SQL-Ausführungspläne

Michael Lahl (03IND-P), Matthias Jauernig (03INB)
FB IMN / HTWK Leipzig

15.06.2006

Inhaltsverzeichnis

1 Einleitung

Jeder Schreib-/Lesezugriff auf eine Datenbank wird mit der Abfragesprache SQL (Structured Query Language) realisiert. Diese stellt eine bestimmte Anzahl an Manipulations- und Abfragebefehlen für Daten aus relationalen Datenbanken zur Verfügung. Heutzutage findet SQL in jedem Datenbanksystem Anwendung. Um die SQL - Anweisung ausführen zu können spielen viele Komponenten im DBMS zusammen. Der Optimizer ist die Hauptkomponente um eine optimale Abarbeitung zu gewährleisten. Dieser generiert einen speziellen Abarbeitungsplan auch Ausführungsplan genannt. Dies ist notwendig um eine SQL - Anweisung abarbeiten zu können. Vom Optimizer werden mehrer mögliche Ausführunspläne generiert und der Aufwand für jeden einzelnen Plan berechnet. Der billigste Plan wird dann zur Ausführung der Anweisung herangezogen.

2 Der Optimizer

Bei jeder E/A Operation, die auf einer Datenbank vorgenommen werden soll, wird eine SQL - Anweisung ausgeführt. Jedoch kann das DBMS (= Datenbank Management System) den SQL Befehl nicht 1:1 ausführen und wandelt diesen in eine ausführbare form (= Ausführungsplan)

um. Dazu kommt der Optimizer zum Einsatz. Dieser ist eine Funktion des DBMS, mit welcher die als effizentesten geltende Art der Ausführung einer SQL - Anweisung ausgeführt wird. Dieser Schritt ist bei der Ausführung von DML - Anweisungen (= Data Manipulating Language) grundlegend. Es gibt verschiedene Arten der Ausführung einer SQL - Anweisung, z.B. das variiren der Reihenfolge, in der auf Tabellen oder Indizes zugegriffen wird. Dabei hat die Wahl der auszuführenden Reihenfolge eine starke Auswirkung auf die Geschwindigkeit mit der die Anweisung abgeabreitet wird.

Auswertung einer SQL - Anweisung

- *Parser* analysiert Query, prüft syntaktische und semantische Korrektheit
- Berechnung des optimalen Zugriffsplanes durch Optimizer
- *zwei Arten:*
  - **RBO** (rule - based optimizer)
  - **CBO** (cost - based optimizer)
- *row source generator* erhält den vom Optimizer errechneten Zugriffsplan und erstellt auf dieser Grundlage den Ausführungsplan für die physischen Datenbankzugriffe
- *SQL - Execution Engine* erzeugt die Resultatmenge

2.1 Der RBO - Optimizer

Der RBO - Optimizer auch Regelbasierter Optimirer genannt, stellt die ältere Form des Optimizers dar. Dieser erstellt den Zugriffsplan aufgrund eines feststehnden Regelwerkes und aus bestehenden Informationen des Data Disctionary, z.B. Informationen über indizierte Spalten und Cluster Segmenten. Der regelbasierte Optimizer kann keine Statistiken auswerten und hat auch eine beschränkte Anzahl von Operationen zur Verfügung. Wenn es nach der Auswertung der SQL - Anweisung mehrere Möglichkeiten der Ausführung gibt, dann wird die Ausführungsfolge mir dem niedriegsten Rang gewählt, da diese schneller ausgeführt werden können als welche mit höheren Rang.

2.2 Der CBO - Optimizer

Der CBO - Optimizer auch kostenbasierter oder statistikoptimierter Optimizer genannt, ist die besser wahl um SQL - Befehle auszuführen. Standardmässig wird nach dem besten Durchsatz oder der geringsten Ressourcennutzung, die für die Verarbeitung aller einzelnen Zeilen erforderlich sind, optimiert. Oracle kann auch auf eine kürzeste Antwortzeit oder geringste Ressourcennutzung für die erste Zeile optimieren. Grundlegend werden bei dieser Variante des Optimizers Statistiken zur Schätzung der Kosten jedes Ausführungsplanes verwendet. Statistiken beinhalten Datenverteilung und Speichereigenschaften von Tabellen, Spalten, Indizes und Partitionen. Unter Oracle werden mittels des Befehls ANALYZE Statistiken generiert. Die generierten Statistiken werden dann vom Optimizer verwendet um zu schätzen wieviel E/A Operationen, CPU-Zeit und Speicher für die Ausführung eines Ausführungsplanes erforderlich sind.

Vorteile

- besser als regelbasierter Optimizer
- liefert gleiche oder bessere Ergebnisse, besonders bei Joins oder Indizes

Nachteil

- hängt von den Objektstatistiken ab, regelmäßige Aktualisierung erforderlich

3 Was sind Ausführungspläne?

Im Ausführungsplan (engl. Execution Plan) werden die Schritte beschrieben die der Optimizer (meist automatisch durch den DBMS Optimzer oder auch manuell durch den DB Programmierer) wählt, um die vom Benutzer eingegebene SQL Anfrage (engl. SQL Query) auszuführen. Der Ausführungsplan gibt also dem DBMS eine Abarbeitungsfolge für das vom Benutzer eingegebenen SQL Statements vor. Dies ist erforderlich, da die SQL Befehle nicht genau so ausgeführt werden können wie diese vom Benutzer eingegeben werden. Die Laufzeit und Antwortzeit eines SQL Statements wird wesentlich vom Ausführungsplan bestimmt. Eine Interpretation des Ausführungsplans kann bei verschachtelten und größeren Anfragen sehr komplziert sein. Um auch Benutzern, die über kein Wissen dieser interne Abarbeitung verfügen, die Möglichkeit zu geben mit Datenbanken arbeiten zu können, wurde die SQL Sprache in einer für den Benutzer verständlicheren und einfachen Form entwicklt. Diese orientiert sich sehr stark an der englischen Umgangssprache. Daher muss eine Überführung aus dieser nicht prozedualen Form intern in eine prozeduale Form erfolgen. In dieser Form können die Anfragen vom DBMS abgearbeitet werden.

Neben dieser Hauptaufgabe der Ausführungspläne ergeben sich aus ihnen noch andere Möglichkeiten. Sie dienen dem DB-Administrator auch zum optimieren von SQL-Statements. Mittels der Darstellung einer SQL Query in einem Ausführungsplan kann manuell oder auch durch spezielle Programme eine Optimierung dieser Query vorgenommen werden. Eine Interpretation dieser Pläne versetzt den DB-Programmierer (bzw. DB-Administrator) in die Lage, Performanceverbesserungen bei Anfragen zu erreichen.

Einen Ausfürhungsplan kann sich jeder als eine Art Baum vorstellen. Dieser enthält in den Knoten Operatoren zur Verknüpfung von einzelnen abzuarbeitenden Befehlen. Daraus entsteht ein Operatorbaum (bzw. Operatorgraph). Um diesen Operatorbaum abarbeiten zu können müssen noch spezielle Implementierungsvarianten ausgeführt werden. Dazu könnte ein Index-Scan oder ein Nested Loop Join gehören. Aus diesem Aufgebauten Operatorbaum entsteht dann der Ausführungsplan für ein SQL Statement.

Die auszuführenden Schritte werden durch folgende Punkte beeinflusst:

- Syntax der SQL - Anweisung
- Optimizer Modus (regel- oder kostenbasiert)
- verfügbare Indizes/Partitionen

3.1 CPU - Kostenmodell

Jede Datenbankoperation benötigt eine bestimmte Anzahl von CPU-Zyklen. In den meisten Fällen ist es so, dass diese CPU-Zugriffszeiten wesentlich zu den Kosten einer DB Operation (SQL-Statements, Trigger, ...) beitragen. I/O-Zugriffe können oft vernachlässigt werden, da die meisten Operationen(Sortieren, Hashing, ...) sogenannte "IN-Memory Operations" sind. D.h.: sie laufen Vollständig im lokalen Arbeitsspeicher ab und benötigen keine zusätzlichen Plattenzugriffe. Ab Oracle9i unterstützt der Optimierer ein neues Modell, welches die Kosten der CPU-Benutzung mit einbezieht. Dies führte dazu, das die Effektivität des Optimierers gestiegen ist.

Da der Kostenwert(Spalte "COST" der plan_table) zu den wichtigsten Indikatoren für die Effizienz einer SQL - Anweisung zählt, sei hier noch mal die genaue Berechnungsvorschrift aufgeführt:

COST = (#SRds * sreadtime + #MRds * mreadtime + #CPUCycles / cpuspeed) / sreadtime

- #SRds = Anzahl der "single block reads"
- #MRds = Anzahl der "multi block reads"
- #CPUCycles = Anzahl der CPU-Zyklen
- sreadtime = benötigte Zeit für ein "single block read"
- mreadtime = benötigte Zeit für ein "multi block read"
- cpuspeed = CPU-Zyklen je Sekunde

CPU - Cycles setzt sich zusammen aus CPU-Zeit für die Ausführung des SQL-Statements und den CPU-Kosten für das Daten-Retrieval (Kosten für Pufferoperationen). Dieses Kostenmodell ist lediglich für die serielle Ausführung geeignet und benötigt verschiedene Anpassung an den Parametern #SRds, #MRds und #CPUCycles.

3.2 Theorie der Ausführungspläne

Bei der Abarbeitung der logischen und physischen Optimierung werden im allgemeinen alternative (aber äquivalente) Ausführungspläne generiert. Die gernerierten Pläne liefern identische Resultatmengen, können aber unterschiedlich hohe Kosten verursachen. Das DBMS muss daher kostengünstigsten Plan zu Abarbeitung verwenden.

Vorgehensweise des DBMS:

1. Aufstellen des gesamten Suchraumes. Dieser enthält alle möglichen Ausführungspläne für ein spezielles SQL - Statement (Generierung aller äquivalenten Ausführungspläne -¿ Plan - Enumeration)
2. Schätzung der Kosten für jeden im Suchraum vorhandenen Ausführungsplan und Auswahl des kostenoptimalen Ausführungsplanes

Durch die Kombination der beiden genannten Teilschritte werden bereits generierte Teilpläne miteinander verglichen und dabei durch fortlaufende Kostenabschätzung die weitere Planerzeugung oder Vervollständigung der Teilpläne gesteuert, um die Plan-Enumeration zu verkürzen. Um den kostengünstigsten Plan zu erhalten, müssen beim Aufspannen des Suchraumes alle äquivalenten Pläne berücksichtigt werden, um dann aus diesen Plänen den optimalsten auswählen zu können. Dazu sollte das DBMS eventuell vorhandene Informationen (Indexe, Statistiken) zur Auswertung heranziehen, um so die Plangernerierung auf sinnvolle Varianten zu beschränken. Besonders bei Joins entsteht eine große Anzahl an unterschiedlichen Varianten.

Beschränkung des Suchraumes durch:

- Heuristiken, wie algebraische Regeln, z.B. zur Überführung von Selektionen und Kreuzprodukten in Verbunde
- Vorgabe einer Baumform

Betrachten wir nun nur die Verbunde einer Anfrage, ergeben sich zwei Grundformen:

- lineare Folgen von Verbunden, d.h. links-orientierte bzw. rechts-orientierte Bäume
- "buschige" Verbundbäume, die lineare Folgen mit einschließen

Lineare Verbundbäume sind dadurch charakterisiert, dass alle inneren Knoten mindestens einen Blattknoten (d.h. den Zugriff auf eine Basisrelation) als Kind besitzen (ermöglichen ein einfaches Pipelining und Reduktion der Verbundvarianten). **Buschige Bäume** hingegen besitzen ein höheres Parallelisierungspotential (unabhängige/parallele Berechnung von Verbunden). Der Nachteil ist eine größere Anzahl zu berücksichtigender Varianten von Plänen. Die Anzahl der verschiedenen Verbundbäume für eine links-orientierte Folge entspricht der Permutation der Blätter (also der Relationen). Demzufolge ergeben sich für einen Mehrwegeverbund zwischen n Relationen allein n! verschiedene links-orientierte bzw. genauso viele rechtsorientierte Bäume.

Anzahl der verschiedenen Formen von buschigen Bäumen S(n) :

$$S(1) = 1; S(n) = \sum (S(i)\, S(n-i))$$

Nach aufstellen aller Permutationen von Plänen muss nun der gesamte Suchraum durlaufen werden, um den kostengünstigsten Plan zu finden. Hierzu wird eine Strategie benötigt, die festlegt in welcher Reihenfolge die unterschiedlichen Varianten durchlaufen werden müssen und ob eine erschöpfende Suche (alle Pläne werden betrachtet) oder nur ein partielle Suche (nur bestimmte Pläne betrachten) durchgeführt werden sollte. Für diese Suche gibt es zwei grundlegende Varianten. Zum einen die deterministische und zum anderen die randomisierte Suchstrategie. **Deterministische** Suchstrategien verfolgen ein systematisches Generieren aller äquivalenten Pläne kombiniert mit einer erschöpfenden Suche (Vertreter sind Greedy- und dynamische Programmierungstechniken). Dabei werden komplexere Pläne durch Verbund von einfachen Plänen (Zugriff auf Basisrelationen) erzeugt. Suchraum kann in diesem Fall dann durch Tiefen- oder Breitensuche durchlaufen werden. Bei den **randomisierten** Verfahren wird eine vollständige Aufspannung des Suchraums vermieden. Dies wird durch die schrittweise Optimierung einzelner Startpläne realisiert. Hier erfolgt eine Untersuchung der "Nachbar"-Pläne. Diese wird durch Anwendung von Transformationsregeln erreicht (z.B. Simulated Annealing, Hill Climbing, genetische Algorithmen). Obwohl die randomisierten Verfahren keine Garantie für das Finden eines kostenoptimalen Plans aufweisen, ergaben experimentelle Untersuchungen eine besser Performance bei Joins im gegensatz zu denn deterministischen Verfahren.

4 Ausführungspläne in Oracle

In Oracle stehen beide oben dargestellten Modi des Optimizers zur Verfügung. Es gibt mehrer Möglichkeiten in Oracle die Ausführungspläne einzelner SQL - Anweisungen zu erzeugen. Mittels Oracle SQL*Plus kann ein Ausführungsplan mit zwei verschiedenen Varianten erzeugt werden.

1. Generierung eines Ausführungsplanes mit dem Befehl EXPLAIN PLAN (ohne Ausführung)

   Das EXPLAIN PLAN Kommando ermöglicht dem Benutzer , sich das Ergebnis einer vorgenommenen Optimierung direkt anzeigen zu lassen. Bevor der Befehl angewendet werden kann muss die Table plan_table angelegt werden. Dies kann mit dem vorgefertigten Script *utlxplan.sql* erledigt werden.

```
SQL> EXPLAIN PLAN
  2  SET statement_id = 'p1'
  3  FOR
  4  SELECT m.name, m.vorname, t.name, a.name
  5  FROM mitarbeiter m, taetigkeit t, abteilung a
  6  WHERE t.taetigkeit_id = m.f_taetigkeit_id AND a.abt_id = m.f_abt_id;

EXPLAIN PLAN ausgeführt.
```

Abbildung 1: Explain Plan generieren

   Nach der Ausführung der oben stehenden Anweisung werden die Informationen in der *plan_table* gespeichert. Die Anweisung wird aber nicht ausgeführt. Es werden nur die Ausführungsinformationen errechnet und gespeichert.

   Jetzt kann man sich das Ergebnis in der Table *plan_table* anschauen (wobei die gewählte *statement_id* angegeben werden sollte). Das Ergebnis weißt jedoch keine ordentlich Formatierung für den Endanwender auf und ist daher schwer lesbar. Um die Informationen in einer lesbaren Form aus der Tabelle zu extrahieren gibt es eine passende SQL - Anweisung. Dieser nimmt eine Formatierung mittels einer hierarchisch-rekursiven Struktur und Einrückungen vor.

```
SQL> SELECT lpad(' ',2*level)
  2  ||operation||' '||options||' '||object_name q_plan
  3  FROM plan_table
  4  WHERE statement_id = 'p1'
  5  CONNECT BY PRIOR id = parent_id AND statement_id = 'p1'
  6  START WITH id = 1;

Q_PLAN
--------------------------------------------------------------------------------
    HASH JOIN
      TABLE ACCESS FULL ABTEILUNG
      TABLE ACCESS FULL MITARBEITER
    TABLE ACCESS FULL TAETIGKEIT
    HASH JOIN
      TABLE ACCESS FULL ABTEILUNG
      TABLE ACCESS FULL MITARBEITER
    TABLE ACCESS FULL TAETIGKEIT
  HASH JOIN
    HASH JOIN
      TABLE ACCESS FULL ABTEILUNG

Q_PLAN
--------------------------------------------------------------------------------
      TABLE ACCESS FULL MITARBEITER
    TABLE ACCESS FULL TAETIGKEIT

13 Zeilen ausgewählt.
```

Abbildung 2: Informationen aus der *plan_table* extrahieren

2. Ausgabe des Ausführungsplanes mit SET AUTOTRACE Funktionalität

   Hier wird automatisch nach der Eingabe der SQL - Anweisung der Ausführungsplan mit an die Resultatmenge angehangen. Um diese Funktionalität nutzen zu können muss diese aktiviert werden. Das erledigt ein einfacher Befehl, der in der nächsten Grafik dargestellt ist.

```
SQL> SET AUTOTRACE on;
```

Abbildung 3: autotrace Funktionalität aktivieren

Wenn jetzt eine SQL - Anweisung ausgeführt wird, hängt das DBMS an die Ausgabe der Resultatmenge den Ausführungsplan und eine kurze Statistik an. In der nächsten Abbildung ist dies für eine einfache Anfrage dargestellt.

```
SQL> SELECT m.name, m.vorname, t.name, a.name
  2  FROM mitarbeiter m, taetigkeit t, abteilung a
  3  WHERE t.taetigkeit_id = m.f_taetigkeit_id AND a.abt_id = m.f_abt_id;

NAME                 VORNAME              NAME                      NAME
-------------------- -------------------- ------------------------- ------------------
Fischer              Matthias             Programmierer             Technik
Müller               Klaus                Programmierer             Technik
Groß                 Herbert              Geschäftsführer           Geschäftsführung
Schulz               Michael              Technischer Leiter        Technik
Lehmann              Anne                 Sekretärin                Verwaltung
Hoffmann             Anke                 Sekretärin                Verwaltung
Eichler              Bianka               Sekretärin                Verwaltung
Schubert             Toraten              Manager                   Management
Heilmann             Rene                 Manager                   Management
Lehmann              Maria                Manager                   Management
Schulze              Klaus                Manager                   Management

NAME                 VORNAME              NAME                      NAME
-------------------- -------------------- ------------------------- ------------------
Groß                 Tanja                Stellv. Geschäftsführer   Geschäftsführung
Thänert              Christian            Sachbearbeiter            Vertrieb
Heinrich             Kristin              Sachbearbeiter            Vertrieb
Laube                Eric                 Sachbearbeiter            Vertrieb
Grobe                Norbert              Sachbearbeiter            Vertrieb
Meier                Michael              Techniker                 Technik
Maier                Carsten              Techniker                 Technik
Schober              Martin               Techniker                 Technik

19 Zeilen ausgewählt.
```

Abbildung 4: Ausführung einer SQL - Anweisung mit *autotrace*

Zuerst wird die Anweisung ausgeführt und die Resultatmenge ausgegeben.

```
Ausführungsplan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=8 Card=19 Bytes=874)
   1    0   HASH JOIN (Cost=8 Card=19 Bytes=874)
   2    1     HASH JOIN (Cost=5 Card=19 Bytes=570)
   3    2       TABLE ACCESS (FULL) OF 'ABTEILUNG' (Cost=2 Card=5 Byte
          s=65)

   4    2       TABLE ACCESS (FULL) OF 'MITARBEITER' (Cost=2 Card=19 B
          ytes=323)

   5    1     TABLE ACCESS (FULL) OF 'TAETIGKEIT' (Cost=2 Card=8 Bytes
          =128)




Statistiken
----------------------------------------------------------
          0  recursive calls
          0  db block gets
         23  consistent gets
          0  physical reads
          0  redo size
       1282  bytes sent via SQL*Net to client
        510  bytes received via SQL*Net from client
          3  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         19  rows processed
```

Abbildung 5: Ausführung einer SQL - Anweisung mit *autotrace*

Nach der Ausgabe der Resultatmenge wird jetzt der zur Ausführung benutzte Ausführungsplan ausgegeben und eine kurze Statistik. Wem die primitive Textdarstellung des SQL*PLUS nicht ansprechend genug ist, kann sich die Ausführungspläne auch mit einem grafischen Tool anzeigen lassen. Dazu ist bei Oracle ein Firmeneigenes Tool im Enterprise Paket enthalten (siehe Abschnitt Ausführungspläne mit SQL - Analyze).

5 Ausführungspläne mit SQL - Analyze

Oracle stellt in der Enterprise Version ein eigenes grafisches Werkzeug zur Darstellung von Ausführungsplänen zur Verfügung. Mit dem Tool SQL - Analyze können kritsche SQL - Anweisung unter die Lupe genommen werden. Hierzu stehen dem Anwender vorgefertigte Wizard und Assistenten zur Verfügung. Damit können zu jeder SQL - Anweisung detaillerte Tunninganweisungen beschafft werden.

5.1 Erster Start und Konfiguration

Um SQL - Analyze benutzen zu können müssen vorher spezielle Einstellungen vom Datenbank - Administrator getroffen werden. Gestartet werden kann das Tool aus dem Startmenü oder direkt aus der Enterprise Management Console.

Der entsprechende Anwender benötigt spezielle Rechte um SQL - Analyze benutzen zu können. Diese Rechte sind generell in der Rolle des DBA enthalten. Um einen Benutzer nur für SQL - Analyze zu erstellen sind jedoch nur folgende Rechte notwendig:

- CREATE TABLE
- SELECT ANY TABLE
- ANALYZE ANY

Für diese Rechte gibt es auch ein Skript, welches eine Rolle mit diesen Rechten anlegt. Zu finden ist dies im Ordner: ORA_HOME/SYSMAN/ADMIN/vmqrole.sql. Nachdem einem Nutzer diese Rolle zugewiesen ist kann dieser sich mit seinen Login Daten bei SQL - Analyze anmelden. Beim ersten Start wird ein Repository, in welches alle Informationen gespeichert werden können, angelegt. Bei dieser Gelegenheit wollen wir auf ein großes Problem in Oracle9i hinweisen. Jeder angelegt Benutzer konnte sich nur einmal bei SQL - Analyze anmelden, beim zweiten Versuch brach das Tool mit einem schweren Ausnahmefehler zusammen. Es kam auch vor, dass der gesamte Prozess oracle beendet wurde.

5.2 Eigenschaften von SQL - Analyze

- TopSQL Funktionalität (suche nach SQL - Anweisung, welche die meisten Ressourcen benötigen)
- Zugriff auf eine SQL - Historie, in der Informationen über vergangene Anweisungen gespeichert wurden

- Ausführen von SQL - Anweisungen mit unterschiedlichen Optimierungen und gibt dazu Ausführungspläne und Statistiken an
- grafische Anzeige von Ausführungen und Erklärungen zu den einzelnen Schritten
- Überprüfung von SQL - Anweisungen auf syntaktische Korrektheit, zeigen von Möglichen Fehlerquellen
- Präsentiert relevante Objekteigenschaften um Probleme zu erkennen und zu beheben um die Performance der Anweisung zu beeinflussen
- Zugang zu Initialisierungsparametern, haben direkten Einfluss auf die Performance
- Hinzufügen von Optimizer - Hints mit Hilfe des Hint Wizards
- Speicherung von Ausführungsplänen, Statistiken von SQL - Anweisungen im Repository
- Vorschläge zu Indexnutzung um die Performance zu erhöhen
- Virtuelle Indexerstellung (testen von Indexen ohne diese zu erstellen)

5.3 TopSQL

Nach einem Erfolgreichen Start von SQL - Analyze zeigt sich folgendes Fenster.

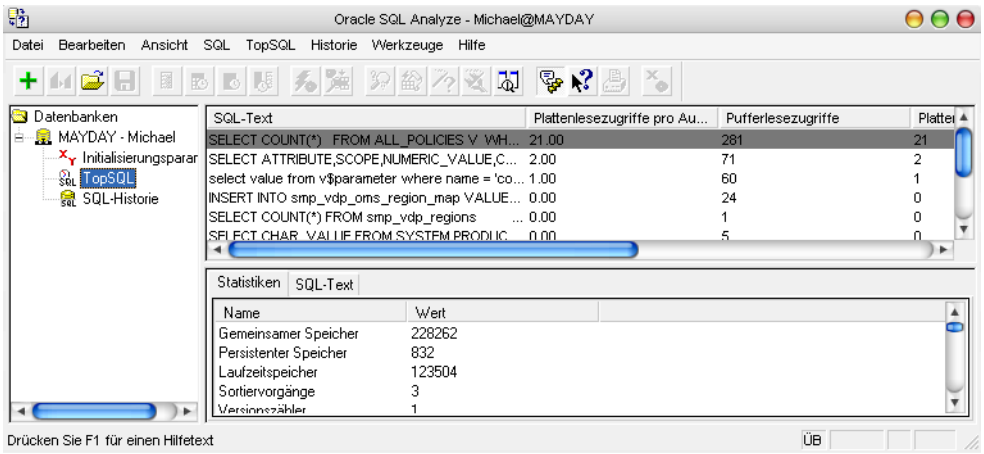

Abbildung 6: erster Start von SQL - Analyze

Zu sehen sind mehrere Fenster. Auf der Linken Seite sehen wir eine Baumstruktur. Für jede Datenbank wird ein vordefinierter Baum erstellt mit den drei zu sehenden Unterpunkten. Dazu gehören die Initialisierungsparameter, das TopSQL Fenster und die SQL - Historie. Auf der Rechten Seite sehen wir Informationen über SQL - Anweisungen und dazu gehörende Statistiken sowie Ausführungspläne.

5.3.1 Was ist TopSQL?

Mit TopSQL können Informationen über Ressourcennutzung einer SQL - Anweisung berechnet werden. Die Funktion stellt diese dann in den Rechten Fenster übersichtlich und strukturiert dar. Anhand der erstellten Informationen zu einer SQL - Anweisung können solche identifiziert werden, die die meisten Ressourcen benötigen. Danach kann diese zur Optimierung bearbeitet werden. TopSQL bietet mehrer Sortierungen nach folgenden gesichtspunkten an:

- Puffer - Cache Trefferrate
- Pufferlesezugriffe
- CPU Auslastung (für jede Ausführung)
- Plattenlesezugriffe (für jede Ausführung)
- verarbeitete Zeilen
- Sortiervorgänge

5.4 Anlegen von eigenen SQL - Anweisungen

Zu den schon bestehenden Knoten in der linken Baumstruktur können weiter Knoten hinzugefügt werden. Mit einem Rechts-klick und der Auswahl von Neue SQL erstellen im Kontextmenü lassen sich eigene SQL - Anweisungen untersuchen. Dazu gibt man in das Eigabefeld SQL - Anweisung die zu untersuchende Anweisung an und lässt sich Informationen wie Statistiken und Ausführungspläne für diese Anweisung erstellen. Zu sehen ist dies in der nächsten Abbildung.

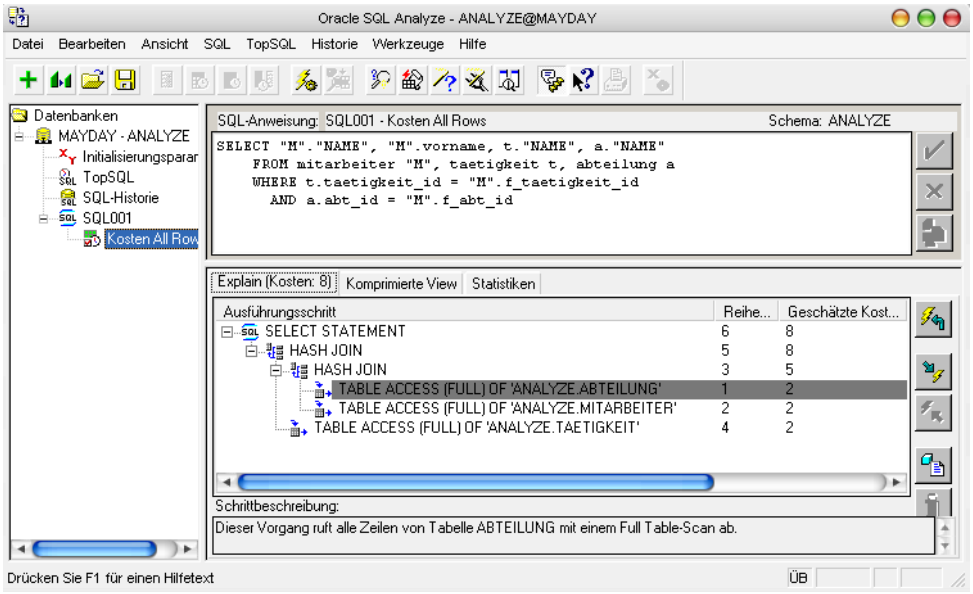

Abbildung 7: SQL - Analyze

Bericht
Zu jeder Anweisung kann ein Bericht erstellt werden. Dazu wählt man unter SQL den Punkt Bericht. Der Bericht wird in Form einer HTML-Datei erstellt und kann dann im Browser betrachtet werden (siehe Abbildung 8).

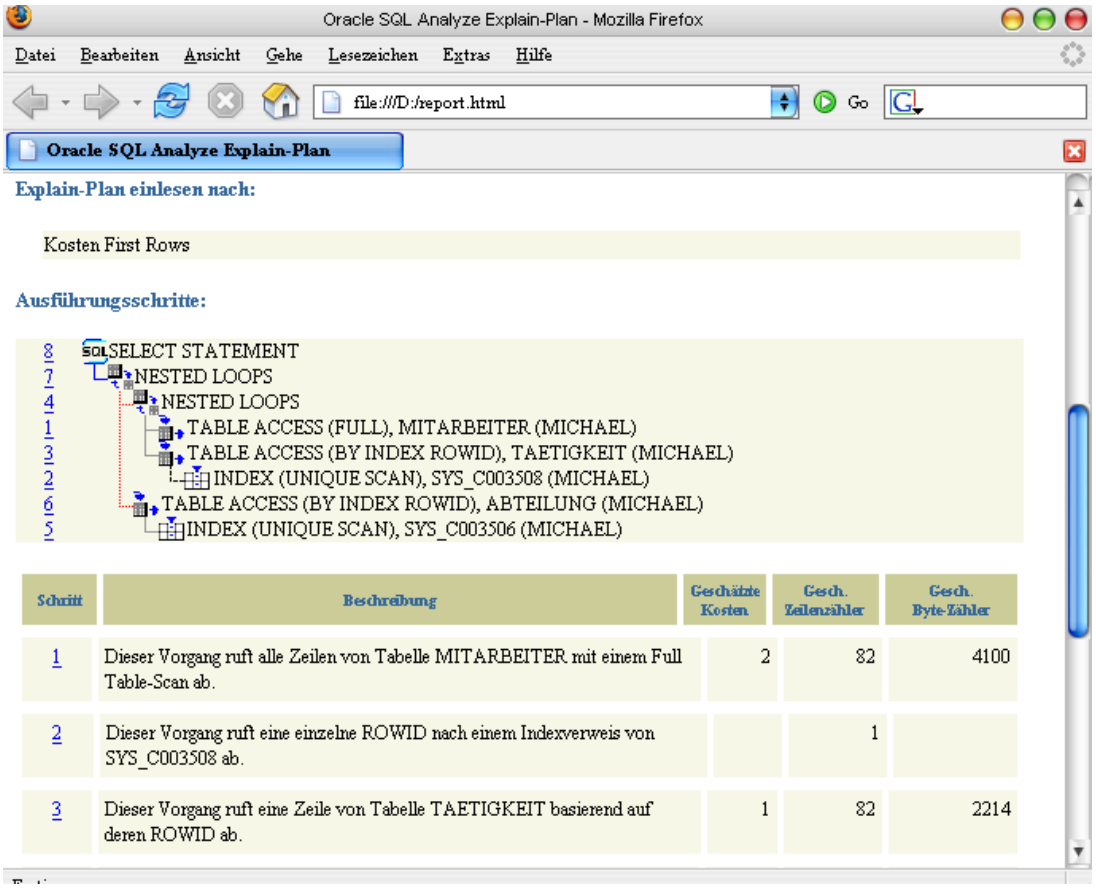

Abbildung 8: Bericht

Es werden alle Infomationen zu der SQL - Anweisung kompakt und strukturiert dargestellt. Zu jeder Zeile des Ausführungsplan gibt es auch eine entsprechende Erklärung.

6 Wie werden Ausführungspläne gelesen?

Das Verständnis von Ausführungsplänen ist nicht ganz einfach. Der entsprechende Anwender möchte schon ein wenig Hintergrundwissen über Datenbanken und der SQL - Sprache besitzen um solche Pläne interpretieren zu können. SQL - Analyze bringt jedoch eine kurze und knappte Erklärung für jede Zeile im Ausführungsplan von Hause aus mit. Dies dient zur Unterstützung das Verständisses des Anwenders. Gehen wir von folgender SQL - Anweisung dem dazugehörigen Ausführungsplan, mit der Optimierung *Kosten First Row*, aus (SQL - Analyze).

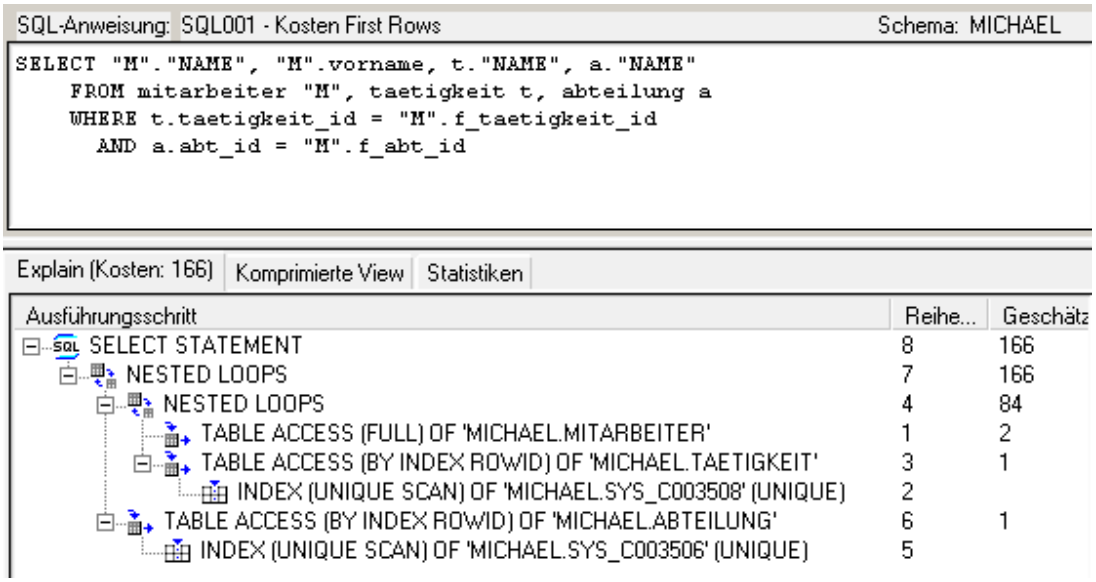

Abbildung 9: Beispiel - Anweisung

Erklärungen zu den einzelnen Zeilen

- (1) Dieser Vorgang ruft alle Zeilen von Tabelle MITARBEITER mit einem Full Table-Scan ab.
- (2) Dieser Vorgang ruft eine einzelne ROWID nach einem Indexverweis von SYS_C003508 ab.
- (3) Dieser Vorgang ruft eine Zeile von Tabelle TAETIGKEIT basierend auf deren ROWID ab.
- (4) Dieser Vorgang vergleicht jede Zeile der ersten untergeordneten Zeilenquelle mit allen Zeilen der zweiten untergeordneten Zeilenquelle, wobei die Zeilenpaare verknüpft werden, bei denen eine bestimmte Bedingung erfüllt ist.
- (5) Dieser Vorgang ruft eine einzelne ROWID nach einem Indexverweis von SYS_C003506 ab.
- (6) Dieser Vorgang ruft eine Zeile von Tabelle ABTEILUNG basierend auf deren ROWID ab.
- (7) Dieser Vorgang vergleicht jede Zeile der ersten untergeordneten Zeilenquelle mit allen Zeilen der zweiten untergeordneten Zeilenquelle, wobei die Zeilenpaare verknüpft werden, bei denen eine bestimmte Bedingung erfüllt ist.
- (8) Dieser Schritt in dem Plan gibt diese Anweisung als SELECT-Anweisung an

Ein Ausführungsplan wird immer von rechts nach links gelesen oder wenn wir uns diesen als Baumstruktur darstellen von den Blätter zur Wurzel hin. Deshlab bingt die Erklärung mit der Zeile die mit (1) markiert ist und geht dann in der oben dargestellten Rangfolge weiter.

7 Optimierungsbeispiel

Die Analyse von Ausführungsplänen wird hauptsächlich für die Optimierung einzelner Kostenaufwendigen Schritte im Ausführungsplan verwendet. Dazu gilt es Kostenaufwendige oder Ressourcenfressende Komponenten des Ausführungsplans zu identifizieren und diese mit geeigneten Mitteln Kostengünstiger zu gestalten.
Im folgenden wollen wir eine Kostenreduzierung an einem Beispiel durch hinzufügen von Indexes erläutern und darstellen.

In Abbildung 10 der Ausführungsplan einer einfachen SQL - Anweisung zu sehen. Die Kosten belaufen sich auf 9 Punkte. Bei diesem Beispiel zeigt SQL - Analyze mit dem Ausrufezeichen in der linken unteren Ecke der Abbildung an, dass es eine Schwachstelle im Ausführungsplan gibt.

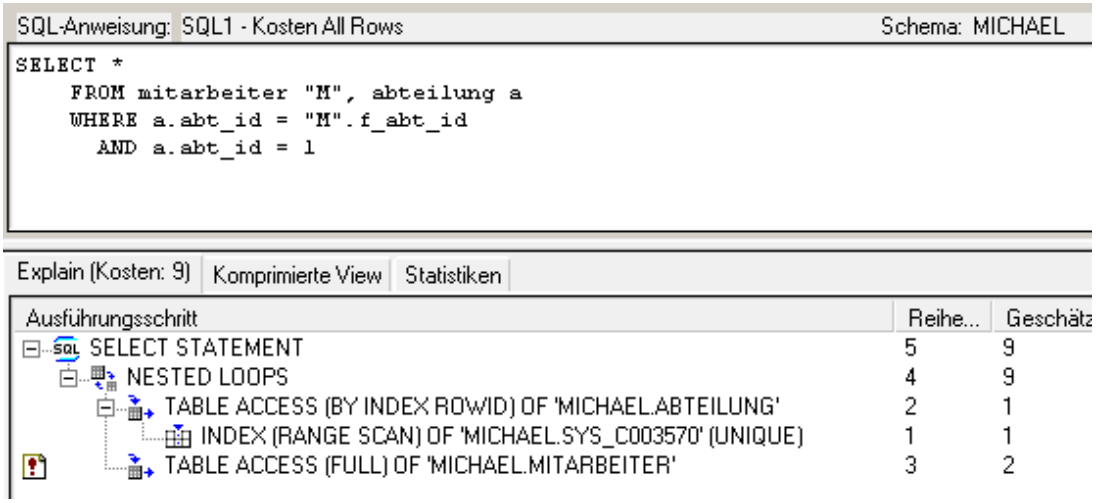

Abbildung 10: Beispiel ohne Index

Im vorliegenden Fall kann der Ausführungsplan durch setzen eines einzelnen Indexes um 4 Punkte reduziert werden. Durch das setzen eines Idexes auf die Spalte F_ABT_ID der Tabelle MITARBEITER lassen sich schnell und einfach die Kosten der Anfrage um 44% senken.

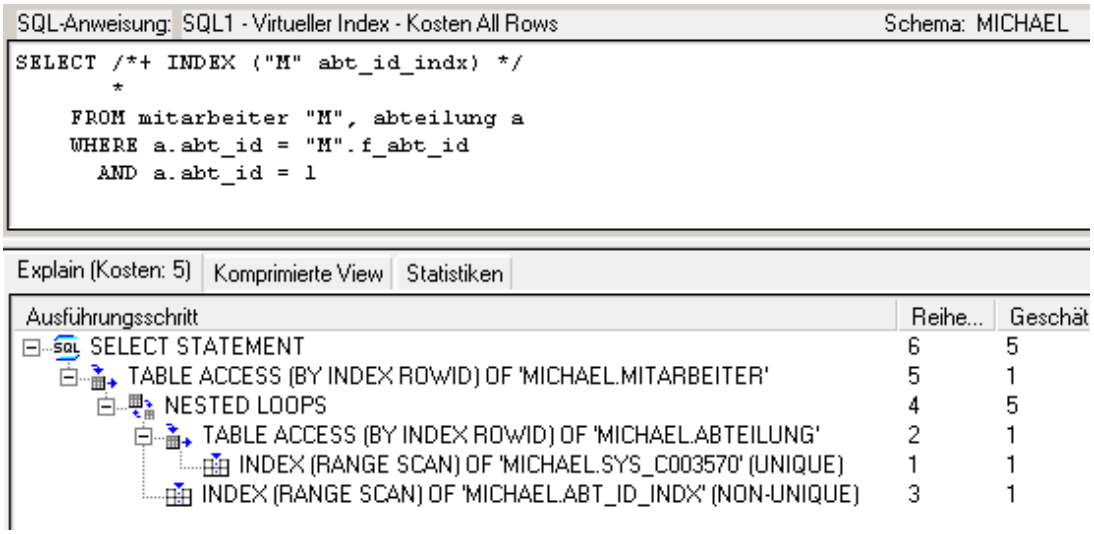

Abbildung 11: Beispiel mit Index

Wie man sehen kann hat es keine großen Auswirkungen bei unseren einfachen Beispiel. Um jedoch zu demonstrieren wie eine manuelle Optimierung durch den Anwender geschehen kann wird es ausreichen. Wenn wir bei großen und Kostenaufwendigen Anfragen ein vergleichbare Optimierung realisieren können ist das schon eine enorme Reduzierung der Kosten.

8 Quellenangaben

- Datenbanken: Implementierungstechniken,G.Saake, A.Heuer,Bonn 1999
- Oracle9i für den DBA, U. Herrmann u. andere Bonn 2000
- Oracle8 für den DBA, U. Herrmann u. andere Bonn 1998
- SQL - Tunning, D. Tow, 2003
- www.oracle.com