Keine dezidierte Abhängigkeit von einer Zentrale
3. Unterbrechungsfreier Betrieb
4. Ortsunabhängigkeit
5. Fragmentierungsunabhängigkeit
6. Replikationsunabhängigkeit
7. Verteilte Anfragen
8. Verteilte Transaktionen
9. Hardwareunabhängigkeit
10. Unabhängigkeit vom Betriebssystem
11. Unabhängigkeit vom Netz
12. Unabhängigkeit vom DBMS

Fundamentales Prinzip

Für eine Anwendung verhält sich ein verteiltes Datenbanksystem wie ein nicht-verteiltes Datenbanksystem.

Typen verteilter Datenbanken

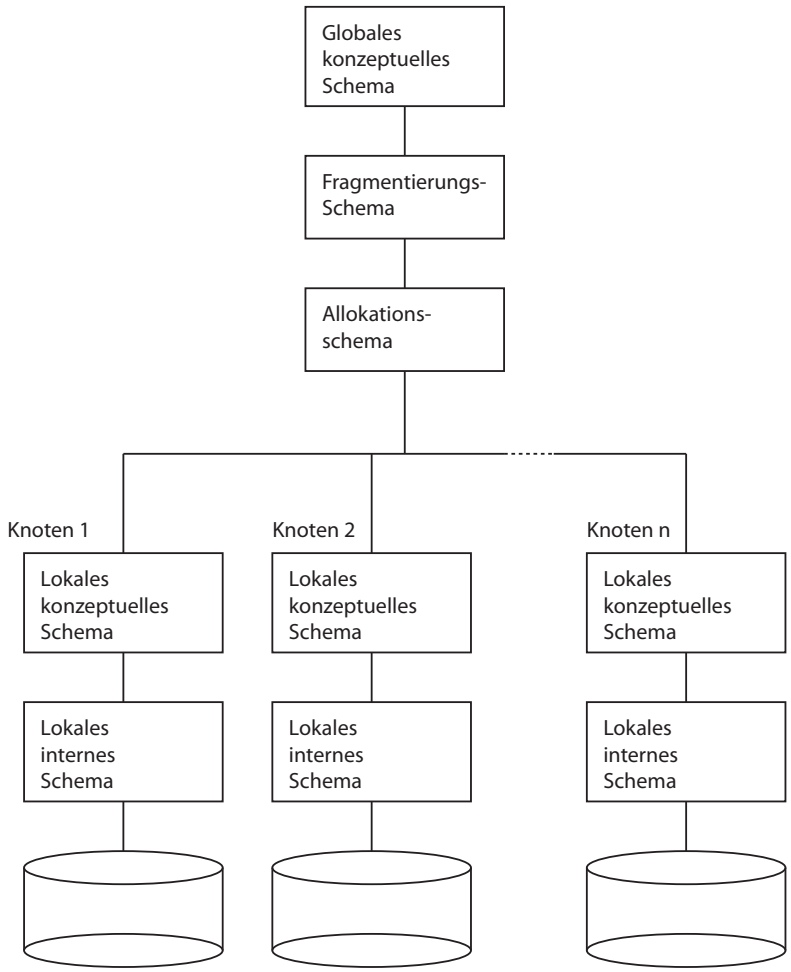
Typisierung bezüglich des DBMS:

- *homogenes* verteiltes DBMS
- *heterogenes* verteiltes DBMS

Typisierung bezüglich des Grades lokaler Autonomie:

- *eng integrierte* verteilte DB
- *föderierte* verteilte DB
- *Multidatenbanksystem*

Referenzarchitektur



Übersicht

- Architektur verteilter Datenbanken
- Speicherung in verteilten Datenbanken
  - Fragmentierung
  - Replikation
  - Katalog einer verteilten Datenbank
- Verteilte Anfragen
- Replikation
- Verteilte Transaktionen

Fragmentierung

- Horizontale Fragmentierung
  Zerlegung durch Restriktion σ
  Verbindung durch Vereinigung $\cup$
- Vertikale Fragmentierung
  Zerlegung durch Projektion π
  Verbindung durch verlustfreien Join $\bowtie$
- Gemischte Fragmentierung

Replikation

Redundante Daten in einer verteilten Datenbank. Zwei Typen:

- Synchrone Replikation
- Asynchrone Replikation

Kataloginformationen in einer verteilten Datenbank

Globaler Katalag braucht Informationen über:

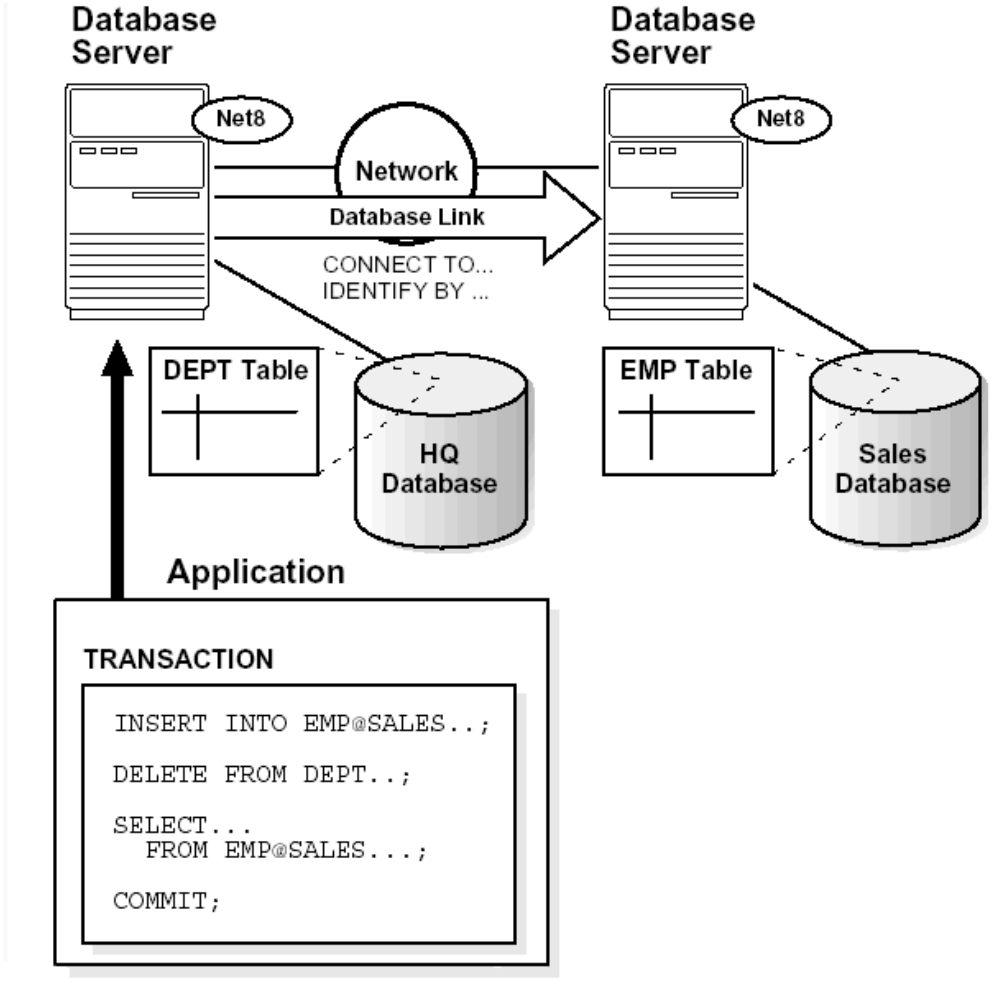
- Schemata
- Fragmentierung
- Allokation

Globaler Katalog, z.B. durch Namensdienst

Verteilter Katalog
Varianten:

- voll redundanter Katalog
- hierarchischer Clusterkatalog
- nur lokale Kataloge

Beispiel


Database Server
Net8
Network
Database Link
CONNECT TO...
IDENTIFY BY ...
Database Server
Net8
DEPT Table
HQ Database
EMP Table
Sales Database
Application
TRANSACTION
INSERT INTO EMP@SALES..;
DELETE FROM DEPT..;
SELECT...
FROM EMP@SALES...;
COMMIT;

Diskussion

Beispiel ist *nicht* ortsunabhängig:

```
insert into EMP@SALES
```

versus

```
delete from DEPT
```

Verwendung von Synonymen:

```
create public synonym EMP
    for EMP@SALES
```

→ voll redundanter Katalog

Übersicht

- Architektur verteilter Datenbanken
- Speicherung in verteilten Datenbanken
- Verteilte Anfragen
  - Ein (übertriebenes) Beispiel
  - Algorithmen für verteilte Joins
- Replikation
- Verteilte Transaktionen

Beispiel

Gegeben folgende Situation:

S(SNo, City)	10.000 Tupel Standort A
P(PNo, Color)	100.000 Tupel Standort B
SP(SNo, PNo)	1.000.000 Tupel Standort A

Jedes Tupel hat 25 Bytes = 200 Bits

Abfrage:

```
select SNo from S natural join SP natural join P
  where City = 'London' and Color = 'red'
```

Situation:
Zahl der roten Teile: 10
Zahl der Lieferungen aus London 100.000

Kostenmodell

C_1 Datenrate = Zahl übertragener Bits pro Sekunde
C_0 Initialisierungszeit = Dauer des Verbindungsaufbaus
n Menge der zu übertragenden Daten

Gesamtkosten $C = C_0 + \frac{n}{C_1}$ in Sekunden

In unserem Beispiel:
$C_1 = 50.000$ Bits pro Sekunde
$C_0 = 0.1$ Sekunden

Strategie 1

- Berechne $S \bowtie SP$ an Standort A mit Restriktion auf die Lieferanten aus `London`
  $\rightarrow$ 100.000 Datensätze
- Prüfe pro Datensatz durch einen Aufruf an Standort B, ob das Teil die Farbe `rot` hat
  $\rightarrow$ 100.000 Nachrichten

- Pro Nachricht wird die `PNo` hin-, die `Color` zurückübertragen:
  $C = 0.1 + \frac{200}{50.000}$
- Also für 100.000 Nachrichten:
  $C = 0.1 \times 100.000 + 400 \approx 10.000$ Sekunden
  $\rightarrow \quad \approx 2.78$ Stunden

Strategie 2

- Übertrage die komplette Tabelle P von Standort B an Standort A $\rightarrow$ 100.000 Datensätze in 1 Nachricht
- Berechne die Ergebnismenge an Standort A

- Übertragung von 100.000 Datensätzen:
  $C = 0.1 + \frac{100.000 \times 200}{50.000} = 400$ Sekunden
  $\rightarrow \quad \approx 6.67$ Minuten

Strategie 3

- Berechne an Standort B die Restriktion von `P` auf die Teile mit `Color = 'red'` und übertrage diese Datensätze an Standort A $\rightarrow$ 10 Datensätze in 1 Nachricht
- Berechne die Ergebnismenge an Standort A

- Übertragung von 10 Datensätzen:
  $C = 0.1 + \frac{10 \times 200}{50.000}$
  $\rightarrow \quad \approx 0.1$ Sekunden

Semijoin

`select * from SP natural join P`
wobei SP an Standort A und P an Standort B

1. Berechne an Standort A die Projektion SP_P auf die Join-Attribute, d.h.
   $SP_P = \pi_{PNo}(SP)$
2. Übertrage SP_P an Standort B
3. Berechne an Standort B den Join
   $SP_P \bowtie P = P_R$
   die *Reduktion* von P bezüglich SP.
4. Übertrage P_R an Standort A
5. Berechne nun den Join
   $SP \bowtie P_R$

Bloomjoin, Bitvektor-Verbund

Das Joinattribut `PNo` sei vom Typ Integer

1. Wähle Modul n für Hash-Funktion $h(x) = x \bmod n$.
2. Bilde einen Bitvektor b der Länge n:
   $b_i = 1$ es gibt eine `PNo` x mit $h(x) = i$
   $b_i = 0$ sonst.
3. Übertrage die Zahl n sowie den Bitvektor an Standort B.
4. Selektiere an Standort B diejenigen Zeilen von `P`, für die gilt:
   Für die `PNo` x ist $b_{h(x)} = 1$
   Auf diese Weise bildet man P_R, die *Reduktion* von P bezüglich des Bitvektors.
5. Übertrage P_R an Standort A.
6. Berechne nun den Join
   $SP \bowtie P_R$

Beispiel Oracle

Der Optimizer kann verteilte Anfragen optimieren.
Beispiel einer Anfrage:

```
select l.a, l.b, r.c, r.d, r.e
  from local l, remote r
  where l.c = r.c and r.e > 300
```

Formulierung der Anfrage mit einer sogenannten collocated inline view:

```
select l.a, l.b, v.c, v.d, v.e
  from local l,
     (select r.c, r.d, r.e from remote r where r.e  > 300) as v
  where l.c = v.c
```

Übersicht

- Architektur verteilter Datenbanken
- Speicherung in verteilten Datenbanken
- Verteilte Anfragen
- Replikation
  - Techniken
  - Beispiele
- Verteilte Transaktionen

Techniken synchroner Replikation

- ROWA-Verfahren
  read one write all
- Abstimmverfahren (Quorum)
  z.B. 7 aus 10

Techniken asynchroner Replikation

- Replikation mittels Masterkopie
  aka Publisher-Subscriber-Verfahren
  - Push-Modell: Publisher initiativ
  - Pull-Modell: Subscriber initiativ
- Peer-to-Peer-Replikation
  braucht Konfliktlösungsstrategie wie z.B.
  - Organisatorische Lösung
  - Priorisierung
  - Zeitliche Kriterien

Beispiele

- Oracle
  - Replication Groups
  - Master- und Sekundärkopien = asynchrone Replikation mittels Masterkopie
  - Multimaster-Replikation
- Microsoft SQL Server
  - Snapshot-Replikation
  - Transaktionsreplikation
  - Mergereplikation

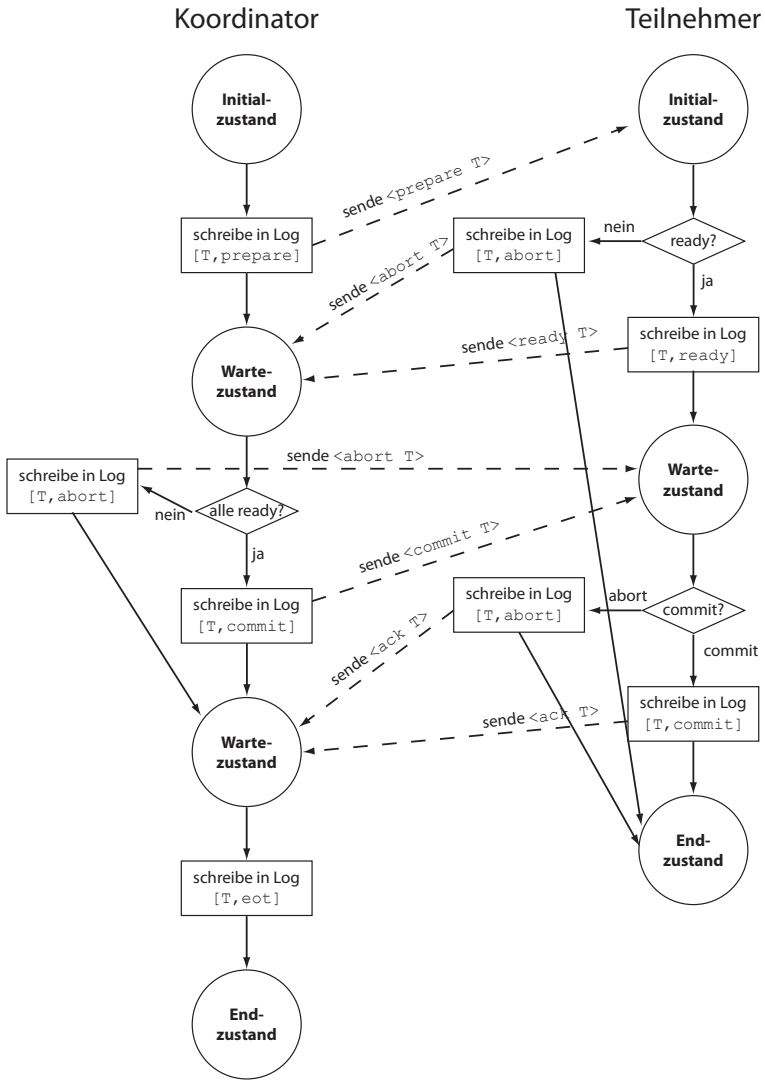
Übersicht

- Architektur verteilter Datenbanken
- Speicherung in verteilten Datenbanken
- Verteilte Anfragen
- Replikation
- Verteilte Transaktionen
  - 2-Phasen-Commit-Protokoll
  - Beispiele

Beteiligte verteilter Transaktionen

- *Transaktionsmanager* im lokalen DBMS
- *Transaktionskoordinator* für verteilte Transaktion
- *Globale* Transaktion
- *Lokale* Subtransaktionen
- *Commit* der globalen Transaktion
  Zusammenspiel der Teilnehmer unter Leitung des Koordinators
  → 2-Phasen-Commit (2PC)

2-Phasen-Commit-Protokoll



Diskussion

Timeout in einem der Zustände:

- Timeout des Koordinators im Wartezustand auf Votum
- Timeout des Koordinators im Wartezustand auf Ack
- Timeout des Teilnehmers im Initialzustand
- Timeout des Teilnehmers im Wartezustand

Ausfall eines der Beteiligten:

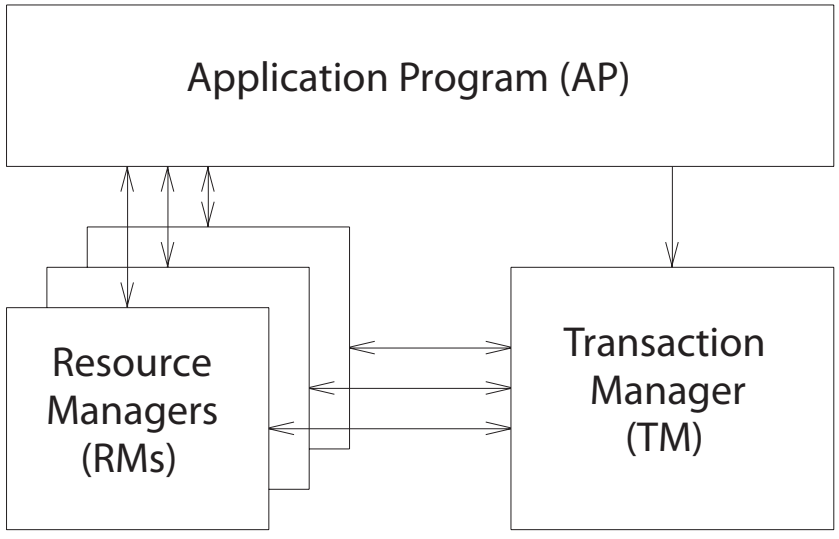
- Ausfall des Koordinators im Initialzustand
- Ausfall des Koordinators im Wartezustand auf Votum
- Ausfall des Koordinators im Wartezustand auf Ack
- Ausfall des Teilnehmers im Initialzustand
- Ausfall des Teilnehmers im Wartezustand

X/Open Distributed Transaction Processing

- Resource Manager (RM)
  stellen Dienste zur Verfügung, die in Transaktionen verwendet werden können
  z.B. DBMS, Printserver, Mailing, Queuing-Systeme
  unterstützen XA-Schnittstelle
- Transaktionsmanager (TM)
  steuert die verteilte Transaktion und verwendet dazu die XA-Schnittstelle der Resource Manager
  implementiert TX-Schnittstelle
- Anwendung
  verwendet die Dienste der Resource Manager und die TX-Schnittstelle des Transaktionsmanagers für die verteilte Transaktion

Konzept

(1) AP uses resources from a set of RMs



(2) AP defines transaction boundaries through the TX interface

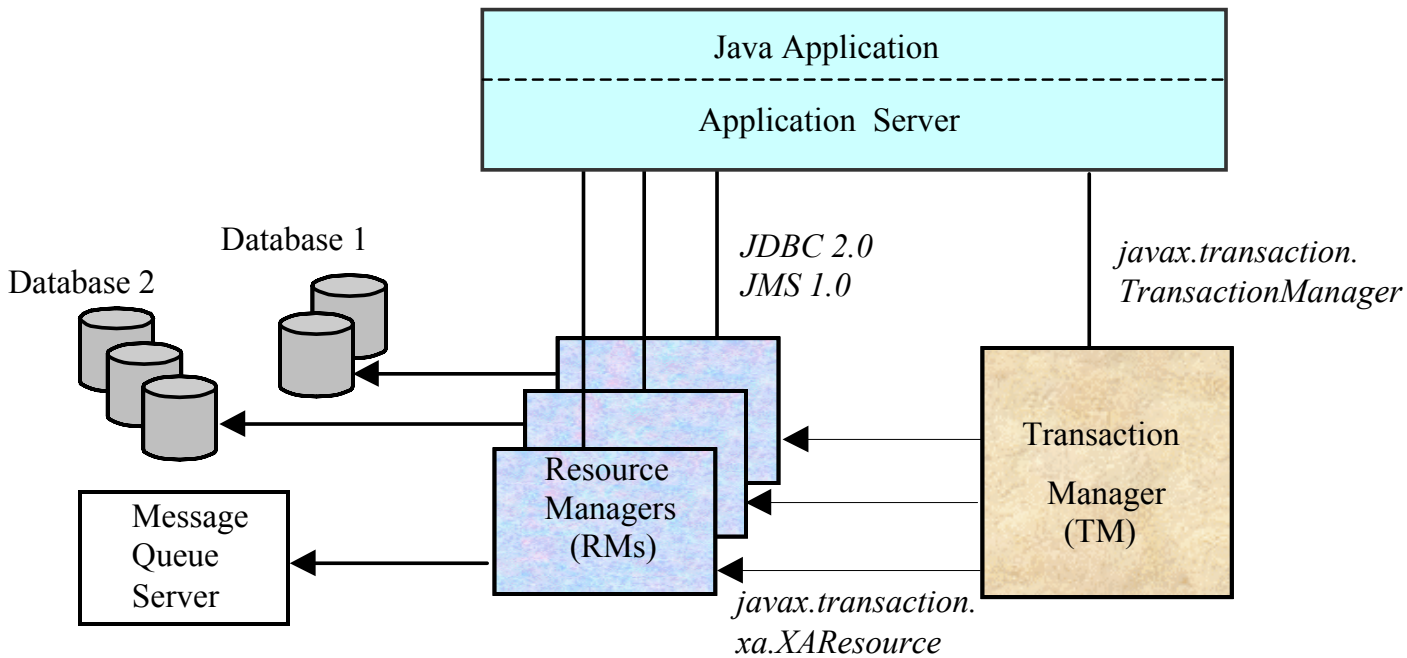
(3) TM and RMs exchange transaction information

Quelle: The Open Group XA-Specification

XA-Schnittstelle

- TX-Schnittstelle
  - tx_begin: Beginn der Transaktion
  - tx_commit: Commit der verteilten Transaktion
  - tx_rollback: Rollback der verteilten Transaktion
- XA-Schnittstelle
  - xa_open: Initialisiere Resource Manager für die Zusammenarbeit mit dem Transaktionsmanager
  - xa_start: Beginn einer Subtransaktion
  - xa_prepare: Prepare des 2PC
  - xa_commit: Mitteilung über globales Commit
  - xa_rollback: Mitteilung über globalen Abbruch
  - xa_end: Ende einer Subtransaktion
  - xa_close: Beende Verwendung des Resource Managers

Beispiel Java Transaction API (JTA)



Quelle: Spezifikation JTA Java Transaction API

Beispiele MS SQL Server

- Dates Vision: Verteilte Transaktion wird gesteuert vom DBMS
- DBMS verwendet Fragmentierungs- und Allokationsschema dafür
- In der Praxis häufig: Anwendung steuert die verteilte Transaktion
- Beispiele mit TransactSQL und ADO.NET
  lokal: Tabelle Books, entfernt: Tabelle Authors
  Wir wollen ein Buch und einen Autor einfügen

Linked Server bei Microsoft SQL-Server

```
-- Linked Server einrichten
EXEC sp_addlinkedserver @server='RemoteServer',
        @srvproduct='', @provider='SQLNCLI',
        @datasrc='Server2/SQLEXPRESS'

-- Login auf dem Linked Server
EXEC sp_addlinkedsrvlogin 'RemoteServer', 'false'
        'sa', 'guest'. 'passwd'
```

Verbundener Server in verteilter Transaktion

In der lokalen Tabelle Books wird ein Buchtitel, in der entfernten Tabelle Authors der Autor eingefügt
Das Beispiel setzt voraus, dass auf beiden Rechnern der DTC (*Distributed Transaction Coordinator*) läuft

```
SET XACT_ABORT ON -- bei Fehler soll rollback gemacht werden

BEGIN DISTRIBUTED TRANSACTION

  -- Titel in lokalen Tabelle einfuegen
  INSERT INTO Books values (...)

  - Autor in entfernte Tabelle einfuegen
  INSERT INTO RemoteServer.DBName.dbo.Authors values (...)

COMMIT  -- hier wird 2PC vom DTC gesteuert
```

DTC verwendet ein Microsoft-spezifisches Protokoll, kann aber auch XA-konforme verteilte Transaktionen machen.

Beispiel mit ADO.NET

```
...
try {
  using ( TransactionScope ts = new TransactionScope() ) {

    using (SqlConnection con1 = new SqlConnection(connectString1) ){
      con1.Open();
      SqlCommand cmd1 = new SqlCommand("insert...", con1);
      cmd1.ExecuteNonQuery();

      // wenn man hier ist, war cmd1 erfolgreich
      using SqlConnection con2 = new SQLConnection(connectString2) ){
        con2.Open();
        SqlCommend cmd2 = new SqlCommand("insert...", con2);
        cms2.ExecuteNonQuery() ;
      }
    }
    // Commit; wenn Exception auftrat, wird dies nicht aufgerufen
    ts.Complete();
  }
} catch (TransactionAbortedException ex) {
  // Transaktion wurde abgebrochen, Rollback erfolgt
}
...
```

Diskussion, 1

- Dates fundamentales Prinzip praktisch nicht verwirklicht in den gängigen SQL-Systemen
- CAP (Eric Brewer 2000, Gilbert & Lynch 2002)
  Folgende drei Eigenschaften können in einem verteilten System nicht gleichzeitig erfüllt sein:
  - **C = Konsistenz**: Alle Knoten sehen zur selben Zeit denselben Zustand der Daten
  - **A = Verfügbarkeit**: Anfragen werden stets beantwortet
  - **P = Partitionstoleranz**: Verteiltes System arbeitet auch bei einer Partition des Netzes weiter.
- **BASE** *Basically Available, Soft state, Eventual consistency* $\Rightarrow$ NoSQL-Systeme

Diskussion, 2

> *In a system that cannot count on distributed transactions, the management of uncertainty must be implemented in the business logic.*
>
> *Pat Helland*

Lektüre

Pat Helland: *Life Beyond Distributed Transactions*
Comm. of the ACM, Febr 2017

Diskussion, 3

> *NOSQL was born out of the necessity to build highly available systems that operate at scale at a comparatively low cost.*
>
> *Transactions are essential for mission-critical applications ...*
>
> *... embracing the relational model and SQL natively, in a tightly coupled fashion has been the latest big leap...*

Lektüre

David F Bacon et al: *Spanner: Becoming a SQL System*
SIGMOD'17, Mai 2017