



Einsatz und Realisierung von Datenbanksystemen

ERDB Übungsleitung

i3erdb@in.tum.de

Folien erstellt von Maximilian Bandle & Alexander Beischl



Organisatorisches

Disclaimer

Die Folien werden von der Übungsleitung allen Tutoren zur Verfügung gestellt.

Sollte es Unstimmigkeiten zu den Vorlesungsfolien von Prof. Kemper geben, so sind die Folien aus der Vorlesung ausschlaggebend.

Falls Ihr einen Fehler oder eine Unstimmigkeit findet, schreibt an i3erdb@in.tum.de mit Angabe der Foliennummer.

Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

- Ausgleich der Leistungsfähigkeit zwischen Lese- und Änderungstransaktionen
- ➔ Teilweise Verlagerung des Overheads von Änderungs- zu Lesetransaktionen:
 - Kopien A_i von A werden individuelle Gewichte zugeordnet
- Lesequorum $Q_r(A)$
- Schreibquorum $Q_w(A)$

Folgende Bedingungen müssen gelten:

1. $Q_w(A) + Q_w(A) > W(A)$
2. $Q_r(A) + Q_w(A) > W(A)$

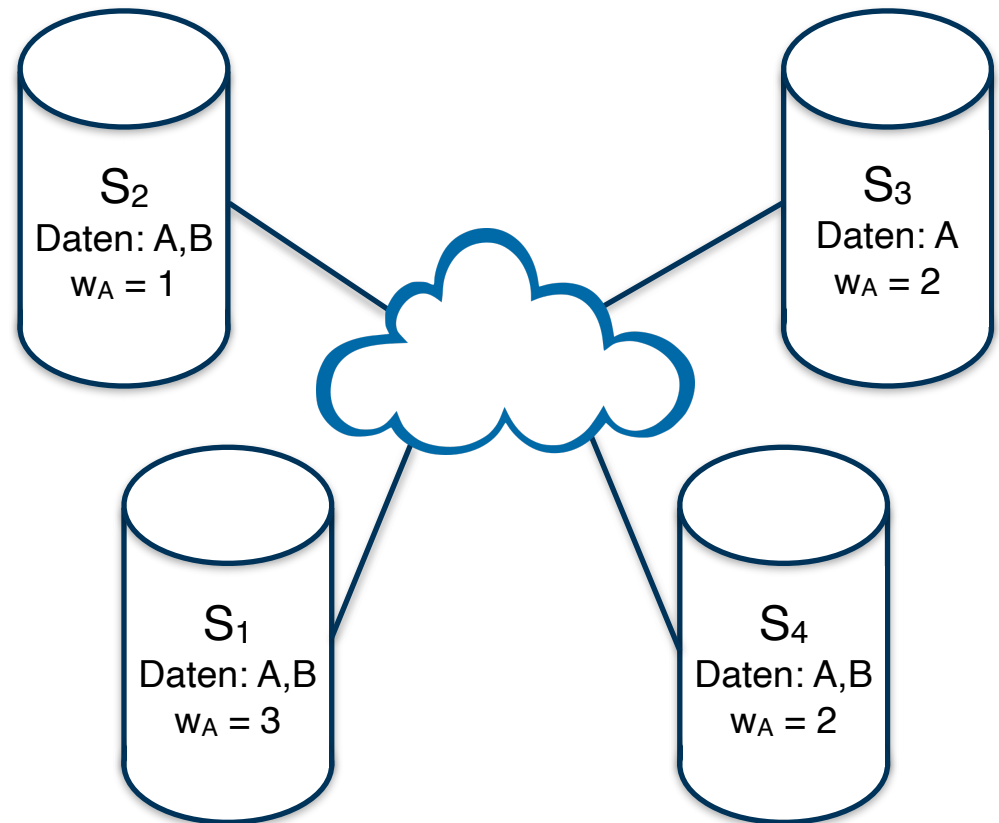
Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Berechne das Schreib- Q_w und Lesequorum Q_r für A und B.

$$W(A) = \sum_{i=1} w_i(A)$$

1. $Q_w(A) + Q_w(A) > W(A)$
2. $Q_r(A) + Q_w(A) > W(A)$



Verteilte Datenbanksysteme

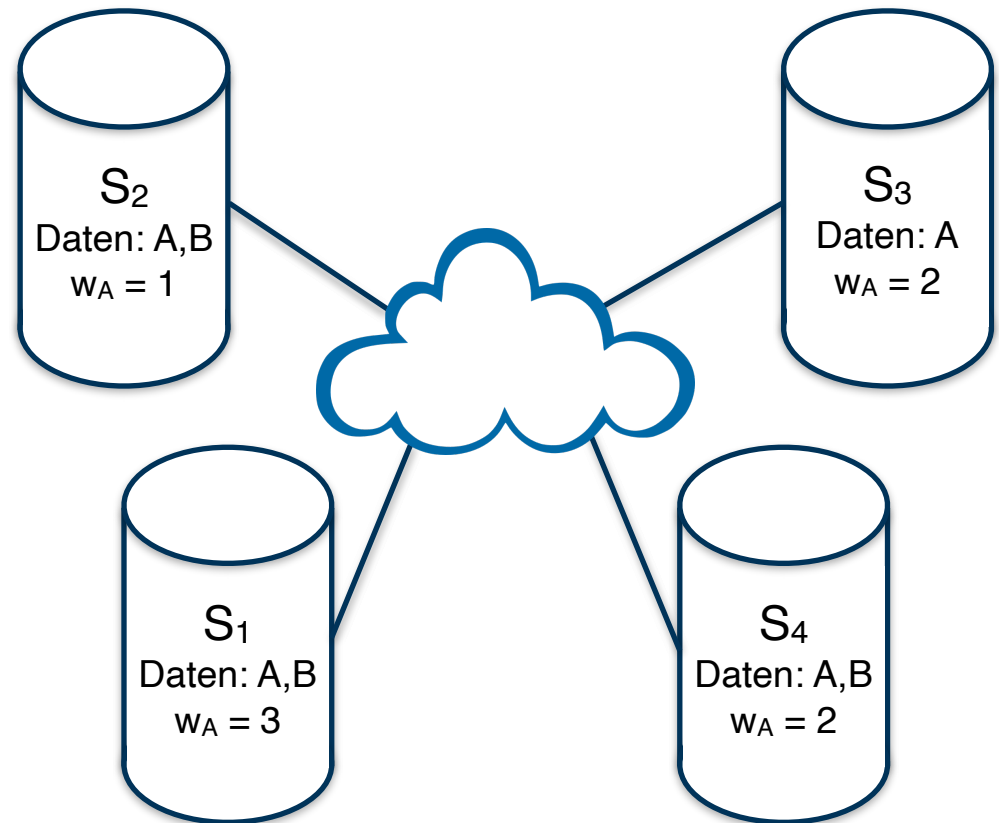
Quorum-Consensus Verfahren

Berechne das Schreib- Q_w und Lesequorum Q_r für A und B.

Für A:

1. Gesamtgewicht berechnen:

$$W(A) = 3+1+2+2 = 8$$



Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Berechne das Schreib- Q_w und Lesequorum Q_r für A und B.

Für A:

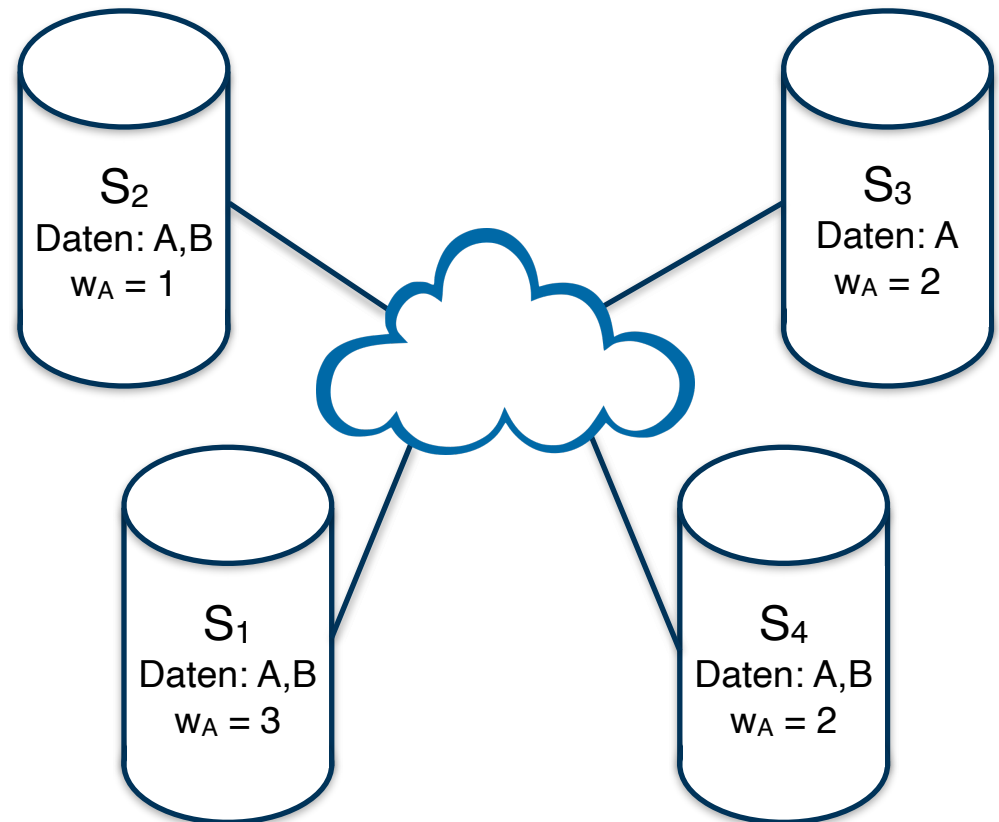
2. Schreibquorum berechnen

$$Q_w(A) + Q_w(A) > W(A)$$

$$\Rightarrow 2 * Q_w(A) > 8 \quad |:2$$

$$\Rightarrow Q_w(A) > 4$$

$$\Rightarrow Q_w(A) = 5$$



Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Berechne das Schreib- Q_w und Lesequorum Q_r für A und B.

Für A:

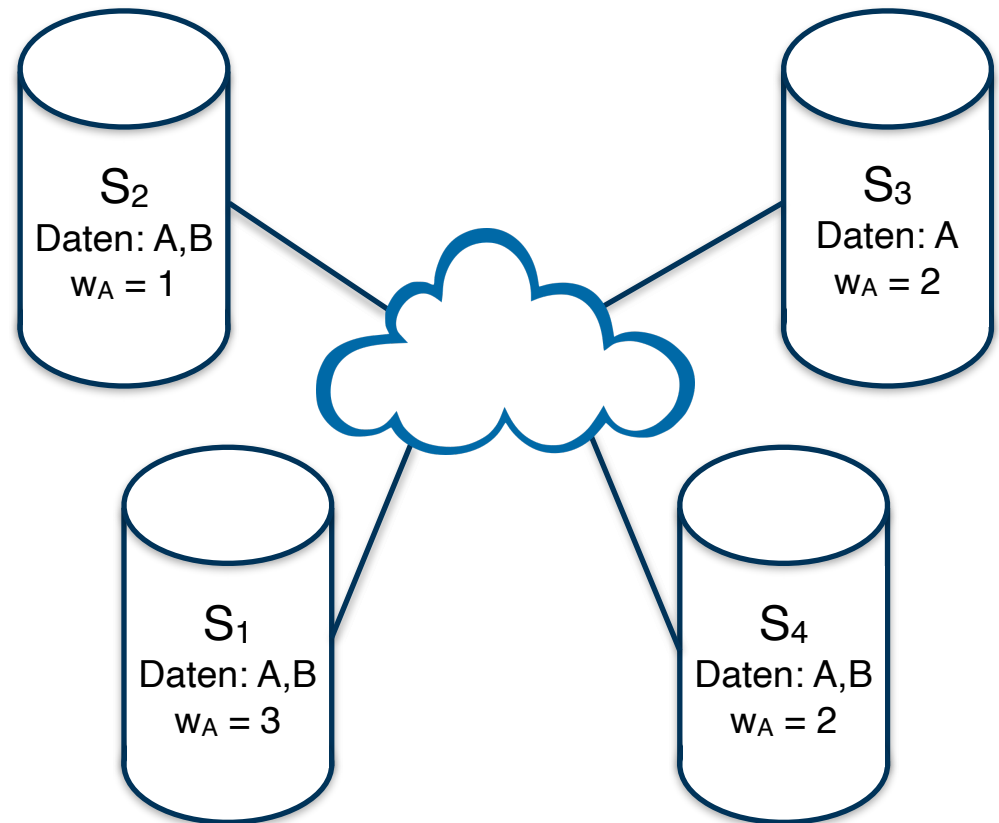
2. Lesequorum berechnen

$$Q_r(A) + Q_w(A) > W(A)$$

$$\Rightarrow Q_r(A) + 5 > 8 \quad /-5$$

$$\Rightarrow Q_r(A) > 3$$

$$\Rightarrow Q_r(A) = 4$$



Verteilte Datenbanksysteme

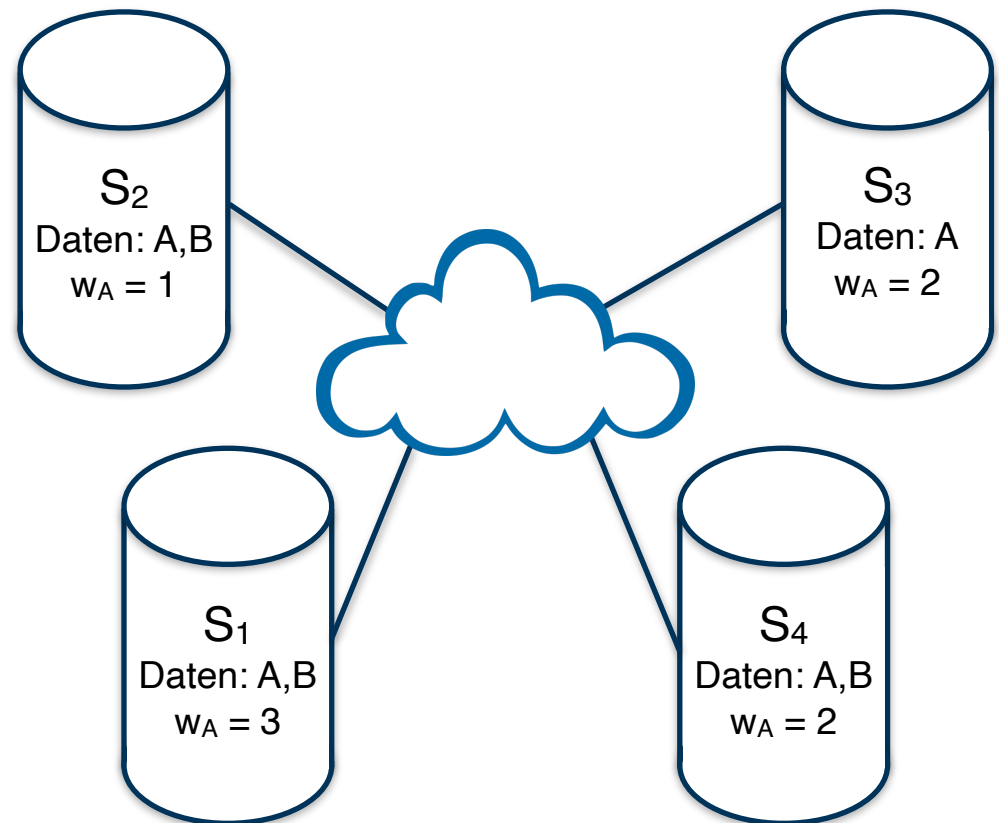
Quorum-Consensus Verfahren

Berechne das Schreib- Q_w und Lesequorum Q_r für A und B.

Für B:

1. Gesamtgewicht berechnen:

$$W(B) = 2+1+3 = 6$$



Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Berechne das Schreib- Q_w und Lesequorum Q_r für A und B.

Für B:

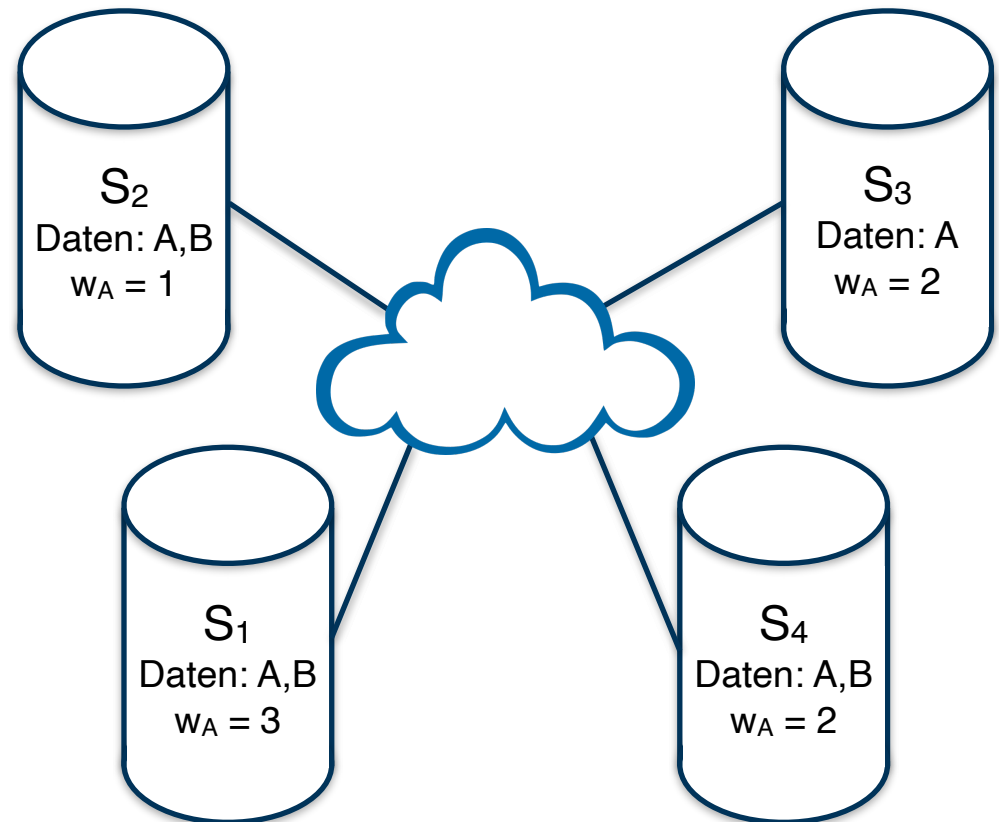
2. Schreibquorum berechnen

$$Q_w(B) + Q_w(B) > W(B)$$

$$\Rightarrow 2 * Q_w(B) > 6 \quad | :2$$

$$\Rightarrow Q_w(B) > 3$$

$$\Rightarrow Q_w(B) = 4$$



Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Berechne das Schreib- Q_w und Lesequorum Q_r für A und B.

Für B:

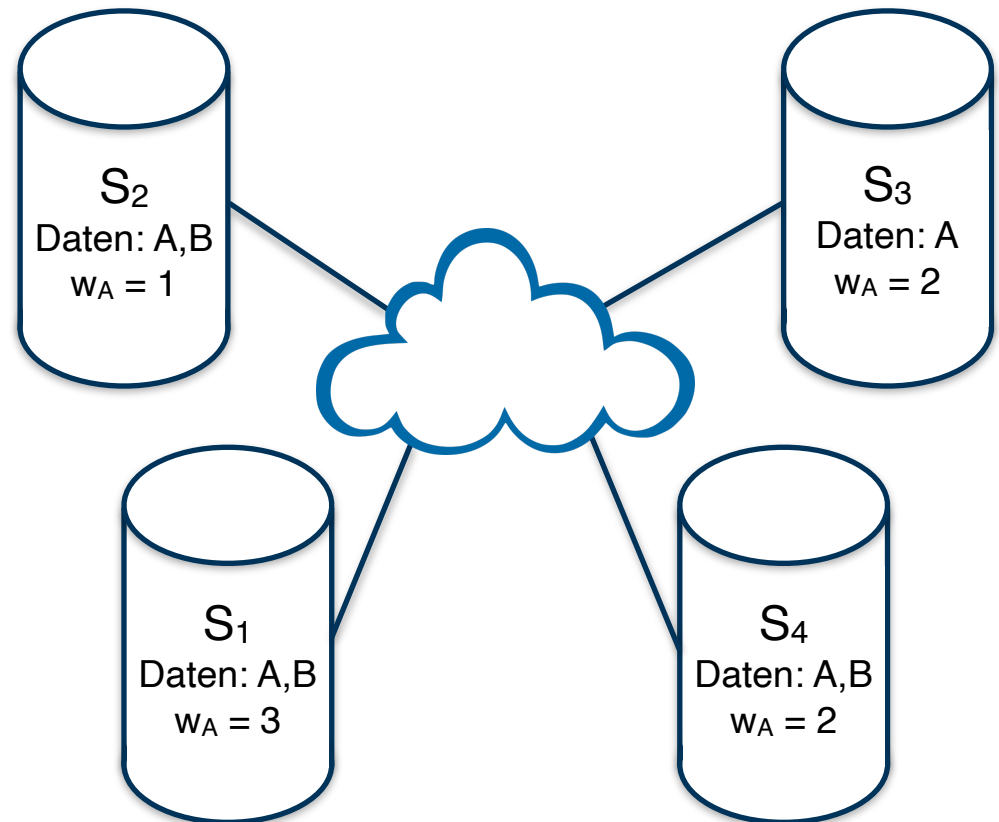
2. Lesequorum berechnen

$$Q_r(B) + Q_w(B) > W(B)$$

$$\Rightarrow Q_r(B) + 4 > 6 \quad | -4$$

$$\Rightarrow Q_r(B) > 2$$

$$\Rightarrow Q_r(B) = 3$$



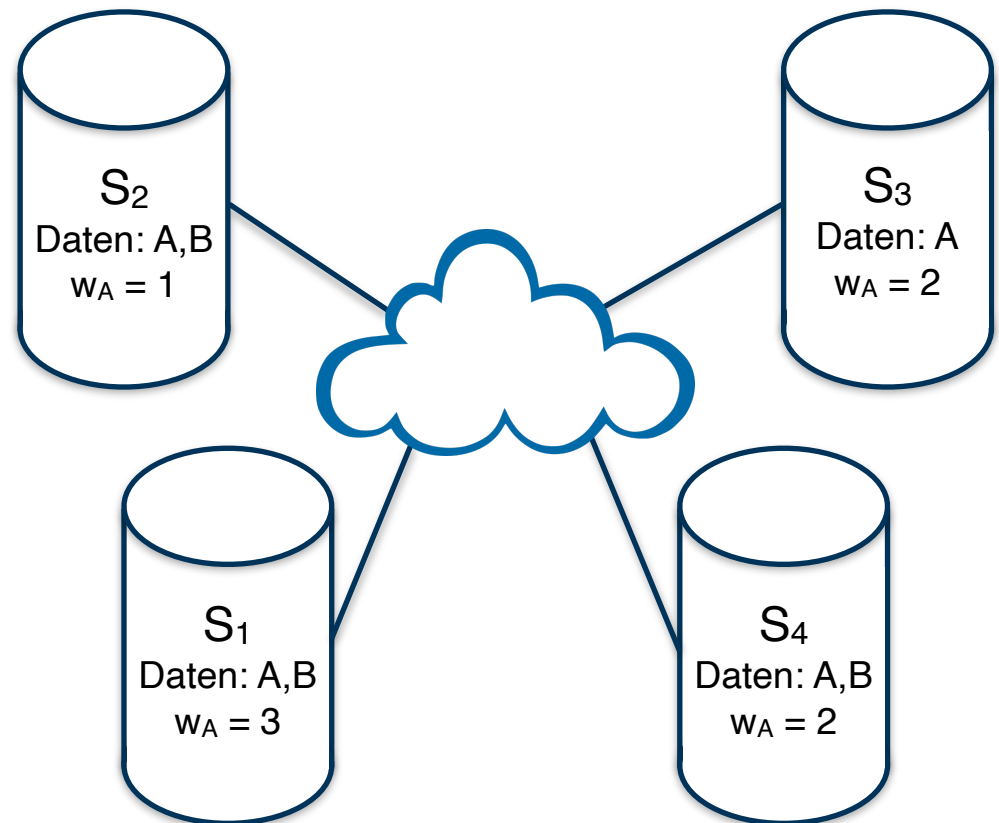
Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Schreibe:

- $A = 5$
- $B = 2$

Station	Daten	Gewichte	Version
S ₁	A = 22 B = 7	w _A = 3 w _B = 2	A ₁ B ₁
S ₂	A = 22 B = 7	w _A = 1 w _B = 1	A ₁ B ₁
S ₃	A = 22	w _A = 2	A ₁
S ₄	A = 22 B = 7	w _A = 2 w _B = 3	A ₁ B ₁



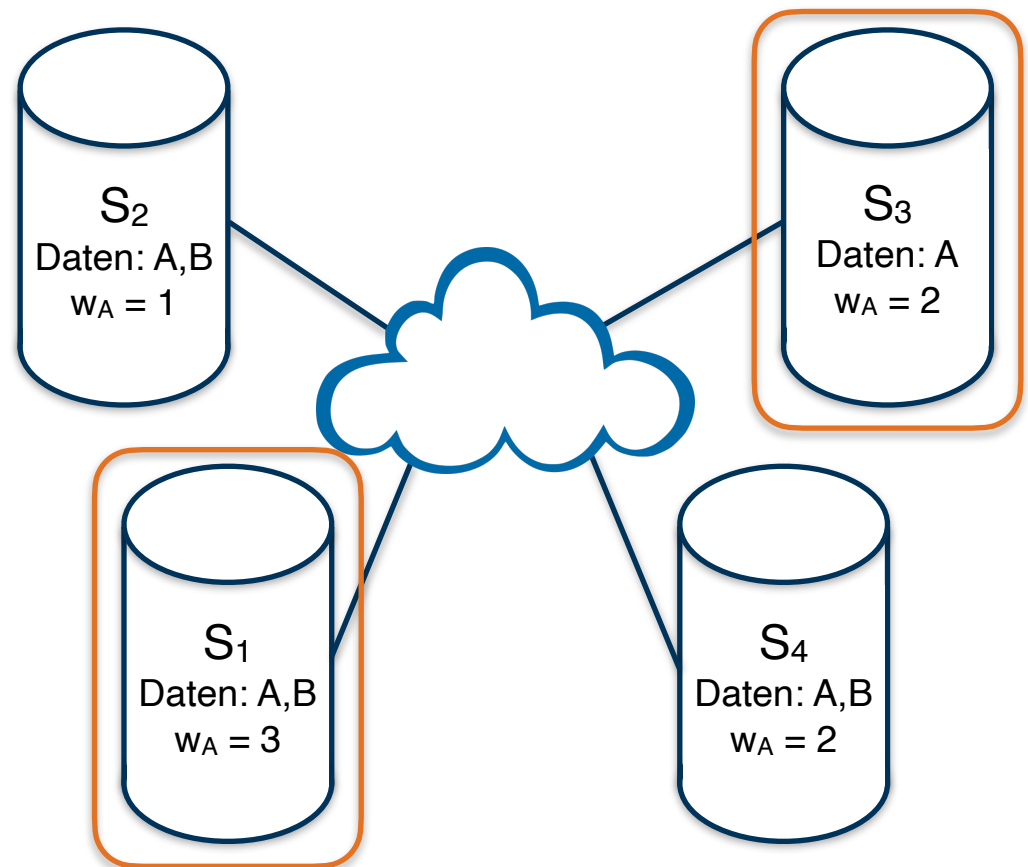
Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Schreibe:

- $A \leftarrow 5$
- $B \leftarrow 2$

Station	Daten	Gewichte	Version
S ₁	A = 22 B = 7	w _A = 3 w _B = 2	A ₁ B ₁
S ₂	A = 22 B = 7	w _A = 1 w _B = 1	A ₁ B ₁
S ₃	A = 22	w _A = 2	A ₁
S ₄	A = 22 B = 7	w _A = 2 w _B = 3	A ₁ B ₁



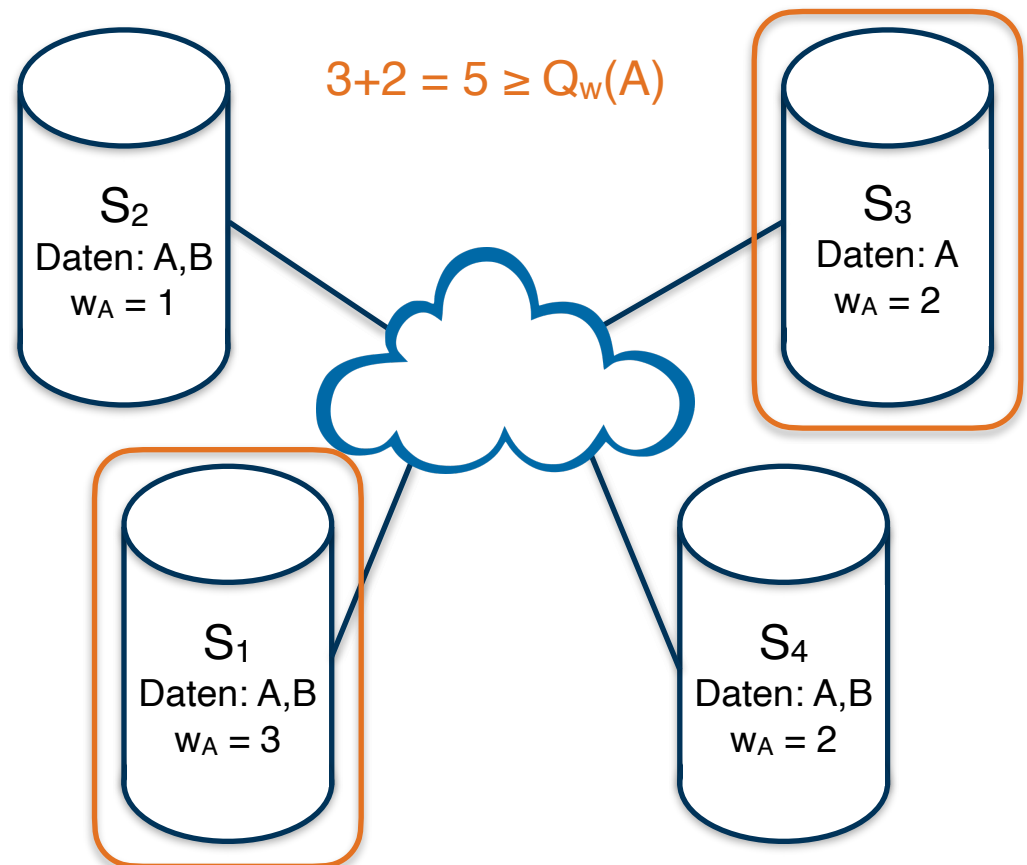
Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Schreibe:

- $A \leftarrow 5$
- $B \leftarrow 2$

Station	Daten	Gewichte	Version
S ₁	A = 22 B = 7	w _A = 3 w _B = 2	A ₁ B ₁
S ₂	A = 22 B = 7	w _A = 1 w _B = 1	A ₁ B ₁
S ₃	A = 22	w _A = 2	A ₁
S ₄	A = 22 B = 7	w _A = 2 w _B = 3	A ₁ B ₁



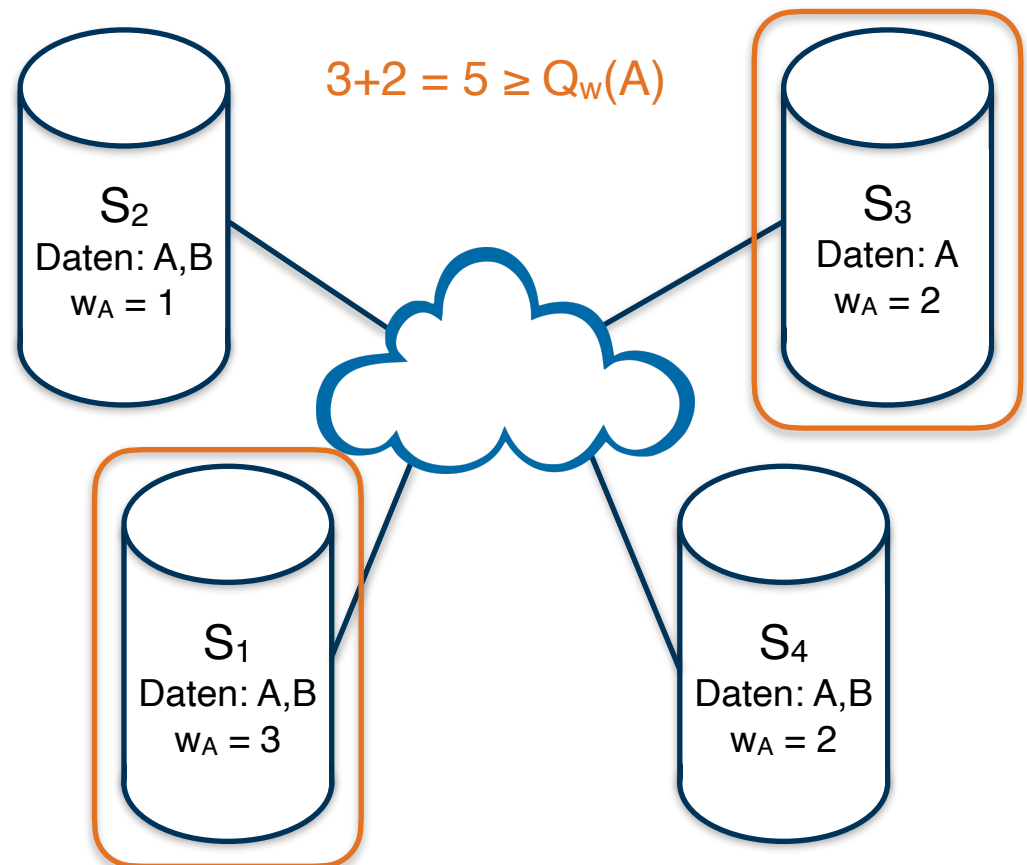
Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Schreibe:

- $A \leftarrow 5$
- $B \leftarrow 2$

Station	Daten	Gewichte	Version
S ₁	$A = 17$ $B = 7$	$w_A = 3$ $w_B = 2$	A_2 B_1
S ₂	$A = 22$ $B = 7$	$w_A = 1$ $w_B = 1$	A_1 B_1
S ₃	$A = 17$	$w_A = 2$	A_2
S ₄	$A = 22$ $B = 7$	$w_A = 2$ $w_B = 3$	A_1 B_1



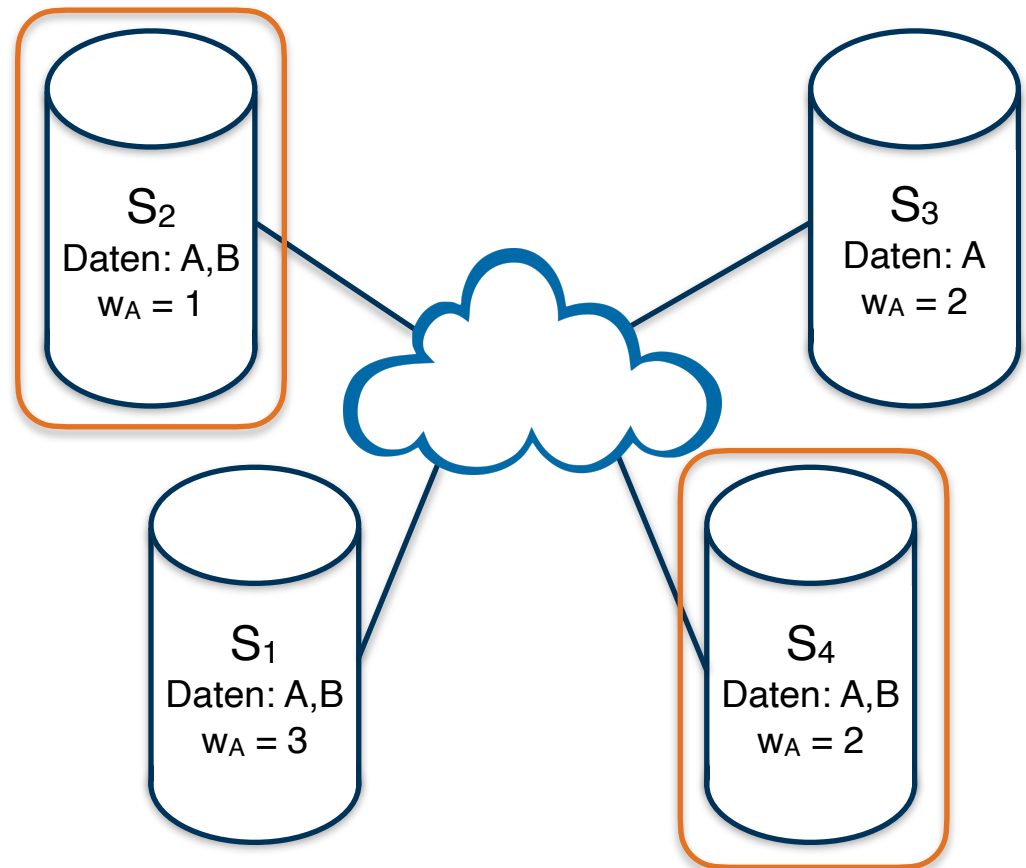
Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Schreibe:

- $A = 5$
- $B = 2$

Station	Daten	Gewichte	Version
S ₁	A = 17 B = 7	w _A = 3 w _B = 2	A ₂ B ₁
S ₂	A = 22 B = 7	w _A = 1 w _B = 1	A ₁ B ₁
S ₃	A = 17	w _A = 2	A ₂
S ₄	A = 22 B = 7	w _A = 2 w _B = 3	A ₁ B ₁



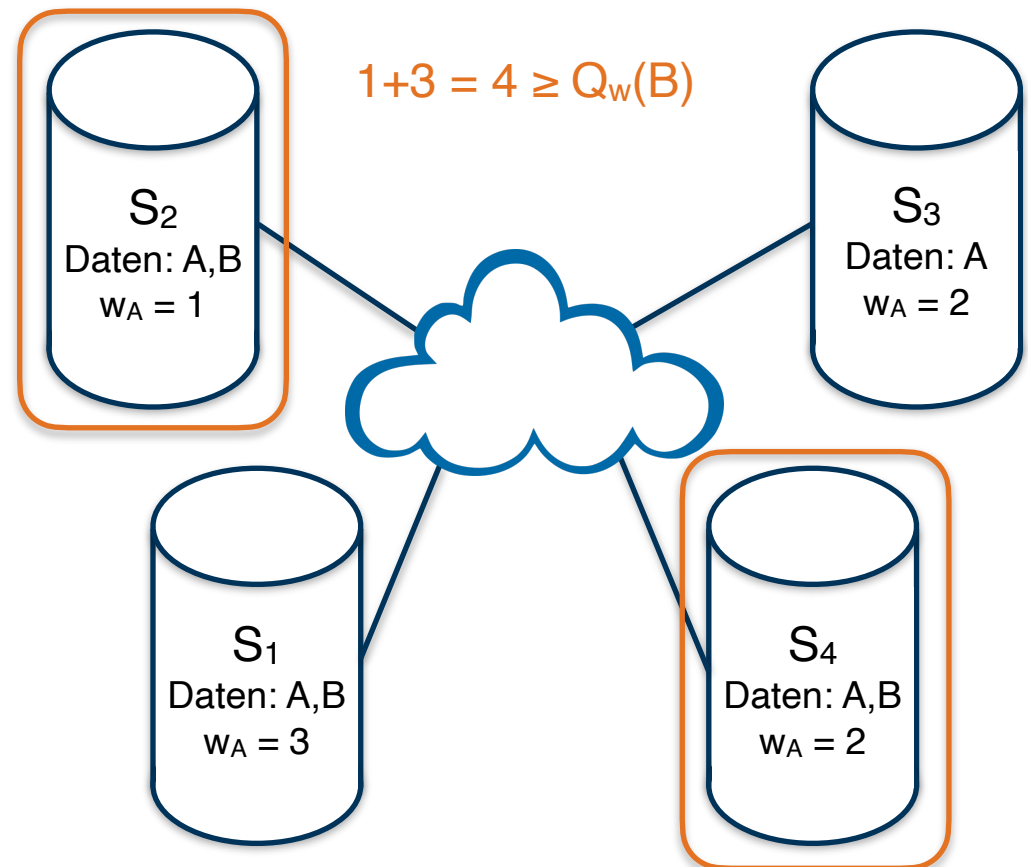
Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Schreibe:

- $A = 5$
- $B = 2$

Station	Daten	Gewichte	Version
S ₁	A = 17 B = 7	w _A = 3 w _B = 2	A ₂ B ₁
S ₂	A = 22 B = 7	w _A = 1 w _B = 1	A ₁ B ₁
S ₃	A = 17	w _A = 2	A ₂
S ₄	A = 22 B = 7	w _A = 2 w _B = 3	A ₁ B ₁



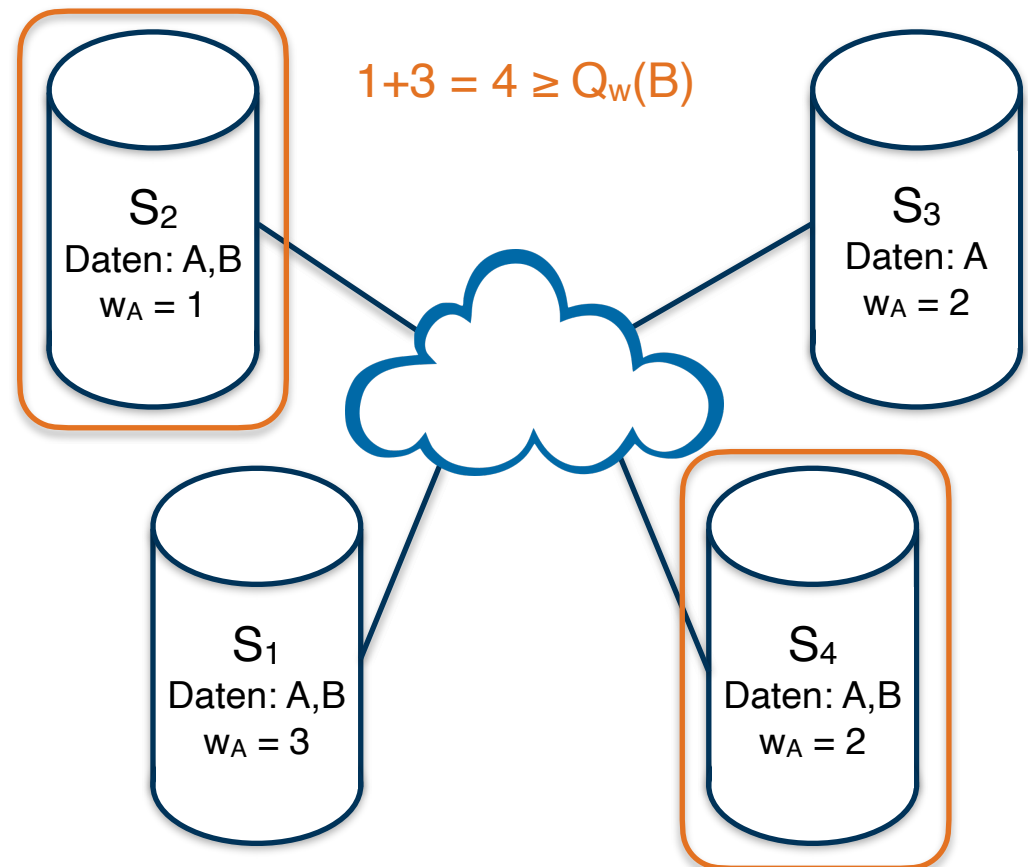
Verteilte Datenbanksysteme

Quorum-Consensus Verfahren

Schreibe:

- A = 5
- B += 2

Station	Daten	Gewichte	Version
S ₁	A = 17 B = 7	w _A = 3 w _B = 2	A ₂ B ₁
S ₂	A = 22 B = 9	w _A = 1 w _B = 1	A ₁ B ₂
S ₃	A = 17	w _A = 2	A ₂
S ₄	A = 22 B = 9	w _A = 2 w _B = 3	A ₁ B ₂



Aufgabe 1

Zeigen Sie, dass die *write-all/read-any* Methode zur Synchronisation replizierter Daten einen Spezialfall der *Quorum-Consensus*-Methode darstellt.

- Für welche Art von Workloads eignet sich dieses Verfahren besonders gut?
- Wie werden Stimmen zugeordnet um *write-all/read-any* zu simulieren?
- Wie müssen die Quoren Q_w und Q_r vergeben werden?

Aufgabe 2

Um Ausfallsicherheit zu garantieren ist ein Datenwert 'A' auf vier Rechnern verteilt. Jeder Rechner hält dabei eine vollständige Kopie von 'A'. Um Konsistenz zu garantieren wird das Quorum-Consensus-Verfahren eingesetzt. Dabei ist jedem Rechner ein Gewicht $w_i(A)$ wie folgt zugewiesen:

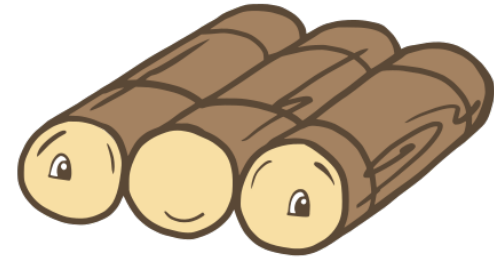
Rechner	Kopie	Gewicht
R_1	A_1	3
R_2	A_2	1
R_3	A_3	2
R_4	A_4	2

Das Lesequorum ist $Q_r(A) = 4$ und das Schreibquorum ist $Q_w(A) = 5$.

- Geben Sie **alle** Lesemöglichkeiten für eine Transaktion auf dem Datum 'A' nach dem Quorum-Consensus-Protokoll an.
- Geben Sie **alle** Schreibmöglichkeiten für eine Transaktion auf dem Datum 'A' nach dem Quorum-Consensus-Protokoll an.
- Zeigen Sie für dieses Beispiel, dass während eine Transaktion T_1 ein Schreibquorum auf A hält es für andere Transaktionen T_x nicht möglich ist ein Lesequorum für A zu bekommen.

Verteilte Datenbanksysteme

Raft Konsensalgorithmus

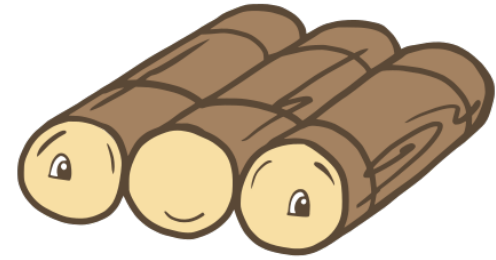


- Server innerhalb eines Clusters können **Follower**, **Candidate** oder **Leader** sein
- Zu Beginn sind alle Knoten Follower
- Es gibt immer nur ein Leader
 - Leader sendet „**Heartbeats**“ zu den Follower-Servern, damit diese wissen, dass der Leader noch existiert und nicht abgestürzt ist
 - Write & Read requests von außerhalb laufen über den Leader
- Jeder Server schreibt Log Einträge („**log replication**“)
 - Leader triggered die Log updates aller Server
- Zur Bestimmung des Leaders wird mit **randomisierten Timeouts** gearbeitet. Die Knoten die zuerst einen Timeout erreichen, werden zu Candidates. Sollte die **Mehrheit** der Server im Cluster für einen bestimmten Candidate stimmen, wird dieser der neue Leader



Verteilte Datenbanksysteme

Raft Konsensalgorithmus

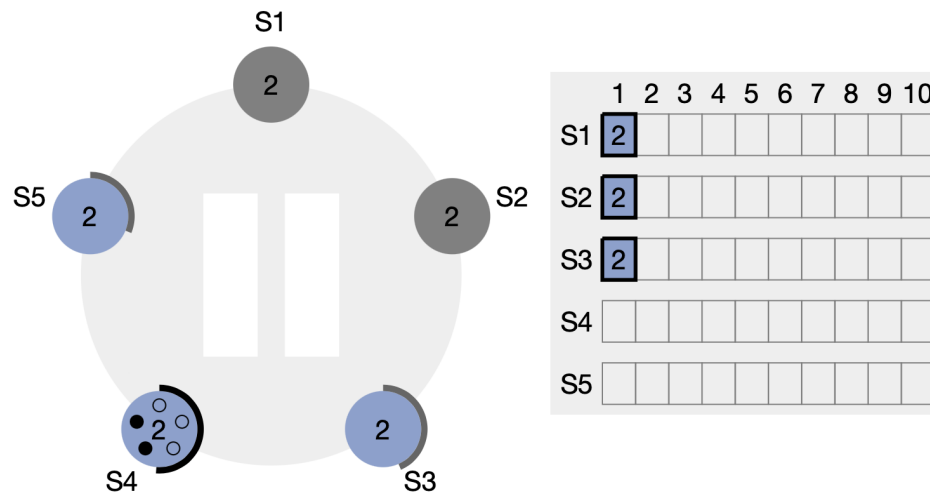


- <http://thesecretlivesofdata.com/raft/>

Aufgabe 3

Beantworten Sie folgende Fragen zum RAFT Protokoll. Verwenden Sie *RaftScope* unter <https://raft.github.io/> um Ihre Vermutungen zu bestätigen.

1. Wie viele Server müssen ausfallen, dass in einem Cluster mit n Servern kein neuer Leader bestimmt werden kann?
2. Wie können Sie mit **restart** und **time-out** in RaftScope einen bestimmten Knoten als neuen Leader erzwingen? Geben Sie die Schritte an.
3. Wie verläuft das Beispiel in Abbildung 1 weiter? Wie kann sicher gestellt werden, dass der neu gewählte Leader den neuesten Logeintrag hält? Beschreiben Sie in Stichpunkten.





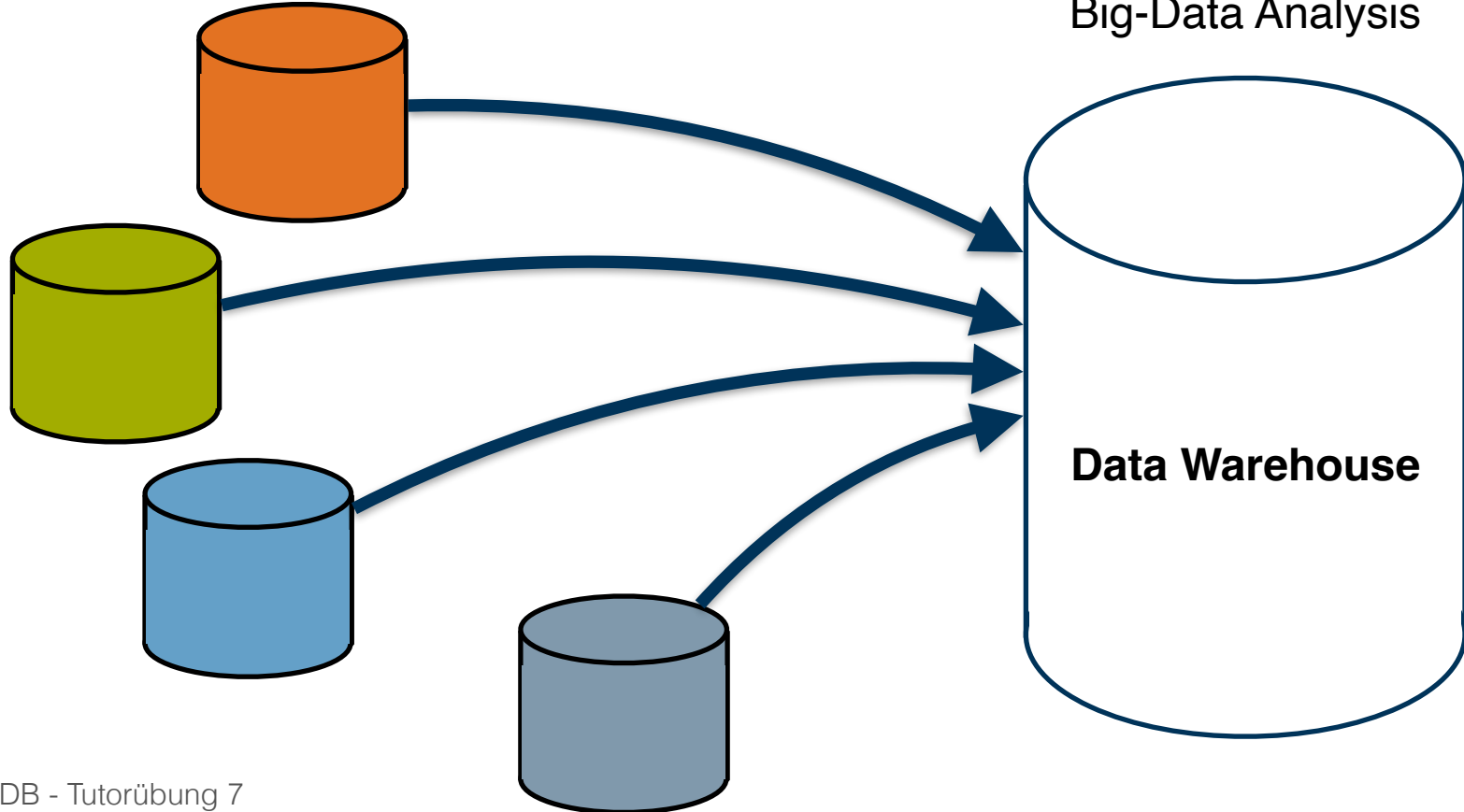
Betriebliche Anwendungen

Betriebliche Anwendungen

Data Warehouses

OLTP-Anfragen & andere
v.a. kleine Transaktionen

OLAP-Anfragen
Big-Data Analysis



Betriebliche Anwendungen

Online Transaction Processing

- Realisiert „operationale“ Tagesgeschäfte („mission-critical“)
- Charakterisierung
 - Hoher Parallelitätsgrad
 - Viele kurze TA (Tausende pro Sekunde)
 - Begrenzte Datenmenge pro TA
 - Operieren auf jüngsten, aktuell gültigen Zustand der DB
 - Hohe Verfügbarkeit muss gewährleistet sein
- Normalisierte Relationen (möglichst geringe Update-Kosten)
- Wenige Indexe (Fortschreibungskosten)

Online Analytical Processing

- Zur strategischen Unternehmensplanung
- Große Datenmengen
- Greift auf historische Daten zu
- ➔ Gewährt Rückschlüsse auf Entwicklungen
- ➔ Bestandteil von Decision-Support-Systeme/Management-Informationssysteme

SQL

Window Functions

- Sehr vielseitig und geeignet für
 - Zeitliche Analysen
 - Rangbasierte Anfragen
 - Top-K
 - Gleitender Durchschnitt
 - Kumulative Summe
- Window Functions werden nach **group by** und vor **order by** ausgewertet

Betriebliche Anwendungen

Window Funktionen

erdb		
name	übung	punkte
Thuy	1	1
Anna	1	1
Domi	1	2
Tobi	2	1
Thuy	2	1
Thuy	3	2
Anna	3	1
Domi	3	1
Thuy	4	3
Tobi	5	2
Domi	5	1
Anna	5	1
Tobi	6	2
Thuy	6	1

```
SELECT name, übung, (100.0*punkte)/
      sum(punkte)
      over (partition by übung) as prozent
FROM erdb
```

Betriebliche Anwendungen

Window Funktionen

erdb		
name	übung	punkte
Thuy	1	1
Anna	1	1
Domi	1	2
Tobi	2	1
Thuy	2	1
Thuy	3	2
Anna	3	1
Domi	3	1
Thuy	4	3
Tobi	5	2
Domi	5	1
Anna	5	1
Tobi	6	2
Thuy	6	1

SELECT name, übung, (100.0*punkte)/
sum(punkte)
over (partition by übung) as prozent
FROM erdb

Ergebnis		
name	übung	prozent
Thuy	1	25.0
Anna	1	25.0
Domi	1	50.0
Tobi	2	50.0
Thuy	2	50.0
Thuy	3	50.0
Anna	3	25.0
Domi	3	25.0
Thuy	4	100.0
Tobi	5	50.0
Domi	5	25.0
Anna	5	25.0
Tobi	6	66.7
Thuy	6	33.3



Betriebliche Anwendungen

Window Funktionen

erdb		
name	übung	punkte
Anna	1	1
Anna	3	1
Anna	5	1
Domi	1	2
Domi	3	1
Domi	5	1
Thuy	1	1
Thuy	2	1
Thuy	3	2
Thuy	4	3
Thuy	6	1
Tobi	2	1
Tobi	5	2
Tobi	6	2

ERDB - Tutorübung 7

```
SELECT name, übung, sum(punkte)
  over ( partition by name
        order by übung)
FROM erdb
```

Ergebnis		
name	übung	sum
Anna	1	1
Anna	3	2
Anna	5	3
Thuy	1	1
Thuy	2	2
Thuy	3	4
Thuy	4	7
Thuy	6	8
Domi	1	2
Domi	3	3
Domi	5	4
Tobi	2	1
Tobi	5	3
Tobi	6	5



Betriebliche Anwendungen

Window Funktionen

erdb		
name	übung	punkte
Anna	1	1
Anna	3	1
Anna	5	1
Domi	1	2
Domi	3	1
Domi	5	1
Thuy	1	1
Thuy	2	1
Thuy	3	2
Thuy	4	3
Thuy	6	1
Tobi	2	1
Tobi	5	2
Tobi	6	2

ERDB - Tutorübung 7

```
SELECT name, übung, sum(punkte)
over ( partition by name
order by übung
range between unbounded
preceding and current row)
FROM erdb
```

Ergebnis		
name	übung	sum
Anna	1	1
Anna	3	2
Anna	5	3
Thuy	1	1
Thuy	2	2
Thuy	3	4
Thuy	4	7
Thuy	6	8
Domi	1	2
Domi	3	3
Domi	5	4
Tobi	2	1
Tobi	5	3
Tobi	6	5



Betriebliche Anwendungen

Window Funktionen

erdb		
name	übung	punkte
Anna	1	1
Anna	3	1
Anna	5	1
Domi	1	2
Domi	3	1
Domi	5	1
Thuy	1	1
Thuy	2	1
Thuy	3	2
Thuy	4	3
Thuy	6	1
Tobi	2	1
Tobi	5	2
Tobi	6	2

ERDB - Tutorübung 7

```
SELECT name, übung, sum(punkte)
over (partition by name
order by übung
range between 1 preceding
and 1 following)
```

FROM erdb

Ergebnis		
name	übung	sum
Anna	1	1
Anna	3	1
Anna	5	1
Thuy	1	2
Thuy	2	4
Thuy	3	6
Thuy	4	5
Thuy	6	1
Domi	1	2
Domi	3	1
Domi	5	1
Tobi	2	1
Tobi	5	4
Tobi	6	4



Betriebliche Anwendungen

Window Funktionen

erdb		
name	übung	punkte
Anna	1	1
Anna	3	1
Anna	5	1
Domi	1	2
Domi	3	1
Domi	5	1
Thuy	1	1
Thuy	2	1
Thuy	3	2
Thuy	4	3
Thuy	6	1
Tobi	2	1
Tobi	5	2
Tobi	6	2

ERDB - Tutorübung 7

```
SELECT name, übung, sum(punkte)
over (partition by name
order by übung
rows between 1 preceding
and 1 following)
```

```
FROM erdb
```

Ergebnis		
name	übung	sum
Anna	1	2
Anna	3	3
Anna	5	2
Thuy	1	2
Thuy	2	4
Thuy	3	6
Thuy	4	6
Thuy	6	4
Domi	1	3
Domi	3	4
Domi	5	2
Tobi	2	3
Tobi	5	5
Tobi	6	4

Betriebliche Anwendungen

Window Funktionen

erdb		
name	übung	punkte
Anna	1	1
Anna	3	1
Anna	5	1
Domi	1	2
Domi	3	1
Domi	5	1
Thuy	1	1
Thuy	2	1
Thuy	3	2
Thuy	4	3
Thuy	6	1
Tobi	2	1
Tobi	5	2
Tobi	6	2

```

SELECT name, sum(punkte) as gesamt
FROM erdb
GROUP BY name
ORDER BY gesamt desc

```

erdb nach group	
name	gesamt
Thuy	8
Tobi	5
Domi	4
Anna	3

Betriebliche Anwendungen

Window Funktionen

erdb nach group	
name	gesamt
Thuy	8
Tobi	5
Domi	4
Anna	3

```

SELECT name, gesamt,
       rank() over (order by gesamt desc)
FROM (
  SELECT name, sum(punkte) as
    gesamt
  FROM erdb
  GROUP BY name desc)

```

Ergebnis		
name	gesamt	rank
Thuy	8	1
Tobi	5	2
Domi	4	3
Anna	3	4

Betriebliche Anwendungen

Window Funktionen

erdb nach group	
name	gesamt
Thuy	8
Tobi	5
Domi	4
Anna	3

```

SELECT name,
  lag(name) over(order by gesamt desc)
  as mehr,
  lead(name) over(order by gesamt desc)
  as weniger
FROM (
  SELECT name, sum(punkte) as gesamt
  FROM erdb
  GROUP BY name desc)

```

Ergebnis		
name	mehr	weniger
Thuy	null	Tobi
Tobi	Thuy	Domi
Domi	Tobi	Anna
Anna	Domi	null

Aufgabe 4

Analysieren wir die Gehälter von Professoren mittels Windowfunctions und führen Sie die Abfragen unter `hyper-db.de` aus. Dazu orientieren wir uns an der Relation *Professoren* des erweiterten Universitätsschemas:

1. Ermitteln Sie zu jedem Professor das Durchschnittsgehalt aller Professoren.
2. Ermitteln Sie zu jedem Professor das Durchschnittsgehalt aller Professoren partitioniert nach Rang.
3. Ermitteln Sie nun die wachsende Summe (das Quantil) des Gehaltes aller Professoren partitioniert nach Rang und absteigend sortiert nach ihrem Gehalt. Gleich verdienende Professoren sind im selben Quartil.
4. Ermitteln Sie nun die wachsende Summe des Gehaltes aller Professoren partitioniert nach Rang und absteigend (total) sortiert nach ihrem Gehalt (reihenweise, nicht als Range-Query).

Aufgabe 4

Analysieren wir die Gehälter von Professoren mittels Windowfunctions und führen Sie die Abfragen unter `hyper-db.de` aus. Dazu orientieren wir uns an der Relation *Professoren* des erweiterten Universitätsschemas:

5. Ermitteln Sie nun das gleitende Durchschnittsgehalt aus genau zwei mehr bzw. weniger verdienenden Professoren sortiert nach Gehalt und partitioniert nach Rang.
6. Ermitteln Sie nun das gleitende Durchschnittsgehalt aus den 500 Einheiten mehr bzw. weniger verdienenden Professoren sortiert nach Gehalt und partitioniert nach Rang. (weder in PostgreSQL noch in HyPer implementiert, fragen Sie Ihren Tutor)
7. Geben sie zu jedem Professor das Gehalt des eins besser wie eins schlechter verdienenden.
8. Ermitteln Sie die drei bestverdienendsten Professoren einmal mit und einmal ohne Windowfunctions.

Aufgabe 5

Betrachten wir das bekannte Uni-Schema mit den Faktentabellen **hoeren** und **pruefen** .

1. Ermitteln Sie in SQL mittels Fensterfunktionen (Windowfunctions) die Top-3 Studenten pro Vorlesung und geben Sie deren Namen aus.
2. Ermitteln Sie mittels SQL-92, um wieviele Notenstufen Studenten, die die Vorlesung gehört haben, in der Prüfung besser abgeschnitten haben.

Aufgabe 6

Betrachten Sie die folgende Tabelle **Waren** mit verkauften Produkten in einem Supermarkt. Die Spalte **verkauft** besagt, wieviele Einheiten des jeweiligen Produktes verkauft worden sind.

Name	Preis	Kategorie	Verkauft
Brot	1.00	Backwaren	8128
Butter	0.80	Kühlwaren	496
Grill	60.00	Haushalt	6
Steak	8.00	Kühlwaren	28
...	...	...	...

- 1 Ermitteln Sie in SQL mittels Fensterfunktionen (Windowfunctions) den prozentualen Umsatzanteil jedes Produktes innerhalb seiner Kategorie.
- 2 Ermitteln Sie in SQL mittels Fensterfunktionen (Windowfunctions) für jedes Produkt das Mittel der Verkaufszahlen aus den 5 besser verkauften (höhere Verkaufszahlen) Produkten geordnet nach Verkaufszahlen.
- 3 Ermitteln Sie in SQL mittels Fensterfunktionen (Windowfunctions) die drei Produkte mit dem meisten Umsatz pro Kategorie.



Fragen?