Notizen zur Diskussion des Papers: Access Path Selection in a Relational Database Management System

von: P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, T. G. Price

Note Taker: Robert Löschinger

amApril 23, 2015

1 What are the 4 steps to handle a SQL query in the IBM System R? Is there a different to common DB systems?

z.Dt.: Was sind die 4 Schritte zum bearbeiten von SQL Anfragen im System R von IBM. Gibt es einen Unterschied zu gängigen DB Systemen ? (Frage von Mario)

Die vier Phasen des Auswertens von Anfragen sind das Parsing, die Optimierung, die Codegenerierung und die Ausführung. Hierbei erstellt der Parser einen Algebraischen Anfragebaum, nach der Optimierung liegt ein Ausführungsplan vor. Bei der Join-Auswertung werden implizit auch logische Optimierungen vorgenommen (Kreuzprodukte in Joins umwandeln, Selektionen möglichst bald ausführen).

Bei heutigen Systemen wird meist kein ausführbarer Code erzeugt. Allerdings ist ein Trend wieder hin zur Codegenerierung erkennbar, was einen interessanten Kreislauf darstellt. Beim System R brauchte das Erzeugen des Code Sekunden, da aber mit wenigen verschiedenen Anfragen gerechnet wurde war die Zeit nach ein paar Ausführungen des Codes bereits wieder eingespart. Beim erzeugten Code handelt es sich um Maschinencode, es wird eine kleine Anzahl an Code Vorlagen verwendet um diesen zu generieren. Da bei PostgreSQL stark verschachtelte Funktionsaufrufe auch eher langsam sind werden diese auch kompiliert. Das geschieht mit LLVM.

Auf die Frage hin ob es noch immer ein Codeerzeugung im Voraus gibt wurde folgende Antwort gefunden. Ja compiled queries, das geht aber vom Client aus nicht vom DBMS.

2 How are relations/tuples stored and accessed in System R?

z.Dt.: Wie werden Relationen/Tupel im System R gespeichert und wie kann man darauf zugreifen ? (Frage von Patrick)

Eine Relation ist eine Sammlung von Tupeln. Tupel werden in System R auf 4 KB Seiten gespeichert, dabei dürfen die Tupel die größe von 4 KB nicht überschreiten. Die Relationen werden auf Segmenten gespeichert, dabei darf eine Relation nicht die Größe eines Segments überschreiten.

An dieser Stelle kam die Frage auf worum es sich bei den Segmenten denn genau handle und wie groß diese Segmente sind. Die Diskusion ergab, dass Segmente wohl nur eine logische Schranke darstellen, wobei Segmente außerdem beim Segmentscan erwähnt werden. Daher entstand nun die Schlussfolgerung, dass Segmente aller Wahrscheinlichkeit nach eine fixe Größe haben, diese in dem Paper aber nicht angegeben wird.

Bei einem Segmentscan wird das ganze Segment durchsucht. Daher ist es wahrscheinlich, dass die Tupel in Segmenten nicht nach Relationen geordnet sind. Das Scannen einer Relation zieht somit einen Scan des gesamten Segments nach sich.

3 Welche Vor- und Nachteile lassen sich aus dem Fakt ableiten, dass der Benutzer keinerlei wissen über etwaige Zugriffspfade hat?

(Frage von Stefan)

Das Benutzer sich nicht um die Zugriffspfade kümmern müssen stellt einen klaren Vorteil dar. Es kann aber sein das Benutzer über Wissen verfügt welches dem Optimierer helfen könnte, dieses Wissen kann in diesem Fall nicht verwendet werden.

Bei heutigen Datenbanksystemen kann man das System zumindest leicht beeinflussen in dem man beispielsweise die Funktion zum Sortieren ausschaltet. Allerdings setzt sich der Optimierer wenn er es als notwendig erachtet auch über solche Entscheidungen hinweg. Dieses Verhalten ist allerdings stark vom jeweiligen System abhängig.

4 Wie lässt sich das im System R beschriebene interne Speichersystem (RSS) charakterisieren? Welche Arten von RSS-Scans gibt es?

(Frage von Stefan)

Bei dieser Frage ist die Runde zu dem Schluss gekommen das der Inhalt dieser Frage bereits in den vorherigen Fragen besprochen wurde.

5 Auf welche Weise bestimmt das vorgestellte System den optimalen Zugriffspfad?

(Frage von Stefan)

Der Optimierer bei System R erhebt Statistiken wie auf Daten zugegriffen werden kann. Diese Arten werden als Zugriffspfade bezeichnet. Die Arten von Zugriffspfaden umfassen Indexscan und Segmentscan. Zugriffspfade beziehen sich daher auf die Physikalische Ebene des Zugriffs. Wenn es sich um Joins handelt wird jede Zugriffsmöglichkeit der ersten und der zweiten Relation abgewogen.

6 What are selectivity factors and why do we need them?

z.Dt.: Was sind Selektivitätsfaktoren und warum sind diese notwendig? (Frage von Patrick)

Der Optimierer von System R hat für jedes Prädikat einen Selektivitätsfaktor. Das ist der Bezug von Eingangs- zu Ausgangstupel. Anschließend wurden verschiedene Arten von Selektionsprädikaten besprochen.

column = value

$$F = \frac{1}{ICARD(column))} \tag{1}$$

Wenn ein Index besteht wird die obige Formel verwendet, andernfalls wird von $F = \frac{1}{10}$ ausgegangen. ICARD gibt an wie viele verschiedene Werte im Index sind. Die obige Formel setzt allerdings eine Gleichverteilung voraus, was bei Realdaten nicht immer zutrifft.

column1 = column2

$$F = \frac{1}{MAX(ICARD(column), ICARD(column2))} \tag{2}$$

$$F = \frac{1}{ICARD(column)} \tag{3}$$

$$F = \frac{1}{10} \tag{4}$$

Wenn beide Spalten einen Index haben gilt 2. Wenn nur eine der beiden Spalten einen Index hat gilt 3. Wenn keine der beiden Spalten einen Index hat, kommt 4 zum tragen. Die Bedingung column1=column2 kommt oft bei Join Attributen vor.

column > value

$$F = \frac{MAX(column) - value}{MAX(column) - min(column)} \tag{5}$$

$$F = \frac{1}{3} \tag{6}$$

Fall 6 wird verwendet wenn die Spalte nicht arithmetisch ist.

An dieser Stelle ergab sich eine Zwischenfrage in der Runde. Patrick stellte die Frage wann und wie die Statistiken aktuell gehalten werden.

Bei System R muss der Benutzer ein Update der Statistiken veranlassen, da sonst das System zu langsam wäre und die Aktualisierung der Statistik einen Flaschenhals darstellen würde. In heutigen Systemen kann die Statistik mit dem Befehl "Analyse" aktualisiert werden. Darüber hinaus werden in heutigen Systemen die 100 häufigsten Werte extra abgespeichert, für die verbleibenden Werte wird von einer Gleichverteilung ausgegangen. Diese Unterscheidung rentiert sich, da 100 Werte unproblematisch zu erfassen sind und bessere Ergebnisse erzielt werden können.

7 How do they determine the joining order of a n-ary join of relations?

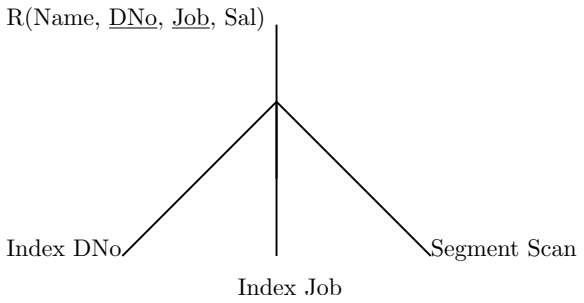
z.Dt.: Wie wird die Join Reihenfolge für multiple Relationen evaluiert? (Frage von Mario)

Zwei Joins werden gleichzeitig verarbeitet, danach wird der nächste Join mit dem Ergebnis des ersten Joins verarbeitet. Verschiedene Reihenfolgen werden in Betracht gezogen um Kreuzprodukte zu vermeiden.

R, S, D
R.ID = S.ID
D.ID = S.ID

(R ⋈ S) ⋈ D
(S ⋈ R) ⋈ D
~~(R ⋈ D) ⋈ S~~
~~(D ⋈ R) ⋈ S~~
(S ⋈ D) ⋈ R
(D ⋈ S) ⋈ R

Karthesische Produkte werden erst als letzte Möglichkeit in betracht gezogen. Daher fallen die beiden ausgestrichenen Möglichkeiten aus der Auswahl. Von den verbleibenden vier Möglichkeiten wird die günstigste ausgewählt.



Joins werden in System R entweder als Nested Loop oder als Merge ausgeführt. Nested Loop ist wie die allseits bekannte Variante ausgeführt. Durch Zusammenspiel mit Scans kann scheint auch ein Index Nested Loop Join möglich. Bei der Join Reihenfolge wird die Heuristik verfolgt, dass die Subsets der möglichst klein sein sollen.

Bei der Auswahl der Join Reihenfolge wird nicht nur der schnellste Teil Baum behalten, sondern auch der Teil Baum der eine "interesting order" aufweist. Eine "interesting order" kann beispielsweise durch ein ORDER BY statement hervorgerufen werden, oder wenn die Ordnung für andere Joins vorteilhaft sein kann. Diese Art der Auswertung nennt man dynamic programming.

8 What is a correlation subquery?

z.Dt.: Was ist eine korrelierende Unteranfrage? (Frage von Patrick)

Korrelierende Unteranfragen können bei Geschachtelten Anfragen auftreten, dabei greifen diese Unteranfragen auf Elemente der eigentlichen Anfrage zu. Das führt dazu, dass bei jeder Auswertung der äußeren Anfrage, die innere auch behandelt werden muss. Das muss in die Kostenabschätzung der Anfrage einfließen.

9 How are Nested Queries handled?

z.Dt.: Wie werden geschachtelte Anfragen bearbeitet? (Frage von Mario)

Der Fall von korrelierenden geschachtelten Anfragen wurde bereits in einer vorherigen Frage behandelt. Im Fall von nicht korrelierenden Unteranfragen kann System R die Unteranfrage einmal auswerten und das Ergebnis für alle Elemente der eigentlichen Anfrage verwenden.

Notizen zur Diskussion des Papers: Access Path Selection in a Relational Database Management System, von Selinger et all

Note Taker: Philipp Neulinger

28. April 2015

1 What are the 4 steps to handle a SQL query in the IBM System R? Is there a different to common DB systems?

z.Dt.: Was sind die 4 Schritte um SQL Anfragen im System R abzuarbeiten? Gibt es einen Unterschied zu heutigen DB Systemen? (Frage von Mario)

- **Parsing**
  Dabei prüft System R, dass das gegebene SQL Statement eine korrekte Syntax hat. Ein **Query Block** wird dabei durch eine SELECT Liste, eine FROM Liste und einen WHERE Baum beschrieben. Der WHERE Baum schränkt dabei die Liste der Ergebnisse ein.
- **Optimization**
  Bei Selinger et al wird hierbei eine Kostenoptimierung hinsichtlich der Plattenzugriffe und der CPU Zeit durchgeführt sowie eine logische Optimierung mittels relationaler Algebra (Selektion nach unten, Projektion nach oben etc). Dabei wählt der Optimizer auch den besten Zugriffspfad aus, das Ergebnis ist dann ein Ausführungsplan der in der Access Specification Language (ASL) vorliegt.
- **Code Generation**
  Nachdem ein solcher Ausführungsplan ausgewählt/erstellt wurde wird der Code Generator gerufen. Dieser erzeugt maschinell lesbare Pfad bzw. Code auf Basis des erstellten Plans und speichert diesen. Der ausführbare Code wird dabei gespeichert, um für spätere Anfragen wiederverwendet zu werden. Da dadurch viel Code entstehen kann, kam man in späterer Folge dazu, dass kein Code mehr produziert wird. PostgresSQL verwendet Funktionen die ineinander verschachtelt werden um den Zugriffsweg zu finden. Speichert dabei also keine Pläne wie System R ab. Das Problem bei der Erzeugung des Codes war, dass dieser Plattformunabhängig sein sollte/musste. Anfangs waren selbige Systeme mehr Compiler als das was heute darunter verstanden wird.
- **Execution**
  Hier wird der erzeugte Code im System R ausgeführt.

2 How are relations/tuples stored and accessed in System R?

z.Dt.: Wie werden Relationen/Tupel gespeichert und wie wird auf sie zugegriffen im System R? (Frage von Mario)

Eine Relation ist in System R eine Sammlung von Tupeln, diese werden dabei in Reihenfolge der Sortierung der Spalten einer Tabelle auf einer Page gespeichert. Tupel können dabei auf einer 4K Page abgelegt werden, ein Tupel muss also kleiner als 4K sein. Diese Pages werden in logischen Units so genannten "Segmenten" organisiert. Ein Segment kann dabei eine oder mehrere Relationen beinhalten. Tupel von zwei oder mehreren Relationen können dabei also auf der selben Seite liegen.

Indizes sind in diesem Fall B-Bäume, wobei die einzelnen Blätter miteinander verbunden sind. Hierbei wird auch der Begriff des Clustered Index eingeführt; dieser ist ein Index, wobei Tupel so in Segment Pages eingefügt werden, dass sie die Index Ordnung berücksichtigen und die Tupel entsprechend deren Ordnung auch physisch abgelegt werden.

Segmente; bestehen aus einer Menge von Pages, es gibt keine Relation die größer als ein Segment ist. Ein Segment ist dabei eine logische Einheit. Leider geht aus der Arbeit nicht heraus, wie groß ein Segment maximal sein kann, ein weiterer Verdacht der während der Diskussion aufkam war, dass Segmente das sind was wir heute als Datenbanken kennen.

Hierbei wurde auch während der Diskussion auf den Unterschied zwischen Row Storage und Columns Storage eingegangen. Des System R sind die Werte der Tubel nebeneinander abgelegt ← System R ist ein Row Storage Model.

3 Welche Vor- und Nachteile lassen sich aus dem Fakt ableiten, dass der Benutzer keinerlei wissen über etwaige Zugriffspfade hat?

(Frage von Stefan)

Vorteile
User muss nichts über die darunter liegende Datenstruktur wissen. Er/Sie muss nur beschreiben welche Daten er/sie gerne haben wollen würde
Trennung der logischen von der physischen Sicht

Heutzutage kann man den Optimizer bestimmte Hinweise geben. Dieser kann diese Hinweise annehmen oder selbige verwerfen. Man kann bspw. in PostgresSQL das Sortieren abschalten (durch das Setzten bestimmter Environment Variablen). Dies kann aber vom Optimizer übergangen werden. Es kann außerdem durch die Angabe von bestimmten Join Wörtern und Kriterien und deren Reihenfolge der Optimiezer beeinflusst werden.

Nachteile
User kann eventuell vorhandenes Wissen für Optimizierungsprozess nicht einbringen.
Trennung der logischen von der physischen Sicht

4 Wie lässt sich das im System R beschriebene interne Speichersystem (RSS) charakterisieren? Welche Arten von RSS-Scans gibt es?

(Frage von Stefan)

- Index Scan
  Dabei wird ein Index von einem System R User angelegt welcher sich über bestimmte Spalten erstreckt. Durch die bestimmten Eigenschaften des Index kann nun bei einem Scan selbigen durch die entsprechenden Kriterien erst ab einem bestimmten Wert begonnen werden (bspw. bei einem WHERE Salary > 5000). Dadurch können bestimmte Pages sogleich ausgeschlossen werden. Wenn eine gesamte Relation nun mittels eines Index durchsucht werden muss, so wird auf jede Page eines Index einmalig zugegriffen ("touched"). Allerdings kann auf eine solche Page mehr als einmal zugegriffen werden - wenn es z.B. zwei oder mehr Tupel beinhaltet welche nicht "nah" zu einander in der Index Ordnung sind.
- Segment Scan
  Alle Tupel einer Relation werden in dem zugehörigen Segment durchsucht um die passende/das passende Tupel zu finden. Dabei wird auf alle nicht leeren Seiten der Relation in dem Segments einmalig zugegriffen.

RSS steht für Research Storage System und ist ein Sub System in System R. Es ist verantwortlich für das physikalische Ablegen der Daten aus Relationen, das Verwalten der Zugriffswege auf Relationen, das Locking von Relationen (in multi-user Umgebungen) und das Logging bzw. das Wiederherstellen von Daten in Relationen.

5 Auf welche Weise bestimmt das vorgestellte System den optimalen Zugriffspfad?

(Frage von Stefan)

Der Optimizer hält Statistiken bereit, auf welche Art und Weise am Besten auf die Tupel zugegriffen werden kann. Die Wege wie auf Tupel zugegriffen werden können wurden in Frage 4 erläutert (Index und Segment Scan). Der Optimizer konstruiert dabei einen Suchbaum. Dieser evaluiert dabei den besten Zugriffsweg.

6 What are selectivity factors and why do we need them?

Was sind Selectivitäts Faktoren und warum braucht man selbige? (Frage von Patrick)

Während der Diskussion wurde diese verschiedenen Selectivity Factors besprochen, ICARD(I) steht in unten angeführten Rechnungen immer für die Anzahl der verschiedenen Werte im Index I.

- **column = value**
  Kann nur angewandt werden, wenn der Index auf die Spalte verwendet wird. Dies funktioniert auch wenn der Index gleich verteilt ist.

```
column = value
         F = 1 / ICARD(column index) if there is an index on column
         This assumes  an even  distribution of tuples  among the  index key
         values.
         F = 1/10    otherwise
```

- **column1 = column2**
  Ist nur relevant für Join Attribute und kann die Selektivität von Joins abschätzen.

```
column1 = column2
         F = 1/MAX(ICARD(column1 index), ICARD(column2 index))
             if there are indexes on both column1 and column2
         This assumes  that each  key value  in the  index with  the smaller
         cardinality has a matching value in the other index.
         F = 1/ICARD(column-i index) if there is only an index on column-i
         F = 1/10    otherwise
```

- **column > value**
  Hierfür muss der Maximale bzw. der Minimale Wert in den Statistiken gespeichert werden.

```
column > value (or any other open-ended comparison)
         F = (high key value - value) / (high key value - low key value)
         Linear interpolation  of the value within  the range of  key values
         yields F if the column is an  arithmetic type and value is known at
         access path selection time.
         F = 1/3 otherwise (i.e. column not arithmetic)
         There is no  significance to this number, other than  the fact that
         it is  less selective  than the  guesses for  equal predicates  for
         which there  are no  indexes, and  that it  is less  than 1/2.  We
         hypothesize that few  queries use predicates that  are satisfied by
         more than half the tuples.
```

Hierbei kam die Frage auf, wie die Statistiken in System R gewartet werden: Hierfür muss ein User des Systems selbständig anfordern dass die Statistiken neu berechnet und abgelegt werden. Würden die Statistiken bei jedem Insert, Update

oder Delete mit gewartet werden, so wäre das ein Bottleneck.

Eine weitere Möglichkeit zum Abschätzen von Selectivity Factors wäre, dass man zu einer Relation die 100 häufigsten Werte mit in die Statistik aufnimmt (natürlich macht das nur Sinn bei Felder die keine Eindeutigkeit innerhalb der Spalte haben, wie Kontostand o.Ä.).

Die Query Zeit darf nicht an die zu nahe an die benötigte Zeit zur Optimierung herankommen.

7 How do they determine the joining order of a n-ary join of relations?

Wie wird die joining Order von n-ary Joins von Relationen erkannt? (Frage von Mario)

Auch n-ary Joins werden im Grunde als Binäre Joins betrachtet, bzw. werden n-ary Joins in binäre Joins aufgespalten. Das soll die Bildung großer Kreuzprodukte verhindern.

Annahme: R, S und D wären Relationen wobei gilt; R.ID = S.ID und S.ID = D.ID

$$
\begin{array}{lllllll}
(& R & \bowtie & S &) & \bowtie & D \\
(& S & \bowtie & R &) & \bowtie & D \\
\cancel{(} & \cancel{R} & \cancel{\bowtie} & \cancel{D} & \cancel{)} & \cancel{\bowtie} & \cancel{S} \\
\cancel{(} & \cancel{D} & \cancel{\bowtie} & \cancel{R} & \cancel{)} & \cancel{\bowtie} & \cancel{S} \\
(& S & \bowtie & D &) & \bowtie & R \\
(& D & \bowtie & S &) & \bowtie & R
\end{array}
$$

Dabei wird versucht die Bildung von kartesischen Produkten so weit als möglich hinauszuzögern. Für die verbleibenden vier erfolgt die Frage, ob ein Index oder ein Segment Scan günstiger wäre. Hierbei wurde das Beispiel der Arbeit besprochen wo die Tabelle EMP(Name, DNO, Job, Salary) gegeben war, wobei die unterstrichenen Felder Spalten sind, für welche Indizes vorhanden sind. Der Segment Scan wird dabei immer aufgeführt, da es oftmals günstiger sein kann, im nach hinein zu sortieren. Des Weiteren wurden noch die verschiedenen Arten von Joins beschrieben:

- Nested Loop Join; zwei ineinander verschachtelte Schleifen wo zwei Werte immer jeweils verglichen werden.
- Merging Join:

Auf Basis der Heuristik und der gegebenen Statistiken durch System R wird versucht möglichst kleine Subsets zu bilden. Hierbei wurde auch der Begriff der

"interesting Orders" im Zusammenhang mit der Auswahl des Besten der 4 Kandidaten angesprochen. Dabei handelt es sich um die verschiedenen *order by*, *group by* und *Join* Kriterien und wie diese angeordnet sind. Hinsichtlich derer Anordnung können Kandidaten ausgeschlossen werden.

Aus den verbleibenden wird mittels Dynamic Programming und Ausprobieren aller Lösungen (natürlich bis auf Kreuzproduktbildungen) die beste Lösung bestimmt.

8 What is a correlation subquery?

Was ist eine "correlation subquery"? (Frage von Patrick)

In der Diskussion wurde auf die folgenden beiden Queries Formen eingegangen:

- **Nested Queries**
  Eine nested Querie sind eine oder mehrere Queries die ineinander verschachtelt wurden. Dazu können die sechs Basis Scalar Funktionen verwendet (=, ¬=, <, >, <=, >=) um einzelne Werte zu erhalten oder mithilfe des IN oder NOT IN Operators können mehr als ein Wert angegeben werden (eine Menge). Dies wäre ein Beispiel für eine solche Nested Querie:

  ```
  SELECT Name
  FROM EMPLOYEE
  WHERE Department_Number in (SELECT Department_Number
                              FROM   DEPARTMENT
                              WHERE  LOCATION='Denver')
  ```

  Dabei muss der Optimizer den gesamten Ausdruck einmalig evaluieren. Dabei muss natürlich dafür gesorgt werden, dass die Subqueries vor der Topquery behandelt wird. Ist die Subquerie durch eine der skalaren Operationen mit eingebunden dann wird die Subquerie mit dem Ergebnis der Operation ersetzt. Ist die Subquerie durch ein IN oder NOT IN eingebunden, so wird das Ergebnis in einer temporären Liste gespeichert. Dabei wird sozusagen die Subquerie durch die temporäre Liste ersetzt.
  Dadurch besteht die Möglichkeit das theoretisch unendlich viele dieser Verschachtelungen auftreten können. Verweist keine Subquerie auf eine höher liegende Subquery bzw. auf die Top Query so werden die Subqueries die am tiefsten verschachtelt sind zuerst aufgelöst.

- **Correlated Subqueries**
  Eine besondere Form der Nested Querie in der die Subqeuries auch auf die Tabelle/Relationen der Top-Query verweisen.

  ```
  SELECT Name
  FROM EMPLOYEE
  WHERE Salary in (SELECT Salary
                   FROM   EMPLOYEE
                   WHERE  Employee_Position = 'Manager')
  ```

Dabei verweist ein ineinander Verschachteltes Subquerie immer auf die Tabelle der darüber liegenden Query. Je nach Art und Weise wie die Query aufgebaut ist, versucht man diese zu dekorrelieren indem man die Subqueries und die Topquery in einen Join umwandelt.

9 How are Nested Queries handled?

Wie werden "nested queries" behandelt? (Frage von Mario)

- Für Uncorrelated Queries:
  Hierbei wird der Wert der tiefer liegenden Query berechnet und mit der darüber liegenden Query evaluiert.
- Für Correlated Queries:
  Hierbei versucht man die correlated Queries in Joins umzuwandeln.

PostgresSQL versucht Optimierung durch dekorellieren korellierter Subqueries und erhält dadurch Blöcke. PostgresSQL versucht das Dekorellieren also ohne die relationale Algebra.