Ausführungspläne und -planoperatoren relationaler Datenbankmanagementsysteme

Christoph Koch

Friedrich-Schiller-Universität Jena
Lehrstuhl für Datenbanken und Informationssysteme
Ernst-Abbe-Platz 2
07743 Jena

Christoph.Koch@uni-jena.de

Katharina Büchse

Friedrich-Schiller-Universität Jena
Lehrstuhl für Datenbanken und Informationssysteme
Ernst-Abbe-Platz 2
07743 Jena

Katharina.Buechse@uni-jena.de

KURZFASSUNG

Ausführungspläne sind ein Ergebnis der Anfrageoptimierung relationaler Datenbankoptimierer. Sie umfassen eine Menge von Ausführungsplanoperatoren und unterscheiden sich je nach Datenbankmanagementsystem auf verschiedene Weise. Dennoch repräsentieren sie auf abstrakter Ebene inhaltlich ähnliche Informationen, sodass es für grundlegende systemübergreifende Analysen oder auch Interoperabilitäten naheliegt einen einheitlichen Standard für Ausführungspläne zu definieren. Der vorliegende Beitrag dient dazu, die Grundlage für einen derartigen Standard zu bilden und stellt für die am Markt dominierenden Datenbankmanagementsysteme Oracle, MySQL, SQL-Server, PostgreSQL, DB2 (LUW und z/OS) ihre Ausführungspläne und -planoperatoren anhand verschiedener Kriterien vergleichend gegenüber.

Kategorien und Themenbeschreibung

Database Performance, Query Processing and Optimization

Allgemeine Bestimmungen

Documentation, Performance, Standardization, Languages

Schlüsselwörter

relationale DBMS, Ausführungsplan, Operator, Vergleich

1. EINLEITUNG

In der Praxis sind Ausführungspläne ein bewährtes Hilfsmittel beim Tuning von SQL-Anfragen in relationalen Datenbankmanagementsystemen (DBMS). Sie werden vom Optimierer berechnet und bewertet, sodass abschließend nur der (vermeintlich) effizienteste Plan zur Bearbeitung einer SQL-Anfrage benutzt wird. Jeder Ausführungsplan repräsentiert eine Menge von miteinander verknüpften Operatoren. Zusätzlich umfasst er in der Regel Informationen zu vom Optimierer für die Abarbeitung geschätzten Kosten hinsichtlich CPU-Last und I/O-Zugriffen.

Auf Basis dieser Daten bewerten DBMS-spezifische Werkzeuge wie beispielsweise der für DB2 for z/OS im InfoSphere Optim Query Workload Tuner verfügbare Access Path Advisor [1] die Qualität von Zugriffspfaden und geben Hinweise zu möglicherweise kritischen Bereichen wie etwa Sortierungen von umfangreichen Zwischenergebnissen. DBMS-übergreifende Werkzeuge zur Ausführungsplananalyse existieren dagegen kaum. Ausnahmen davon beschränken sich auf Werkzeuge wie Toad [2] oder Aqua Data Studio [3]. Diese können zwar DBMS-übergreifend Ausführungspläne verarbeiten, verwenden dazu allerdings je nach DBMS separate Speziallogik, sodass es sich intern ebenfalls um quasi eigene „Werkzeuge“ handelt.

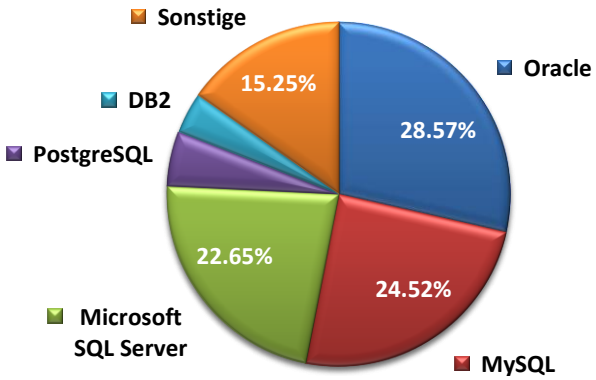

Abbildung 1: Top 5 der relationalen DBMS

Je nach DBMS unterscheiden sich Ausführungspläne und darin befindliche Ausführungsplanoperatoren in verschiedenen Aspekten wie Umfang und Format. Inhaltlich wiederum sind sie sich jedoch recht ähnlich, sodass die Idee einer DBMS-übergreifenden Standardisierung von Ausführungsplänen naheliegt. Der vorliegende Beitrag evaluiert diesen Ansatz. Dazu vergleicht er die Ausführungspläne und Ausführungsplanoperatoren für die Top 5 der in [4] abhängig von ihrer Popularität gelisteten relationalen DBMS (siehe *Abbildung 1*) anhand unterschiedlicher Kriterien. Im Detail werden dabei Oracle 12c R1, MySQL 5.7 (First Release Candidate Version), Microsoft SQL-Server 2014, PostgreSQL 9.3 sowie die bedeutendsten DBMS der DB2-Familie DB2 LUW 10.5 und DB2 z/OS 11 einander gegenübergestellt.

Der weitere Beitrag gliedert sich wie folgt: **Kapitel 2** gibt einen Überblick über die für den Vergleich der Ausführungspläne und Ausführungsplanoperatoren gewählten Kriterien. Darauf aufbauend vergleicht **Kapitel 3** anschließend die Ausführungspläne der ausgewählten DBMS. **Kapitel 4** befasst sich analog dazu mit den Ausführungsplanoperatoren. Abschließend fasst **Kapitel 5** die Ergebnisse zusammen und gibt einen Ausblick auf weitere mögliche Forschungsarbeiten.

27th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 26.05.2015 - 29.05.2015, Magdeburg, Germany.

2. VERGLEICHSKRITERIEN

Ausführungspläne und –planoperatoren lassen sich anhand verschiedener, unterschiedlich DBMS-spezifischer Merkmale miteinander vergleichen. Der folgende Beitrag trennt dabei strikt zwischen allgemeinen Plan-Charakteristiken und Operator-spezifischen Aspekten. Diese sollen nun näher erläutert werden.

2.1 Ausführungspläne

Der Vergleich von Ausführungsplänen konzentriert sich auf externalisierte Ausführungspläne, wie sie vom Anwender über bereitgestellte DBMS-Mechanismen zur Performance-Analyse erstellt werden können. DBMS-intern vorgehaltene Pläne und die Kompilierprozesse von Anfragen, bei denen sie erstellt werden, sollen vernachlässigt werden. Einerseits sind sie als inhaltlich äquivalent anzusehen, andererseits sind ihre internen Speicherstrukturen im Allgemeinen nicht veröffentlicht. Für den Vergleich von Ausführungsplänen wurden im Rahmen des Beitrags folgende Kriterien ausgewählt.

Erzeugung bezieht sich auf den Mechanismus, mit dem durch den Anwender Zugriffpläne berechnet und externalisiert werden können. Dies sind spezielle Anfragen oder Kommandos.

Speicherung ist die Art und Weise, wie und wo externalisierte Zugriffspläne DBMS-seitig persistiert werden. In der Regel handelt es sich dabei um tabellarische Speicherformen.

Ausgabeformate/Werkzeuge erfassen die Möglichkeit zur Ausgabe von Zugriffsplänen in unterschiedlichen Formaten. Dabei wird zwischen rein textueller, XML- und JSON-basierter, sowie visueller und anderweiter Ausgabe differenziert. Unterstützt ein DBMS ein Ausgabeformat, erfolgt zusätzlich die Angabe des dazu zu verwendenden, vom DBMS-Hersteller vorgesehenen Werkzeugs. Drittanbieterlösungen werden nicht berücksichtigt.

Planumfang gibt Aufschluss über im Zugriffsplan verzeichnete generelle Verarbeitungsabläufe. Dahingehend wird verzeichnet, ob Index- und „Materialized Query Table (MQT)“-Pflegeprozesse sowie Prüfungen zur Erfüllung referentieller Integrität (RI) bei Datenmanipulationen im Zugriffplan berücksichtigt werden. Mit MQTs sind materialisierte Anfragetabellen gemeint, die je nach DBMS auch als materialisierte Sichten (MV) oder indexierte Sicht (IV) bezeichnet werden.

Aufwandsabschätzung/Bewertungsmaße bezieht sich auf Metriken, mit denen Aufwandsabschätzungen im Zugriffsplan ausgewiesen werden. Es wird unterschieden zwischen CPU-, I/O- und Gesamtaufwand. Sofern konkrete Bewertungsmaße des DBMS bekannt sind, wie etwa, dass CPU-Aufwand als Anzahl von Prozessorinstruktionen zu verstehen ist, werden diese mit erfasst.

2.2 Ausführungsplanoperatoren

Die Vergleichskriterien für die Ausführungsplanoperatoren gehen über einen reinen Vergleich hinaus. Während einerseits anhand verschiedener Aspekte eine generelle Gegenüberstellung von vorhandenen Operatordetails erfolgt, befasst sich ein überwiegender Teil der späteren Darstellung mit der Einordnung DBMS-spezifischer Operatoren in ein abstrahiertes, allgemeingültiges Raster von Grundoperatoren. Dieses gliedert sich in Zugriffs-, Zwischenverarbeitungs-, Manipulations- sowie Rückgabeoperatoren. Sowohl die Operatordetails, als auch die zuvor genannten Operatorklassen werden nachfolgend näher beschrieben.

Operatorendetails umfassen die Vergleichskriterien Rows (Anzahl an Ergebniszeilen), Bytes (durchschnittliche Größe einer Ergebniszeile), Aufwand (in unterstützten Metriken), Projektionen (Liste der Spalten der Ergebniszeile) und Aliase (eindeutige Objektkennung). Es soll abgebildet werden, welche dieser Details allgemein für Ausführungsplanoperatoren abhängig vom DBMS einsehbar sind. Für den Aufwand wird dabei nochmals unterschieden in absoluten Aufwand des einzelnen Operators, kumulativen Aufwand aller im Plan vorangehenden Operatoren (einschließlich dem aktuellen) und einem Startup-Aufwand. Letzterer bezeichnet den kumulativen Aufwand, der notwendig ist, um den ersten „Treffer“ für einen Operator zu ermitteln.

Zugriffsoperatoren sind Operatoren, über die auf gespeicherte Daten zugegriffen wird. Die Daten können dabei sowohl physisch in Datenbankobjekten wie etwa Tabellen oder Indexen persistiert sein, oder als virtuelle beziehungsweise temporäre Zwischenergebnisse vorliegen. Ähnlich dazu erfolgt die Einteilung der Zugriffsoperatoren in tableAccess-, indexAccess-, generatedRowAccess- und remoteAccess-Operatoren. TableAccess und indexAccess bezeichnen jeweils den Tabellenzugriff und den Indexzugriff auf in den gleichnamigen Objekten vorgehaltenen Daten. Unter generatedRowAccess sind Zugriffe zu verstehen, die im eigentlichen Sinn keine Daten lesen, sondern Datenzeilen erst generieren, beispielsweise um definierte Literale oder spezielle Registerwerte wie etwa „USER“ zu verarbeiten. RemoteAccess repräsentiert Zugriffe, die Daten aus externen Quellen lesen. Bei diesen Quellen handelt es sich in der Regel um gleichartige DBMS-Instanzen auf entfernten Servern. Für weitere spezielle Zugriffsoperatoren, die sich nicht in eine der genannten Kategorien einteilen lassen, verbleibt die Gruppe otherAccess.

Zwischenverarbeitungsoperatoren entsprechen Operatoren, die bereits gelesene Daten weiterverarbeiten. Dazu zählen Verbund- (join), Mengen- (set), Sortier- (sort), Aggregations- (aggregate), Filter- (filter) und Bitmap-Operatoren (bitmap). Spezielle Zwischenverarbeitungsoperatoren, die sich nicht einordnen lassen, werden in einer separaten Kategorie otherIntermediate erfasst.

Manipulationsoperatoren sind Operatoren, die geänderte Daten in physische oder auch temporäre Datenbankobjekte schreiben. Abhängig von in SQL standardisierten Manipulationsanfragen erfolgt die grundsätzliche Einteilung in die Operatoren insert, update, delete und merge. Zusätzlich werden auch remoteManipulation-Operatoren unterschieden, die analog dem bereits betrachteten remoteAccess eine Datenmanipulation auf einem fernen Server durchführen. Analog den vorangehenden Operatorklassen werden spezielle, nicht kategorisierbare Manipulationsoperatoren unter otherManipulation geführt.

Rückgabeoperatoren umfassen Operatoren, die die Wurzel eines (hierarchischen) Ausführungsplans bilden. Oftmals repräsentieren diese nochmals den Typ des SQL-Statements im Sinne einer Unterscheidung in SELECT, INSERT, UPDATE, DELETE und MERGE.

3. AUSFÜHRUNGSPLÄNE IN RDBMS

Ausführungspläne sind grundlegende Elemente in der (relationalen) Datenbanktheorie. Damit sind sie zwar allgemein DBMS-übergreifend von Bedeutung, unterscheiden sich jedoch im Detail voneinander. So existiert beispielsweise für die im Beitrag gewählten Vergleichskriterien und den daran bewerteten Systemen kein Merkmal, in dem sich alle DBMS in ihren Ausführungsplänen gleichen. Abbildung *2* veranschaulicht die Ergebnisse des Vergleichs, auf die gegliedert nach System nun ausführlicher Bezug genommen wird.

DBMS	Erzeugung	Speicherung	Ausgabeformate/Werkzeuge					Planumfang			Aufwandsabschätzung /Bewertungsmaße		
			TEXT	*XML*	*JSON*	*andere*	*grafisch*	*Index-pflege*	*MQT-/MV-/IV-Pflege*	*RI-Prüfung*	*CPU*	*I/O*	*Gesamt*
Oracle (12c R1)	EXPLAIN PLAN	PLAN_TABLE	DBMS_XPLAN. DISPLAY, DBMS_XPLAN. DISPLAY_PLAN	DBMS_XPLAN. DISPLAY_PLAN	-	HTML	SQL Developer, Enterprise Manager	-	-	-	proportional zur Anzahl an Maschinenzyklen	proportional zur Anzahl gelesener Datenblöcke	x
MySQL (5.7)	EXPLAIN	(nur intern)	TRADITIONAL	-	JSON	-	MySQL Work-bench	-	-	-	x	x	x
Microsoft SQL-Server (2014)	(stets mit Ausgabe verknüpft)	(nur intern)	SHOWPLAN_TEXT, STATISTICS PROFILE, SHOWPLAN_ALL (mehr Details)	SHOWPLAN_XML, STATISTICS XML	-	-	Management Studio	x	x	x	x	x	x
PostgreSQL (9.3)	EXPLAIN [ANALYZE| VERBOSE]	(nur intern)	TEXT	XML	JSON	YAML	pgAdmin	-	-	-	-	-	x
DB2 LUW (V10.5)	EXPLAIN	tabellarisch (17 Tabellen)	db2exfmt/ db2expln	-	-	-	Data Studio, OQWT	-	x	x	CPU-Instruktionen	Datenseiten-I/Os	Timerons
DB2 z/OS (V11)	EXPLAIN	tabellarisch (20 Tabellen)	-	-	-	-	Data Studio, OQWT	-	-	-	Millisekunden, Service Units	x	x

Abbildung 2: Vergleich der Ausführungspläne

Oracle [5, 6]: Ausführungspläne werden in Oracle über die SQL-Anfrage EXPLAIN_PLAN **erstellt** und daraufhin innerhalb einer speziellen Tabelle mit der Bezeichnung PLAN_TABLE **gespeichert**. Jede Zeile dieser Tabelle enthält Informationen zu einem Planoperator und bildet deren Kombination im Ausführungsplan über die Spalten ID und PARENT_ID ab. Zusätzlich zur Möglichkeit, per SQL-Anfrage Ausführungspläne aus der PLAN_TABLE auszulesen, unterstützt Oracle die **Planausgabeformate** TEXT, XML und HTML. Diese können jeweils über die im Package DBMS_XPLAN enthaltenen Tabellenfunktionen DISPLAY und deren Erweiterung DISPLAY_PLAN erzeugt werden. Ebenfalls möglich in Oracle ist die grafische Ausgabe von Ausführungsplänen, entweder über das Werkzeug SQL-Developer oder den mächtigeren Enterprise Manager. Hinsichtlich des **Planumfangs** sind Ausführungspläne in Oracle sehr beschränkt. Weder Index- noch MV-Prozesse noch RI-Prüfungen werden im Ausführungsplan als Operatoren ausgewiesen und sind damit nur in einem erhöhten Anfrageverarbeitungsaufwand ersichtlich. **Aufwände** allgemein weist Oracle in allen betrachteten Metriken CPU, I/O und als Gesamtaufwand aus. Während der CPU-Aufwand proportional zu den Maschineninstruktionen und der I/O-Aufwand zur Anzahl gelesener Datenblöcke ausgewiesen werden, existiert für die sich daraus berechneten Gesamtkosten keine Maßeinheit.

MySQL [7]: Ausführungspläne werden in MySQL mittels SQL-Anfrage EXPLAIN **erstellt** und direkt in Tabellenform **ausgegeben**. Eine explizite **Speicherung** der Pläne erfolgt nicht. Seit der Version 5.6 besteht auch die Möglichkeit, sich die Ausführungspläne in MySQL Workbench grafisch darstellen oder sie mit erweiterten Details im JSON-Format ausgeben zu lassen. Bei der Planerstellung wird in MySQL die Gültigkeit der Anfrage nicht überprüft, es können also Pläne für Anfragen erstellt werden, die unzulässig sind (beispielsweise kann beim Einfügen der Datentyp falsch sein). Auch bezüglich des **Planumfangs** zeigt sich MySQL eher rudimentär. Es gibt keine materialisierten Sichten und über Indexpflege oder das Überprüfen referentieller Integrität macht der Plan keinerlei Angaben. Bezüglich der **Aufwandsabschätzung** weist MySQL CPU-, I/O- und den Gesamtaufwand aus.

Microsoft SQL-Server [7]: Im SQL-Server werden Pläne erst bei ihrer Ausgabe externalisiert. Ein Mechanismus zur reinen **Erzeugung** und spezielle Strukturen zur **Speicherung** sind damit nicht nötig und folglich nicht vorhanden. Zur **Ausgabe** von Ausführungsplänen existieren verschiedene SHOWPLAN SET-Optionen. Sobald diese aktiviert sind, werden für nachfolgend ausgeführte SQL-Anfragen automatisch Pläne in den Formaten TEXT oder XML erstellt. Eine grafische Ausgabe bietet das Management Studio. Bezogen auf den **Planumfang** sind Informationen zu sämtlichen betrachteten Prozessen enthalten. Dies gilt analog für die **Aufwandsabschätzung**, für die CPU-, I/O- und Gesamtaufwand ausgewiesen werden. Maßeinheiten sind allerdings nicht bekannt.

PostgreSQL [10]: Ausführungspläne werden in PostgreSQL mittels SQL-Anfrage EXPLAIN **erstellt** und ohne explizite **Speicherung** direkt **ausgegeben**. Soll der Plan dabei zusätzlich ausgeführt werden, muss dies mittels Schlüsselwort ANALYZE proklamiert werden. Über ein weiteres Schlüsselwort VERBOSE lassen sich besonders ausführliche Informationen abrufen. Auch das **Format**, in welchem der Plan ausgegeben werden soll, lässt sich mit entsprechendem Schlüsselwort (FORMAT {TEXT | XML | JSON | YAML}) angeben. Eine grafische Ausgabe bietet das Werkzeug pgAdmin. Zu beachten ist zudem, dass PostgreSQL die Gültigkeit der Anfrage beim Erstellen des Plans (ohne Schlüsselwort ANALYZE) nicht überprüft. Bezüglich des **Planumfangs** macht PostgreSQL keine Angaben über Indexpflege oder die Prüfung referentieller Integrität. Es existieren zwar materialisierte Sichten, diese können aber lediglich manuell über den Befehl REFRESH aktualisiert werden. Für die **Aufwandsabschätzung** liefert PostgreSQL nur „costs“, deren Einheit die Zeit darstellt, die für das Lesen eines 8KB-Blocks benötigt wird.

DB2 LUW [13]: Ausführungspläne werden in DB2 LUW über die SQL-Anfrage EXPLAIN **erstellt** und in einem Set von 20 Explain-Tabellen **gespeichert**. Für die **Ausgabe** der darin abgelegten Informationen besteht neben dem Auslesen per SQL-Anfrage nur die Möglichkeit der textuellen Darstellung mithilfe der mitgelieferten Werkzeuge db2exfmt und db2expln. Über das Data Studio und den Infosphere Optim Query Workload Tuner können Ausführungspläne grafisch ausgegeben werden. In ihrem **Umfang** berücksichtigen Pläne von DB2 LUW zwar die MQT-Pflege und RI-Prüfungen, die Pflege von Indexen jedoch wird nicht dargestellt. Hinsichtlich der **Aufwandsabschätzung** deckt DB2 LUW als einziges der betrachteten Systeme nicht nur alle Metriken ab, sondern liefert zu diesen auch konkrete Bewertungsmaße. CPU-Aufwand wird in CPU-Instruktionen, I/O-Aufwand in Datenseiten-I/Os und der Gesamtaufwand in Timerons angegeben. Da letztere eine IBM-interne Maßeinheit darstellen, entkräftet sich allerdings diese positive Sonderrolle wieder.

DB2 z/OS [16]: In der **Erstellung** und **Speicherung** von Ausführungsplänen ähnelt DB2 z/OS dem zuvor betrachteten DB2 LUW. Es wird ebenfalls die EXPLAIN-Anfrage und eine tabellarische Speicherung verwendet. Die Struktur der verwendeten Tabellen ist jedoch grundverschieden. Neben der Variante, DB2-Zugriffspläne direkt aus der tabellarischen Struktur auszulesen, ist lediglich die grafische **Planausgabe** über separate Werkzeuge wie Data Studio oder den Infosphere Optim Query Workload Tuner möglich. Der **Umfang** des Ausführungsplans in DB2 z/OS ist sehr beschränkt. Es werden keine der betrachteten Prozesse dargestellt. Die **Aufwandsschätzung** dagegen umfasst Angaben zu CPU-, I/O- und Gesamtaufwand. Erstere wird sowohl in CPU-Millisekunden als auch in sogenannten Service Units ausgewiesen.

4. PLANOPERATOREN IN RDBMS

Ähnlich zu den Ausführungsplänen setzt sich die Verschiedenheit der betrachteten DBMS auch in der Beschreibung ihrer in weiten Teilen gleichartig arbeitenden Planoperatoren fort. Diese werden in den folgenden Ausführungen gegenübergestellt. Abschnitt 4.1. vergleicht sie anhand der bereits beschriebenen Kriterien. Im anschließenden Abschnitt 4.2. wird eine Kategorisierung der einzelnen Operatoren nach ihrer Funktionalität beschrieben.

4.1 Operatordetails

Auf Basis der in Abschnitt 2.2 beschriebenen Vergleichskriterien ergibt sich für die Operatordetails der miteinander verglichenen DBMS die in *Abbildung 3* gezeigte Matrix. Auf eine Darstellung der DBMS-spezifischen Bezeichnungen ist darin aus Kompaktheitsgründen verzichtet worden. Die in der Matrix ersichtlichen Auffälligkeiten sollen nun näher ausgeführt werden.

DBMS	Operatorendetails						
	Rows	Bytes	Aufwand absolut	Aufwand kumulativ	Aufwand StartUp	Projektion	Aliase
Oracle (12c R1)	x	x	-	x	-	x	x
MySQL (5.7)	x	-	x	-	x	x	x
Microsoft SQL-Server (2014)	x	x	x	x	-	x	x
PostgreSQL (9.3)	x	x	-	x	x	x	x
DB2 LUW (V10.5)	x	-	-	x	x	x	x
DB2 z/OS (V11)	x	-	-	x	-	-	x

Abbildung 3: Vergleich der Operatorendetails

Eine der wesentlichen Aussagen der Matrix ist, dass ein überwiegender Teil an zentralen Details existiert, die von nahezu allen DBMS für Ausführungsplanoperatoren ausgewiesen werden. Dies sind die Ergebniszeilenanzahl (Rows), der kumulative Aufwand, Projektionslisten und Aliase. Andere Details wie die Größe des Zwischenergebnisses (Bytes), der absolute oder der Startup-Aufwand hingegen stehen nur bei einzelnen DBMS zur Verfügung. Bei horizontaler Betrachtung fällt auf, dass kein DBMS alle Detailinformationen bereithält. Stattdessen schwankt der Detailumfang vom lediglich nicht ausgewiesenen Startup-Aufwand beim SQL-Server oder dem einzig fehlenden absoluten Aufwand bei PostgreSQL bis hin zum DB2 z/OS, das nur die zuvor als zentral betitelten Details mitführt.

4.2 Operatoren-Raster

Planoperatoren sind in den betrachteten DBMS verschieden granular ausgeprägt. Damit gemeint ist die Anzahl von Operatoren, die von knapp 60 im SQL-Server bis hin zu etwa halb so vielen Operatoren in PostgreSQL reicht. Trotzdem decken die einzelnen DBMS, wie in *Abbildung 4* auf Basis einer Matrix dargestellt, mit wenigen Ausnahmen die betrachteten Grundoperatoren funktional mit mindestens einem spezifischen Operator ab. Die weiteren Ausführungen nehmen detaillierter Bezug auf die Matrix. Dabei werden aus Umfangsgründen nur ausgewählte Besonderheiten behandelt, die einer zusätzlichen Erklärung bedürfen.

Oracle [5, 6]: Ausführungsplanoperatoren in Oracle charakterisieren sich durch eine Operation und zugehörige Optionen. Während erstere eine allgemeine Beschreibung zum Operator liefert, beschreiben zweite den konkreten Zweck eines Operators näher. Beispielsweise kann für eine SORT-Operation anhand ihrer Option unterschieden werden, ob es sich um eine reine Sortierung (ORDER BY) oder einen Aggregationsoperator (AGGREGATE) handelt. Allgemein fällt zu den Planoperatoren auf, dass in Oracle viele OLAP-Operatoren wie etwa CUBESCAN oder PIVOT ausgewiesen werden, die sich keinem der Grundoperatoren zuordnen lassen und damit jeweils als otherIntermediate kategorisiert wurden. Besonders ist auch die Tatsache, dass zwar ein REMOTE-Zugriffsoperator, jedoch kein entsprechender Manipulationsoperator existiert. Ein solcher wird jedoch nicht benötigt, da Oracle ändernde Ausführungspläne stets von der DBMS-Instanz ausgehend erstellt, auf der die Manipulation stattfindet, und gleichzeitige Manipulationen auf mehreren Servern nicht möglich sind.

MySQL [7]: MySQL unterstützt lediglich "Left-deep-linear" Pläne, in denen jedes zugegriffene Objekt direkt mit dem gesamten bisherigen Vorergebnis verknüpft wird. Dies wirkt sich auch auf die tabellarische Planstruktur aus, die mit wenigen Ausnahmen wie etwa UNION RESULT je involvierter Tabelle genau eine Zeile enthält. Operatoren werden darin nicht immer explizit verzeichnet, was ihre Kategorisierung sehr erschwert. Die Art des Datenzugriffs ergibt sich im Wesentlichen aus der „type“-Spalte, welche in der MySQL-Dokumentation mit „join type“ näher beschrieben wird. Als eigentlicher Verbund-Operator existiert nur der „nested loop“-Join. Welche Art der Zwischenverarbeitung zum Einsatz kommt, muss den Spalten „Extra“ und „select_type“ entnommen werden. Manipulationsoperatoren werden in Plänen erst seit Version 5.6.3 und dort unter „select_type“ ausgewiesen.

Microsoft SQL-Server [8, 9]: Im SQL-Server kennzeichnen sich Ausführungsplanoperatoren durch einen logischen und einen physischen Operator. Für die abgebildete Matrix sind letztere relevant, da, wie ihr Name bereits suggeriert, sie diejenigen sind, die Aufschluss über die tatsächliche physische Abarbeitung von Anfragen geben. Charakteristisch für den SQL-Server sind die verschiedenen Spool-Operatoren Table, Index, Row Count und Window Spool. All diesen ist gemein, dass sie Zwischenergebnisse in der sogenannten tempDB materialisieren („aufspulen“) und diese im Anschluss wiederum nach bestimmten Werten durchsuchen. Die Spool-Operatoren sind daher sowohl den Manipulations- als auch den Zugriffsoperatoren zugeordnet worden. Einzig der Row Count Spool wurde als otherIntermediate-Operator klassifiziert, da sein Spulen ausschließlich für Existenzprüfungen genutzt wird und er dabei die Anzahl von Zwischenergebniszeilen zählt. SQL-Server besitzt als einziges der betrachteten DBMS dedizierte remoteManipulation-Operatoren, die zur Abarbeitung einer Manipulation bezogen auf ein Objekt in einer fernen Quelle verwendet werden. Zuletzt sollen als Besonderheit die Operatoren Collapse und Split erwähnt werden. Diese treten nur im Zusammenhang mit UPDATE-Operatoren auf und wurden daher ebenfalls in deren Gruppe eingeordnet. Funktional dienen sie dazu, das sogenannte Halloween Problem zu lösen, welches beim Update von aus einem Index gelesenen Daten aufkommen kann [10].

DBMS	Zugriffsoperatoren					Zwischenverarbeitungsoperatoren							Manipulationsoperatoren						Rückgabe-Operatoren
	tableAccess	indexAccess	generatedRow Access	remoteAccess	otherAccess	join	set	sort	aggregate	filter	bitmap	other Intermediate	insert	update	delete	merge	remote Manipulation	other Manipulation	
Oracle (12c R1)	MAT_VIEW ACCESS, MAT_VIEW REWRITE ACCESS, TABLE ACCESS	BITMAP, DOMAIN INDEX, INDEX	-	RE-MOTE	CUBE SCAN, SEQUENCE	CUBE JOIN, HASH JOIN, MERGE JOIN, NESTED LOOPS	CONCATE-NATION, INTERSEC-TION, MINUS, UNION	SORT	CONCATE-NATION, HASH, SORT	COUNT, FILTER, FIRST ROW	BITMAP	AND-EQUAL, CONNECT BY, FOR UPDATE, INLIST ITERATOR, PARTITION, PX (COORDINATOR | PARTITION | RECEIVE | SEND), PIVOT, UNPIVOT, VIEW	INSERT STATEMENT	UPDATE STATE-MENT	DELETE STATE-MENT	-	-	-	(SELECT | INSERT | UPDATE | DELETE) STATE-MENT
MySQL (5.7)	ALL, const, index, Full scan on NULL key, system, Using temporary	const, eq_ref, fulltext, index_sub-query, range, index, ref, ref_or_null, system, unique_sub-query	DERI-VED - no tables used	-	index_merge, Using index condition, Using join buffer (Block Nested Loop | Batched Key Access), Using MRR	nested loop	[DEPEN-DENT | UNCACHE-ABLE] UNION	Using file-sort	Using index for group-by, Using filesort	Range checked for each record, Using where [with pu-shed con-dition]	-	-	INSERT, REPLACE, Using temporary	UPDATE	DELETE	-	-	-	-
Microsoft SQL-Server (2014)	Deleted Scan, Inserted Scan, Parameter Table Scan, RID Lookup, Table Scan, Table Spool, Table-valued Function, Window Spool	Clustered Index Scan, Clustered Index Seek, Index Scan, Index Seek, Index Spool	Con-stant Scan	Remote Index Scan, Remote Index Seek, Remote Query, Remote Scan	Log Row Scan	Hash match, Merge Join, Nested Loops	Concate-nation, Hash Match	Sort	Hash Match, Stream Aggre-gate	Filter, Top	Bitmap	Assert, Compute Scalar, Merge Interval, Parallelism, Row Count Spool, Switch, Segment, Sequence Project, UDX	Batch Hash, Clustered Index Insert, Index Insert, Index Spool, Online Index Insert, Table Build, Table Insert, Table Spool, Win-dow Spool	Clustered Index Collapse, Update, Index Update, Split, Table Update	Clustered Index Delete, Index Delete, Table Delete	Clustered Index Merge, Table Merge	Remote Delete, Remote Insert, Remote Update	Se-quence	SELECT, INSERT, UPDATE, DELETE, MERGE
PostgreSQL (9.3)	Materialize, Seq Scan	Bitmap (Index|Heap) Scan, Index Scan, Index Only Scan, Materialize	Func-tion Scan on gene-rate_-series	Func-tion Scan on dblink	CTE scan, Subquery Scan, Values Scan	Nested Loop, Hash Join, Merge Join	HashAggre-gate, HashSetOp, Append	Sort	Aggregate, HashAggre-gate, Win-dowAgg, GroupAg-gregate, Sort, Unique	Filter, Limit	-	-	Insert, Materialize	Update	Delete	-	Function Scan on dblink	-	-
DB2 LUW (V10.5)	FETCH, TBSCAN	EISCAN, IXSCAN, XISCAN	GEN-ROW	RPD, SHIP	RIDSCN, XSCAN	HSJOIN, MSJOIN, NLJOIN, ZZJOIN	UNION	SORT	GRPBY	FILTER	IXAND, XANDOR	CMPEXP, MGSTREAM, PIPE, REBAL, TQ, UNIQUE	INSERT, TEMP	UPDATE	DELETE	-	-	-	RETURN
DB2 z/OS (V11)	BTBSCAN, CORSUB ACCESS, FETCH, DFETCH, HSSCAN, TBSCAN, WFSCAN	DIXSCAN, FFETCH, FIXSCAN, IXSCAN, RGLIST, SIXSCAN, XIXSCAN	-	-	MIXSCAN	NLJOIN, STARJOIN, MSJOIN, SEMIJOIN, HBJOIN	EXCEPT, EXCEPTA, INTERSECT, INTERSECTA, UNION, UNIONA	SORT, SORT-RID	-	-	IXAND, XIXAND, XIXOR	INLIST, PARTITION, REPARTITION, RID FETCH	INSERT, WKFILE	UPDATE	DELETE, TRUNCATE	MERGE	-	-	QUERY, QB

Abbildung 4: Kategorisierung der Ausführungsplanoperatoren

PostgreSQL [11]: Ausführungspläne in PostgreSQL sind gut strukturiert und bieten zu jedem Operator die für ihn entscheidenden Informationen, sei es der verwendete Schlüssel beim Sortieren oder welcher Index auf welcher Tabelle bei einem Indexscan zum Tragen kam. Für die Sortierung wird zudem angegeben, welche Sortiermethode benutzt wurde (quicksort bei genügend Hauptspeicherplatz, ansonsten external sort). Bei Unterabfragen wird der Plan zusätzlich unterteilt (sub plan, init plan). Soll auf Daten einer anderen (PostgreSQL-) Datenbank zugegriffen werden, so kommt das zusätzliche Modul dblink zum Tragen. Dieses leitet die entsprechende Anfrage einfach an die Zieldatenbank weiter, sodass aus dem Anfrageplan nicht ersichtlich wird, ob es sich um lesenden oder schreibenden Zugriff handelt. Auf einen expliziten Ausgabeoperator wurde in PostgreSQL verzichtet.

DB2 LUW [14]: Analog den zuvor betrachteten DBMS zeigen sich auch für die Planoperatoren von DB2 LUW verschiedene Besonderheiten. Mit XISCAN, XSCAN und XANDOR existieren spezielle Operatoren zur Verarbeitung von XML-Dokumenten. Für den Zugriff auf Objekte in fernen Quellen verfügt DB2 LUW über die Operatoren RFD (nicht relationale Quelle) und SHIP (relationale Quelle). Operatoren zur Manipulationen von Daten in fernen Quellen existieren nicht. Diese Prozesse werden im Ausführungsplan gänzlich vernachlässigt. Die Operatoren UNIQUE (einfach) und MGSTREAM (mehrfach) dienen zur Duplikateleminierung. Da sie dabei aber weder Sortier- noch Aggregationsfunktionalität leisten, wurden sie als otherIntermediate eingeordnet. Dort finden sich auch die Operatoren CMPEXP und PIPE, die lediglich für Debugging-Zecke von Bedeutung sind.

DB2 z/OS [15]: Obwohl DB2 z/OS und DB2 LUW gemeinsam zur DB2-Familie gehören, unterscheiden sich deren Ausführungsplanoperatoren nicht unerheblich voneinander. Lediglich etwa ein Drittel der DB2 LUW Operatoren findet sich namentlich und funktional annähernd identisch auch in DB2 z/OS wieder. Besonders für DB2 z/OS Planoperatoren sind vor allem die insgesamt 5 verschiedenen Join-Operatoren, die vielfältigen Indexzugriffsoperatoren und das Fehlen von Remote-, Aggregations- und Filteroperatoren. Remote-Operatoren sind nicht notwendig, da DB2 z/OS ferne Anfragen nur dann unterstützt, wenn diese ausschließlich Objekte einer DBMS-Instanz (Subsystem) referenzieren. Föderierte Anfragen sind nicht möglich. Aggregationen werden in DB2 z/OS entweder unmittelbar beim Zugriff oder über Materialisierung der Zwischenergebnisse in Form von temporären Workfiles (WKFILE) und anschließenden aggregierenden Workfile-Scans (WFSCAN) realisiert. Ein dedizierter Aggregationsoperator ist damit ebenfalls nicht nötig. Filteroperatoren existieren nicht, weil sämtliche Filterungen direkt in die vorausgehenden Zugriffsoperatoren eingebettet sind.

5. ZUSAMMENFASSUNG UND AUSBLICK

Der Beitrag verglich die Ausführungspläne und –planoperatoren der populärsten relationalen DBMS. Es konnte gezeigt werden, dass der grundlegende Aufbau von Ausführungsplänen sowie den dazu verwendeten Operatoren zu weiten Teilen systemübergreifend sehr ähnlich sind. Auf abstrakterer Ebene ist es sogar möglich, Grundoperatoren zu definieren, die von jedem DBMS in Abhängigkeit seines Funktionsumfangs gleichermaßen unterstützt werden. Der vorliegende Beitrag schlägt diesbezüglich eine Menge von Grundoperatoren und eine passende Kategorisierung der existierenden DBMS-spezifischen Operatoren vor. Darauf basierend ließe sich zukünftig ein Standardformat für Pläne definieren, mit dem die Entwicklung von Werkzeugen zur abstrakten DBMS-unabhängigen Ausführungsplananalyse möglich wäre. Zusätzlich könnte es auch genutzt werden, um föderierte Pläne zwischen unterschiedlichen relationalen DBMS zu berechnen, die über bislang vorhandene abstrakte Remote-Operatoren hinausgehen.

6. LITERATUR

[1] International Business Machines Corporation. *Solution Brief – IBM InfoSphere Optim Query Workload Tuner*, 2014. ftp://ftp.boulder.ibm.com/common/ssi/ecm/en/ims14099usen/IMS14099USEN.PDF

[2] Dell Software Inc. *Toad World*, 2015. https://www.toadworld.com

[3] AquaFold. *Aqua Data Studio – SQL Queries & Analysis Tool*, 2015. http://www.aquafold.com/aquadatastudio/query_analysis_tool.html

[4] DB-Engines. *DB-Engines Ranking von Relational DBMS*, 2015 http://db-engines.com/de/ranking/relational+dbms

[5] Oracle Corporation. *Oracle Database 12c Release 1 (12.1) – Database SQL Tuning Guide*, 2014. https://docs.oracle.com/database/121/TGSQL.pdf

[6] Oracle Corporation. *Oracle Database 12c Release 1 (12.1) – PL/SQL Packages and Types Reference*, 2013. https://docs.oracle.com/database/121/ARPLS.pdf

[7] Oracle Corporation. *MySQL 5.7 Reference Manual*, 2015. http://downloads.mysql.com/docs/refman-5.7-en.a4.pdf

[8] Microsoft Corporation. *SQL Server 2014 – Transact-SQL Reference (Database Engine)*, 2015. https://msdn.microsoft.com/de-de/library/bb510741.aspx

[9] Microsoft Corporation. *SQL Server 2014 – Showplan Logical and Physical Operators Reference*, 2015. https://technet.microsoft.com/de-de/library/ms191158.aspx

[10] PostgreSQL Global Development Group. *PostgreSQL 9.3.6 Documentation – EXPLAIN*, 2015. http://www.postgresql.org/docs/9.3/static/sql-explain.html

[11] EnterpriseDB. *Explaining EXPLAIN*, 2008. https://wiki.postgresql.org/images/4/45/Explaining_EXPLAIN.pdf

[12] F. Amorim. *Complete Showplan Operators*, Simple Talk Publishing, 2014. https://www.simple-talk.com/simplepod/Complete_Showplan_Operators_Fabiano_Amorim (without video).pdf

[13] International Business Machines Corporation. *DB2 10.5 for Linux, UNIX and Windows – Troubleshooting and Tuning Database Performance*, 2015. http://public.dhe.ibm.com/ps/products/db2/info/vr105/pdf/en_US/DB2PerfTuneTroubleshoot-db2d3e1051.pdf

[14] International Business Machines Corporation. *Knowledge Center – DB2 10.5 for Linux, UNIX and Windows – Explain operators*, 2014. http://www-01.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.admin.explain.doc/doc/r0052023.html

[15] International Business Machines Corporation. *Knowledge Center – Nodes for DB2 for z/OS*, 2014. https://www-304.ibm.com/support/knowledgecenter/SS7LB8_4.1.0/com.ibm.datatools.visualexplain.data.doc/topics/znodes.html

[16] International Business Machines Corporation. *DB2 11 for z/OS – SQL Reference*, 2014. http://publib.boulder.ibm.com/epubs/pdf/dsnsqn05.pdf