Seminar 1912, 19912
– Verteilte und parallele Datenbanken –

Thomas Behr

29. August 2017

1 Einleitung

Datenbanken trifft man heutzutage ständig im täglichen Leben. An einigen Stellen ist dies offensichtlich, an anderen arbeiten Datenbanken versteckt im Hintergrund. Durch die fortlaufend wachsenden Herausforderungen an solche Systeme können einzelne Computer diese häufig nicht mehr bewältigen. Daher arbeiten in modernen Systemen häufig mehrere Computer im Verbund. Da die anfallende Arbeit nun auf mehrere Computer aufgeteilt werden kann, ist es möglich, Standard-PCs statt teurer Spezialhardware zu verwenden. Auch das Zuschalten weiterer Komponenten ist bei einer verteilten Datenbanklösung recht einfach. Es gibt zwei Hauptgründe, warum ein einzelner Computer nicht in der Lage ist, die Anforderungen, die an ein modernes Datenbanksystem gestellt werden, zu erfüllen. Einerseits kann die Anzahl an Anfragen an ein solches System sehr hoch sein, andererseits kann es sich um wenige, aber extrem rechenaufwändige Anfragen handeln. Für die verschiedenen Fälle werden unterschiedliche Architekturen benötigt, die charakteristische Eigenschaften besitzen. Im Rahmen des Seminars „Verteilte und parallele Datenbanksysteme“ werden verschiedene Lösungen aufgezeigt, wie die gestellten Aufgaben gemeistert werden können.

2 Themenauswahl

Dieses Dokument beschreibt die verschiedenen Themen des Seminars 1912/19912 im WS 2017/2018. Lesen Sie sich das Dokument gut durch und sehen Sie sich kurz die dazugehörige Literatur an. Erstellen Sie eine Liste mit Ihren Prioritäten. In dieser Liste erhält das Thema, welches Sie

am liebsten bearbeiten möchten, die Priorität 1, wogegen das Thema, das Sie keinesfalls bearbeiten möchten, die Priorität 18 bekommt. Vergeben Sie für **jedes** Thema eine eindeutige Priorität.

Senden Sie diese Liste bis spätestens 02.10.2017 per E-Mail an die Kursbetreuung (thomas.behr@fernuni-hagen.de). Ich werde versuchen, bei der Themenvergabe Ihre Prioritäten weitestgehend zu berücksichtigen. Natürlich kann ich nicht garantieren, dass Sie Ihr Wunschthema erhalten. Sobald alle Listen eingegangen sind, jedoch spätestens nach Ende der angegebenen Frist, werden die Themen auf die Seminarteilnehmer/Seminarteilnehmerinnnen verteilt. Beleger/Belegerinnen des Seminars 19912 (für Master Wirtschaftsinformatik) erhalten zwei Themen zur Bearbeitung. Sollte von Ihnen bis Fristende keine Prioritätenliste vorliegen, wird Ihnen irgendein Thema zugewiesen. Sie bekommen Ihr Thema bis zum 8.10.2017 per E-Mail mitgeteilt.

3 Bearbeitung des Themas

Lesen Sie sich die Themenbeschreibung zusammen mit der dort angegebenen Literatur gut durch. Es wird erwartet, dass Sie selbständig weitere Literatur zu Ihrem gewählten Thema recherchieren. Verwenden Sie dabei nach Möglichkeit wissenschaftliche Publikationen. Achten Sie darauf, dass vor allem der Inhalt der angegebenen Basisliteratur zu beschreiben ist. Weitere Literatur sollte vor allem dann verwendet werden, um bestimmte Aspekte in der Veröffentlichung zu erklären oder um einen Vergleich mit einem anderen Ansatz zu ziehen.

Übersetzen Sie englischsprachige Texte nicht einfach, sondern geben Sie den Inhalt mit eigenen Worten wieder. Vermeiden Sie lange, direkte Zitate. Geben Sie alle verwendeten Quellen an. Beachten Sie auch die Gestaltungshinweise, die Sie im Abschnitt 4 dieses Dokuments finden können.

Senden Sie eine Gliederung Ihrer Ausarbeitung im PDF-Format bis spätestens 12.11.2017 an die Kursbetreuung. Sie erhalten eine Rückmeldung, inwiefern diese Gliederung in Ordnung ist. Sie haben dann bis zum 13.2.2018 Zeit, die vollständige Ausarbeitung sowie Ihre Präsentationsfolien zu erstellen. Senden Sie Ihre Ausarbeitung und Ihre Folien wieder per E-Mail an die Kursbetreuung. Die Ausarbeitung ist im PDF-Format abzugeben. Die Präsentationsfolien können wahlweise als PDF, als PPT (Version 2010) oder als ODP eingereicht werden. Ich werde mir Ihre Dokumente ansehen. Bei Bedarf erhalten Sie **einmalig** Gelegenheit, diese nachzubessern.

Am 9.3.2018 und am 10.3.2018 findet die Präsenzphase des Seminars in Hagen statt. Ein genauer Zeitplan wird noch bekanntgegeben. **Alle** Teilnehmer und Teilnehmerinnen des Seminars sind **an beiden Tagen zu allen Vorträgen** präsent.

Die Vortragsdauer beträgt 45 Minuten. Halten Sie Ihren Vortrag vorab zur Probe, um sicherzustellen, dass Sie diese Zeit einhalten. Verlieren Sie sich während des Vortrags nicht in Einzelheiten und hasten Sie nicht von einem Detail zum nächsten. Versuchen Sie vielmehr in Ruhe die wichtigen Elemente Ihres Themas herauszustellen. An jeden Vortrag schließt sich eine Diskussionsrunde an. Es wird erwartet, dass Sie sich an den Diskussionen aktiv beteiligen. Die Diskussionsrunde dauert pro Vortrag maximal 15 Minuten.

4 Gestaltungshinweise

4.1 Gliederung

Die Gliederung einer Ausarbeitung ist nicht als bloße Auflistung der Abschnitte und Unterabschnitte zu verstehen. Erlären Sie zu jedem Abschnitt und Unterabschnitt stichpunktartig, welches die dazugehörigen Inhalte sein werden. Geben Sie zusätzlich an, wieviel Platz Sie für die einzelnen Abschnitte einplanen.

4.2 Ausarbeitung

- *Keinesfalls* sollen Sie den oder die zugrundeliegende(n) Quelle(n) *übersetzen bzw. wiederholen*. Versuchen Sie vielmehr, den Inhalt mit möglichst einfachen, eigenen Worten zu beschreiben.

- Grundsätzlich ist, zum besseren Verständnis der jeweiligen Thematik, die *Erarbeitung eigener Beispiele* zu empfehlen.

- Häufig ist es nötig, *weitere Literatur* zu berücksichtigen. Ein wichtiger Anhaltspunkt sind die Referenzen, auf die der Basistext verweist. Führen Sie im *Literaturverzeichnis* jedes Werk auf, das Sie zitieren. Dies gilt auch für übernommene Tabellen oder Abbildungen.

- *Bewerten* Sie den vorgestellten Ansatz *kritisch*. Beschreiben Sie die Vorteile, zeigen Sie aber auch die Grenzen und Schwächen auf.

- Sie sollen eine *Ausarbeitung* des Themas erstellen, nicht aber eine Zusammenfassung oder Abschrift/Übersetzung der Literatur oder DIN-A4-Version der Folien.

- Achten Sie auf *korrekte Grammatik und Rechtschreibung*. Alle heutigen Textverarbeitungsprogramme bieten hilfreiche Funktionen zur *Fehlerkorrektur*. Lassen Sie notfalls Ihre Ausarbeitung vor der Abgabe *korrekturlesen*.

- Führen Sie am Schluss eine *komplette Literaturliste* auf. Innerhalb Ihrer Ausarbeitung sollen Verweise auf die Quellen zu finden sein. Achten Sie auf eine einheitliche Darstellung der Literaturangaben.

- Typischerweise haben Seminarausarbeitungen einen Umfang von ungefähr 15 DIN-A4-Seiten bezogen auf die folgenden Rahmenbedingungen:

  - Basis Absatzschriftart: Times-Roman (12 pt)
  - Seitenformat Doppelseitig: Abstand zur Bindekante (2,5 cm), Abstand Außen (2,0 cm), Unten (2,0 cm), Oben (1,5 cm bis zur Kopfzeile und 2,3 cm bis zum Textrahmen).
  - Die Schriftgröße der laufenden Kopfzeile beträgt 9 pt.
  - Der Zeilenabstand beträgt 1,5 Zeilen.

- Geben Sie Ihre Ausarbeitung *ausschließlich im PDF-Format* ab.

- Bitte erwähnen Sie in Ihrer Ausarbeitung *weder Ihre Adresse noch Ihre Matrikelnummer*, da es sich hierbei um sensible Informationen handelt. Vor der Präsenzphase werden wir eine Sammlung aller Ausarbeitungen erstellen, die ggf. auch über die Webseiten des Lehrgebiets abrufbar sein wird.

4.3 Vortrag

- Die *Vortragsdauer* beträgt *45 Minuten*, an die sich eine etwa *15-minütige Diskussionsphase* anschließt.

- Wir erwarten, dass Sie nicht nur bei Ihrem eigenen Vortragsthema mitdiskutieren.

- Versuchen Sie nicht, in Ihrem Vortrag alle Einzelheiten der Basisartikel zusammenzufassen, sondern konzentrieren Sie sich auf die

wichtigsten Ideen und Konzepte. Lassen Sie lieber den einen oder anderen Gesichtspunkt aus und stellen Sie dafür die übrigen Punkte ausführlich und in Ruhe dar, statt von einem Detail zum nächsten zu „hasten“.

- Halten Sie Ihren *Vortrag zur Probe* vor Bekannten (oder auch alleine, aber in jedem Fall laut) und vergewissern Sie sich so, dass Ihr *Zeitplan korrekt* ist und dass Ihre Erklärungen einleuchtend sind.

- Geben Sie zu Beginn Ihres Vortrags eine *kurze Übersicht* über die Inhalte, die Sie erläutern werden. Auch sollten Sie Ihren Vortrag mit einem kurzen *Fazit* über Ihr Seminarthema abschließen.

- Überprüfen Sie nach einzelnen Abschnitten Ihres Vortrags jeweils, ob Sie noch *innerhalb des Zeitplans* liegen. Lassen Sie ggf. weniger wichtige Teile aus, um Ihren Vortrag in der vorgegebenen Zeit abschließen zu können.

4.4 Folien

- Für die bequeme *Erstellung* optisch sehr ansprechender Folien ist die *LaTeX-Beamer-Klasse* empfehlenswert. Sie können auch ein Präsentationsprogramm wie z.B. LibreOffice Impress oder Powerpoint verwenden.

- Achten Sie auf *gute Lesbarkeit*:

  - Benutzen Sie einen genügend großen (16 oder 18 Punkt) und gut lesbaren Zeichensatz.
  - Bedenken Sie auch, dass manche *Farben* von einem Projektor nicht so gut dargestellt werden.

- *Hervorhebungen* erreichen Sie durch Farbwechsel, Kursiv- bzw. Fettschrift und verschiedene Schriftarten. Gestalten Sie Hervorhebungen *einheitlich*. Verwenden Sie so wenige Hervorhebungen wie nötig.

- Die *Titelseite* nennt den Seminartitel, das Thema (Nr. und Titel) und Ihren Namen.

- Eine *Übersicht* (Gliederung Ihres Vortrags) ist auf der zweiten Seite zu finden.

- Nach der Titelseite sind alle Seiten mit einer laufenden *Kopfzeile* (mit Unterstrich), die die Informationen Thema (Nr. und Titel), Ihren Namen und die Seitennummer (Inhaltsverzeichnis auf Seite 1) enthält. Bitte verwenden Sie keine Fußzeile.
- *Vermeiden Sie ganze Sätze* auf Ihren Folien. Geben Sie stattdessen Stichpunkte an und verwenden Sie Skizzen, Diagramme, Bilder oder Formeln, die Sie dann in Ihrem Vortrag erläutern.
- Überladen Sie einzelne Folien nicht mit Informationen. Psychologen haben herausgefunden, dass drei bis fünf Punkte pro Folie optimal sind (allerhöchstens sieben).
- Die Zuhörer und Zuhörerinnen sollen in erster Linie Ihren Ausführungen folgen. Die Folien dienen als *zusätzliche Stütze* für das Publikum.
- Wenn Sie auf jede Folie 3 Minuten verwenden, können Sie in Ihrem Vortrag 15 Folien präsentieren. Planen Sie die Folien und deren Inhalt deshalb vorher genau.

5 Themen

Im Folgenden werden die verschiedenen Themen des Seminars kurz umrissen. Die Nummer des Themas sowie den zu verwendenden Basisartikel finden Sie am Rand.

5.1 Indexe

Thema 1
[VT15]

Werden Daten verteilt abgespeichert, so wird es schwieriger, diese zu indizieren. Insbesondere ist es wichtig, die Indexstruktur ebenfalls verteilt zu speichern, so dass die verschiedenen beteiligten Knoten auch parallel im Index suchen können. In [VT15] wird ein solcher verteilter Index vorgestellt. Auf Basis von HBase, einem Key-Value-Store, werden die Daten in R-Bäumen gehalten, um Anfragen zu beschleunigen.

5.2 Assoziationsregeln

Thema 2
[JSRJ16]

In großen Datenmengen sollen oft Zusammenhänge zwischen verschiedenen Datensätzen automatisch erkannt werden. Dieses Wissen kann z.B.

in der Marktforschung eingesetzt werden, da damit beispielsweise ermittelt werden kann, welche Produkte oft zusammen gekauft werden. Allgemein soll ermittelt werden, welche häufigen Muster in der Datenbank vorkommen. Diese können in einer speziellen Struktur - dem FP-Tree - gehalten werden. In einem verteilten System müssen Daten zwischen den Knoten ausgetauscht werden, um solche häufigen Muster für den vollständigen Datenbestand zu ermitteln. Dies soll so erfolgen, dass die Sicherheit des Systems nicht beeinträchtigt und der Schutz der Daten gewährleistet wird. In [JSRJ16] wird ein Ansatz dafür vorgestellt.

5.3 Partitionierung und Replika-Verwaltung

Thema 3
[DLL+17]

In einem verteilten System wird der Gesamtdatenbestand unterteilt und die einzelnen Partitionen auf die beteiligten Knoten verteilt. Aus Gründen der Ausfallsicherheit und der Vermeidung von Datenübertragungen zwischen den Knoten, werden Kopien der einzelnen Partitionen auf verschiedenen Computern gehalten. Die Partitionierung sowie die Positionierung der Daten und der Kopien beeinflusst die Performance des Systems sehr stark. Offensichtlich hängt eine optimale Strategie von den erwarteten Anfragen an das System ab. In [DLL+17] wird eine Strategie vorgestellt, die eine gute Partitionierung sowie eine gute Allokation der Replikas verfolgt.

5.4 Anfragesprachen

Thema 4
[FLCY14]

In den meisten relationalen Datenbanksystemen ist SQL die verwendete Anfragesprache. Es handelt sich um eine deklarative Sprache, d.h., der Benutzer/die Benutzerin gibt nur an, was das Ergebnis sein soll, nicht, wie dieses berechnet wird. Der Anfrageoptimierer baut aus einer SQL-Anfrage einen Plan, wie das Ergebnis berechnet wird. Natürlich kann es zu einer SQL-Anfrage mehrere solcher Ausführungspläne geben. Welcher tatsächlich verwendet wird, entscheidet allein der Optimierer, der Benutzer/die Benutzerin hat hierauf keinerlei Einfluss. In einem verteilten System erhöht sich die Anzahl der Pläne noch einmal deutlich, da zusätzlich diejenigen Knoten angegeben werden müssen, auf denen die Berechnungen durchgeführt werden. Soll z.B. ein Join zweier Relationen berechnet werden, die auf verschiedenen Knoten liegen, so kann der Join auf dem einem oder dem anderen Knoten stattfinden. Die benötigten Daten müssen dazu vom anderen Knoten übertragen werden. Je nach Vertrauen in die beteiligten Knoten könnte es nun jedoch nicht erwünscht sein, dass Daten auf einen bestimmten Knoten übertragen werden. In

[FLCY14] stellen die Autoren mit PAQO einen Mechanismus vor, mit dessen Hilfe eine SQL-Anfrage angereichert werden kann, um auf die Planerzeugung Einfluss zu nehmen und somit z.B. steuern zu können, welche Knoten verwendet werden dürfen.

5.5 Konsistente Daten

Thema 5
[SYC+16]

Eine wichtige Anforderung an ein Datenbanksystem ist die Konsistenz der bereitgestellten Daten sicherzustellen. So darf es beispielsweise nicht passieren, dass während eines Updates, das die Datensätze A und B betrifft, eine Anfrage sowohl Daten des bereits aktualisierten Datensatzes A und des noch nicht aktualisierten Datensatzs B liest. In verteilten Systemen ist ein solches konsistentes Lesen schwieriger zu gewährleisten, da sich die Datensätze auf verschiedenen Knoten befinden können. Über geeignete Sperrmechanismen lässt sich erreichen, dass ein konsistentes Lesen der Daten erzwungen wird. Dagegen wird durch Sperren die Parallelität des Systems negativ beeinflusst, was längere Antwortzeiten zur Folge hat. In [SYC+16] wird ein Ansatz vorgestellt, mit dem konsistentes Lesen von Daten gewährleistet wird, jedoch lesende Transaktionen schreibende Transaktionen nicht blockieren.

5.6 Transaktionskontrolle

Thema 6
[TDW+14]

In einem klassischen Datenbanksystem werden die ACID-Eigenschaften von Transaktionen garantiert. Verteilt man die Daten auf verschiedene Knoten und verwendet zusätzlich noch Kopien der Daten um die Ausfallsicherheit und die Geschwindigkeit bei Anfragen des Systems zu erhöhen, ist der Aufwand, diese Eigenschaften sicherzustellen, ungleich höher. Daher werden die strengen ACID-Eigenschaften, die es in relationalen Datenbanken gibt, in verteilten Systemen oft aufgeweicht oder nur mit Einschränkungen gewährt. In [TDW+14] stellen die Autoren einen Transaktionsscheduler namens Calvin vor, der es ermöglicht, volle ACID-Eigenschaften von Transaktionen zu erreichen.

5.7 Anfrageanalyse

Thema 7
[MHHH15]

Verteilte Datenbanken stellen komplexe Systeme dar. Stellt man fest, dass Anfragen langsam laufen, wird es schwierig festzustellen, woran dies liegen könnte. Mögliche Ursachen sind ein schlechter Ausführungsplan, hohe Kommunikationskosten oder ein langsam arbeitender Knoten. Da im Regelfall weder der Ausführungsplan sichtbar ist noch angezeigt wird,

wieviele Daten zwischen den Knoten übertragen werden usw., ist es nur durch die Analyse von (hoffentlich vorhandenen) Log-Dateien möglich herauszufinden, was die Ursache der langen Antwortzeiten ist. Eine visuelle Aufbereitung des Plans, der Datenflüsse usw. kann die Problemsuche enorm vereinfachen. In [MHHH15] stellen die Autoren das Tool Perfopticon vor, welches genau diese Aufgabe übernimmt.

5.8 Aktualisierung von Duplikaten

Thema 8
[GOT13]

Um die Verfügbarkeit eines verteilten Systems zu erhöhen und die Antwortzeiten bei Anfragen zu verkürzen, werden Daten mehrfach gespeichert. Während sich replizierte Daten auf Abfragen positiv auswirken, stellen sie bei Updates ein Problem dar. Grundsätzlich gibt es zwei Möglichkeiten, Updates durchzuführen. Entweder sämtliche Kopien eines Datensatzes werden zusammen mit dem Datensatz selbst aktualisiert oder es wird nur der Datensatz aktualisiert und die Aktualisierung der Kopien erfolgt asynchron. Erstere Variante ist kaum von praktischer Bedeutung, da erstens bei Ausfall eines einzigen Knotens, auf dem sich eine Kopie des betroffenen Datensatzes befindet, gar keine Updates mehr durchgeführt werden können und zweitens das Update erst abgeschlossen werden kann, wenn alle Knoten aktualisiert wurden. Durch die gehaltenen Sperren wird dadurch das System verlangsamt. Bei der zweiten Variante gibt es hingegen das Problem, dass sich die Werte, die in einer Tabelle stehen, von den Werten, die in einer Kopie der Tabelle stehen, unterscheiden können. Insbesondere bei hohen Updateraten auf den Daten muss entschieden werden, wann eine Kopie aktualisiert werden sollte. Durch Zusammenfassen mehrerer Updates kann hierbei die Netzwerklast reduziert und die Performance des Gesamtsystems verbessert werden. In [GOT13] wird ein Verfahren vorgestellt, das sich dieses Themas annimmt.

5.9 Datenaustausch

Thema 9
[LYB17]

Da die Daten in einem verteilten System auf unterschiedlichen Knoten liegen, müssen diese zur Beantwortung von Anfragen zwischen den Knoten übertragen werden. Da die einzige Verbindung zwischen den Knoten das dazwischengeschaltete Netzwerk ist, müssen die Daten darüber übertragen werden. Klassischerweise erfolgt die Datenübertragung über das TCP/IP Protokoll. Neuere Hardware ermöglicht ein direktes Schreiben von Daten in den Hauptspeicher eines entfernten Computers über das Netzwerk (Remote Direct Memory Access, RDMA). Diese Art der Übertragung ermöglicht weitaus höhere Übertragungsraten zwischen den

beteiligten Knoten. In [LYB17] stellen die Autoren einen Operator vor, mit dessen Hilfe RDMA zur Datenübertragung in einem parallelen Datenbanksystem genutzt werden kann.

5.10 Anfrageoptimierung

Thema 10
[CJW+16]

Wird an ein Datenbanksystem eine SQL-Anfrage gestellt, so muss diese in einen Ausführungsplan überführt werden, mit dem das Ergebnis der Anfrage berechnet werden kann. Dieser Ausführungsplan sollte so gestaltet sein, dass die Ausführung der Anfrage möglichst schnell erfolgt. In einem verteilten System muss zusätzlich berücksichtigt werden, auf welchen Knoten die Daten liegen und welche Knoten welche Berechnungen durchführen sollten. Im Regelfall sollten umfangreiche Datenübertragungen durch den verteilten Ausführungsplan vermieden werden. Die Erstellung des Plans ist Aufgabe des Anfrageoptimierers. In [CJW+16] beschreiben die Autoren den Anfrageoptimierer des Systems MemSQL. Da das System auch Echtzeitanfragen unterstützen soll, ist es wichtig, dass der Optimierungsprozess ebenfalls sehr schnell erfolgt.

5.11 Protokolle

Thema 11
[LCC+16]

Um die Konsistenz parallel ablaufender Transaktionen zu gewährleisten, wird in den meisten verteilten Systemen für verteilte Transaktionen das sogenannte 2-Phasen-Commit-Protokoll verwendet. Dieses hält Sperren auf Objekte recht lange aufrecht und erfordert relativ viel Kommunikation zwischen den beteiligten Knoten und dem Knoten, der die Transaktion koordiniert. Lokale Transaktionen, also solche, die nur auf Daten eines einzelnen Knotens zugreifen, können wesentlich effizienter durchgeführt werden. In [LCC+16] beschreiben die Autoren einen alternativen Ansatz, bei dem versucht wird, verteilte Transaktionen durch lokal ablaufende Transaktionen zu ersetzen. Dazu ist es notwendig, alle für eine Transaktion erforderlichen Daten an einem Knoten zu haben.

5.12 Deadlocks

Thema 12
[SR11]

Eine Verklemmung tritt in einem Datenbanksystem dann auf, wenn Transaktionen wechselseitig auf die Freigabe von Sperren warten. In einem verteilten System können sich diese Sperren auf verschiedenen Knoten befinden. Um dennoch Deadlocks zu erkennen, müssen Nachrichten zwischen den beteiligten Knoten ausgetauscht werden. In [SR11] wird eine

Deadlockerkennung vorgestellt, die die Komplexität der Kommunikation gegenüber anderen Ansätzen reduziert.

5.13 Graphen

Thema 13
[LBO+15]

Mittels Graphen lassen sich einfach Zusammenhänge zwischen Objekten darstellen. Heutzutage müssen zum Teil riesige Graphen verwaltet werden. Man denke hier nur an die Graphen sozialer Netzwerke, die Millionen von Knoten und pro Knoten viele Kanten enthalten. Solche Graphen können in speziell dafür konzipierten Datenbanken gespeichert werden, die aufgrund der Größe dieser Graphen auch verteilt werden. Viele Graphen unterliegen ständigen Veränderungen, d.h., der Graph ändert seine Struktur im Laufe der Zeit. Hier spricht man auch von dynamischen Graphen. In [LBO+15] wird die G*-Graph-Datenbank vorgestellt, die große dynamische Graphen verwalten kann.

5.14 Heterogene Datenquellen

Thema 14
[KVB+16]

Trotz der Möglichkeit, riesige Datenmengen in verteilten Systemen speichern und verwalten zu können, kann es notwendig sein, Daten aus verschiedenen Datenbanksystemen zusammen abfragen zu können. Die Gründe hierfür sind vielseitig. Zum einen können die Daten nicht öffentlich sein und somit nicht einfach in ein gemeinsames System gespeichert werden. Andererseits ist es möglich, dass bestimmte Daten oder Anfragen ein spezielles Datenbankdesign erfordern. Stehen nun heterogene Systeme zur Verfügung, so ist es wünschenswert, diese auf eine einheitliche Art benutzen zu können. Dies setzt eine gemeinsame Anfragesprache voraus. In [KVB+16] wird eine solche Anfragesprache vorgestellt.

5.15 Verteilte Indexe

Thema 15
[KML17]

Einzelne Datenbanksysteme können Daten, die bestimmte Eigenschaften haben sollen, durch die Verwendung von Indexen sehr schnell finden. Entsprechend kann für verteilte Systeme ein verteilter Index hilfreich sein. In [KML17] wird ein solcher verteilter Index speziell für Positionsdaten vorgestellt.

5.16 Allokation

Thema 16
[TB16]

Die Verteilung der Daten auf die einzelnen Knoten bestimmt wesentlich die Zeit, die für Anfragen und Updates benötigt wird. Die Verteilung

der Daten wird im Regelfall so durchgeführt, dass alle ankommenden Anfragen in einer vernünftigen Zeit beantwortet werden können. Die Komponente, die bestimmt, welcher Teil der Daten auf welchen Knoten gespeichert wird, wird auch Ressourcenmanager genannt. Neben der allgemeinen Forderung, dass Anfragen schnell beantwortet werden sollen, können zusätzliche Bedingungen gelten, die eine andere Art der Zuteilung der Daten auf die Knoten erfordern. Ein Ressourcenmanager, der solche zusätzlichen Bedingungen unterstützt, nennt sich Tempo und wird in [TB16] vorgestellt.

5.17 Bewegte Objekte

Thema 17
[DYCG16]

Durch immer preiswerter werdende GPS-Geräte stehen auch immer mehr Positionsdaten von Fahrzeugen, Schiffen, Tieren oder Fußgängern zur Verfügung. Sollen diese genutzt werden, um beispielsweise den öffentlichen Nahverkehr zu optimieren, ist es notwendig, große Mengen solcher Daten zu sammeln und zu analysieren. In [DYCG16] stellen die Autoren ein verteiltes System vor, das die Speicherung und Analyse solcher Daten erlaubt.

5.18 Partitionierung

Thema 18
[ZBS15]

Die Partitionierung der Daten hat einen großen Einfluss auf die Geschwindigkeit des Systems. Insbesondere haben bei einer ungünstigen Partitionierumg Joins einen enormen Overhead durch die notwendigen Datenübertragungen. In [ZBS15] wird eine Methode für eine automatische horizontale Partitionierung von Daten vorgestellt.

Literatur

[CJW+16] Jack Chen, Samir Jindel, Robert Walzer, Rajkumar Sen, Nika Jimsheleishvilli, and Michael Andrews. The memsql query optimizer: A modern optimizer for real-time analytics in a distributed database. *PVLDB*, 9(13):1401–1412, 2016.

[DLL+17] Liming Dong, Weidong Liu, Renchuan Li, Tiejun Zhang, and Weiguo Zhao. Replica-aware partitioning design in parallel database systems. In Francisco F. Rivera, Tomás F. Pena, and José Carlos Cabaleiro, editors, *Euro-Par 2017: Parallel Processing - 23rd International Conference on Parallel and Distributed Computing, Santiago de Compostela, Spain, August 28 - September 1, 2017, Proceedings*, volume 10417 of *Lecture Notes in Computer Science*, pages 303–316. Springer, 2017.

[DYCG16] Zhiming Ding, Bin Yang, Yuanying Chi, and Limin Guo. Enabling smart transportation systems: A parallel spatio-temporal database approach. *IEEE Trans. Computers*, 65(5):1377–1391, 2016.

[FLCY14] Nicholas L. Farnan, Adam J. Lee, Panos K. Chrysanthis, and Ting Yu. PAQO: preference-aware query optimization for decentralized database systems. In Isabel F. Cruz, Elena Ferrari, Yufei Tao, Elisa Bertino, and Goce Trajcevski, editors, *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 424–435. IEEE Computer Society, 2014.

[GOT13] Javier García-García, Carlos Ordonez, and Predrag T. Tosic. Efficiently repairing and measuring replica consistency in distributed databases. *Distributed and Parallel Databases*, 31(3):377–411, 2013.

[JSRJ16] Yaoan Jin, Chunhua Su, Na Ruan, and Weijia Jia. Privacy-preserving mining of association rules for horizontally distributed databases based on fp-tree. In Feng Bao, Liqun Chen, Robert H. Deng, and Guojun Wang, editors, *Information Security Practice and Experience - 12th International Conference, ISPEC 2016, Zhangjiajie, China, November 16-18, 2016, Proceedings*, volume 10060 of *Lecture Notes in Computer Science*, pages 300–314, 2016.

[KML17] Shashank Kumar, Sanjay Madria, and Mark Linderman. M-grid: a distributed framework for multidimensional indexing and querying of location based data. *Distributed and Parallel Databases*, 35(1):55–81, 2017.

[KVB+16] Boyan Kolev, Patrick Valduriez, Carlyna Bondiombouy, Ricardo Jiménez-Peris, Raquel Pau, and José Pereira. Cloudmdsql: querying heterogeneous cloud data stores with a common language. *Distributed and Parallel Databases*, 34(4):463–503, 2016.

[LBO+15] Alan G. Labouseur, Jeremy Birnbaum, Paul W. Olsen, Sean R. Spillane, Jayadevan Vijayan, Jeong-Hyon Hwang, and Wook-Shin Han. The g* graph database: efficiently managing large distributed dynamic graphs. *Distributed and Parallel Databases*, 33(4):479–514, 2015.

[LCC+16] Qian Lin, Pengfei Chang, Gang Chen, Beng Chin Ooi, Kian-Lee Tan, and Zhengkui Wang. Towards a non-2pc transaction management in distributed database systems. In Fatma Özcan, Georgia Koutrika, and Sam Madden, editors, *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 1659–1674. ACM, 2016.

[LYB17] Feilong Liu, Lingyan Yin, and Spyros Blanas. Design and evaluation of an rdma-aware data shuffling operator for parallel database systems. In Gustavo Alonso, Ricardo Bianchini, and Marko Vukolic, editors, *Proceedings of the Twelfth European Conference on Computer Systems, EuroSys 2017, Belgrade, Serbia, April 23-26, 2017*, pages 48–63. ACM, 2017.

[MHHH15] Dominik Moritz, Daniel Halperin, Bill Howe, and Jeffrey Heer. Perfopticon: Visual query analysis for distributed databases. *Comput. Graph. Forum*, 34(3):71–80, 2015.

[SR11] Selvaraj Srinivasan and Ramasamy Rajaram. A decentralized deadlock detection and resolution algorithm for generalized model in distributed systems. *Distributed and Parallel Databases*, 29(4):261–276, 2011.

[SYC+16] Jie Shao, Boxue Yin, Bujiao Chen, Guangshu Wang, Lin Yang, Jianliang Yan, Jianying Wang, and Weidong Liu. Read consistency in distributed database based on DMVCC. In *23rd IEEE International Conference on High Performance Computing, HiPC 2016, Hyderabad, India, December 19-22, 2016*, pages 142–151. IEEE Computer Society, 2016.

[TB16] Zilong Tan and Shivnath Babu. Tempo: Robust and self-tuning resource management in multi-tenant parallel databases. *PVLDB*, 9(10):720–731, 2016.

[TDW+14] Alexander Thomson, Thaddeus Diamond, Shu-Chun Weng, Kun Ren, Philip Shao, and Daniel J. Abadi. Fast distributed transactions and strongly consistent replication for OLTP database systems. *ACM Trans. Database Syst.*, 39(2):11:1–11:39, 2014.

[VT15] Le Hong Van and Atsuhiro Takasu. An efficient distributed index for geospatial databases. In Qiming Chen, Abdelkader Hameurlain, Farouk Toumani, Roland Wagner, and Hendrik Decker, editors, *Database and Expert Systems Applications - 26th International Conference, DEXA 2015, Valencia, Spain, September 1-4, 2015, Proceedings, Part I*, volume 9261 of *Lecture Notes in Computer Science*, pages 28–42. Springer, 2015.

[ZBS15] Erfan Zamanian, Carsten Binnig, and Abdallah Salama. Locality-aware partitioning in parallel database systems. In Timos K. Sellis, Susan B. Davidson, and Zachary G. Ives, editors, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 17–30. ACM, 2015.