EDUCETH
Das Bildungsportal der ETH Zürich
Publikation von www.educeth.ch

Query Optimierung

Ein Leitprogramm von Andreas Meier

Inhalt und Lernziel:
Das Leitprogramm liefert die Grundlagen zur Optimierung von Datenbank-Abfragen.

Unterrichtsmethode: Leitprogramm
Das Leitprogramm ist ein Selbststudienmaterial. Es enthält alle notwendigen Unterrichtsinhalte, Übungen, Arbeitsanleitungen und Tests, die die Student/innen brauchen, um ohne Lehrperson lernen zu können.

Fachliches Review:
Juraj Hromkovic, Informationstechnologie und Ausbildung, ETH Zürich

Fachdidaktisches Review:
Juraj Hromkovic, Informationstechnologie und Ausbildung, ETH Zürich

Publiziert auf EducETH:
18. Juni 2007

Publizieren auf EducETH?
Möchten Sie eine eigene Unterrichtseinheit auf EducETH publizieren? Auf folgender Seite finden Sie alle wichtigen Informationen: http://www.educeth.ch/autoren

Weitere Informationen:
Weitere Informationen zu dieser Unterrichtseinheit und zu EducETH finden Sie im Internet unter http://www.educ.ethz.ch oder unter http://www.educeth.ch.

Query Optimierung

Einführung ins Tuning von Datenbank-Abfragen am Beispiel von Oracle

Ein Leitprogramm in Informatik

Schultyp: Fachhochschule

Schulerprobung am 7.Juli 2005
Vorlesung Datenbank Betrieb, 2.Semester
Studiengang Kommunikation und Informatik
Zürcher Hochschule Winterthur

Verfasst von
A.Meier

Version Oktober 2005

Vorwort

Liebe Leserin, lieber Leser,

Vielleicht haben Sie als Programmierer oder Studentin auch schon eine Datenbank-Applikation entwickelt. In der Testphase hat alles bestens funktioniert, aber in der produktiven Phase beschweren sich plötzlich die Benutzer, dass es „so langsam“ sei.

Bevor Sie jetzt hingehen und leistungsfähigere Hardware kaufen, sollten Sie zuerst die Datenbank tunen. In diesem Leitprogramm betrachten wir ein Teilgebiet des Datenbank-Tunings, nämlich das Optimieren von SQL-Abfragen.

An Beispielen, welche Sie selber ausprobieren können, erhalten Sie eine leicht verständliche Einführung in diese interessante Thematik.

Verwendung dieses Leitprogramms

Als Voraussetzung für dieses Leitprogramm sollten Sie gute Kenntnisse über eine relationale Datenbank sowie die Abfragesprache SQL mitbringen. Begriffe wie physisches Datenbanklayout, Tabellen oder Indices sollten Ihnen geläufig sein. Weiter sollten Sie wissen, wie man Tabellen und Indices erzeugt und wieder löscht.

Dieses Lernprogramm verwendet die Datenbank und Werkzeuge von Oracle. Sie sollten deshalb wissen, wie man den OEM (Oracle Enterprise Manager) startet und verwendet.

Technische Voraussetzungen

Für die Bearbeitung des Leitprogramms ist ein Computer mit einer lauffähigen Installation von Oracle9i oder neuer notwendig. Als Betriebssystem können Sie sowohl Windows als auch Linux/Unix verwenden. Alternativ reicht es auch, wenn Sie sich auf einen Oracle-Server mit entsprechender Client-Software einloggen können. In diesem Fall müssen Sie den Datenbank-Administrator (DBA) bitten, dass er Ihnen einen Account einrichtet.

Die Oracle-Software können Sie für Ausbildungszwecke kostenlos im WWW unter folgender Adresse herunterladen:

```
http://www.oracle.com
```

Einführung ins Thema

Worum geht es?

Sie haben in Ihrem Studium bereits einige Erfahrung gesammelt mit Datenbanken. Sie haben sich mit Datenbank-Design und Datenbankabfragen (engl. Database Queries) auseinander gesetzt. Sie haben komplexe SQL-Statements entwickelt und ausgeführt. Dabei haben Sie sich sicher auch gefragt, wie die Datenbank solche SQL-Statements verarbeitet.

Wie werden aus den einzelnen Datenbank-Tabellen die richtigen Datensätze extrahiert und korrekt zur Resultatsmenge hinzugefügt? Und vor allem, wie geschieht das auch bei grossen und sehr grossen Datenbanken effizient?

Queryplans, Optimizer und Datenbank-Tuning sind die Schlagworte dazu. Dieses Leitprogramm soll Ihnen einen ersten Einblick über das Performancetuning von Datenbanken geben.

Was wir bieten:

Dieses Leitprogramm soll die Grundlagen der Optimierung von Datenbankabfragen bieten:

- Wie werden Datenbank-Abfragen ausgeführt?
- Wann ist es sinnvoll, einen Index zu erstellen?
- Was versteht man genau unter dem Begriff Executionplan?
- Was ist ein Optimizer? Welche Arten von Optimizern gibt es?
- Wie werden meine Abfragen schnell?

Das Tuning von Datenbanken muss gelernt sein. Nach dem Durcharbeiten des Leitprogramms werden Sie wissen, dass es dazu Strategien braucht. Es wird Ihnen (hoffentlich) klar geworden sein, dass es für das erfolgreiche Optimieren von Datenbank-Abfragen sehr viel Erfahrung braucht.

Was wir nicht bieten können (und auch nicht bieten wollen):

- Die neuesten technischen Tipps und Tricks zu Oracle 9.4.6.12 oder wie die aktuellste Version zur Zeit gerade heisst
- Bedienungsanleitungen zu spezifischen Servern und Software
- Zertifizierung als Oracle Experte

Solche Themen halten wir für eher belanglos und kurzlebig. Wer sich trotzdem dafür interessiert, schaut in den (zahlreichen) Manuals nach oder sucht auf dem World Wide Web.

Inhaltsverzeichnis

Arbeitsanleitung

Mit Hilfe dieses Leitprogramms können Sie sich *selbständig* auf den Einstieg ins Optimieren von SQL-Abfragen vorbereiten. Das Programm besteht aus verschiedenen Kapiteln. Diese sind immer gleich strukturiert:

Übersicht Worum geht es in diesem Kapitel? Was ist zu tun?

Lernziele Was kann ich nach dem Bearbeiten des Kapitels?

Aufgabe / Lernkontrolle habe ich alles verstanden?

Lösungen zu den gestellten Aufgaben.

Übungen am Computer. Hier können Sie selber Hand anlegen

Begriffe die im Zusammenhang mit Datenbanken und Performancetuning von Bedeutung sind

1 Einführung: Der Optimizer

Übersicht

Vorbemerkungen

„Schon in den frühen Siebziegerjahren haben sich die Hersteller“ [Buff, S.213] von Datenbanken und Forschungsabteilungen von Universitäten überlegt, wie man die von E.F.Codd gewünschte Abfragesprache möglichst effizient in reale Zugriffe auf die Datentabellen umsetzen kann. Die Entwicklung von effizienten Optimizern für die Auswertung von Queries hat den Siegeszug der relationalen Datenbanken erst ermöglicht.

Einleitung

Der Optimizer ist eine Funktion der Datenbank, mit welcher die als am effizientesten geltende Art der Ausführung einer SQL-Anweisung ausgewählt wird. Dieser Schritt ist bei der Verarbeitung von DML-Anweisungen (Data Manipulation Language): SELECT, INSERT, UPDATE oder DELETE grundlegend. Es gibt häufig viele verschiedene Arten zur Ausführung einer SQL-Anweisung, z.B. durch Variieren der Reihenfolge, in der auf Tabellen oder Indizes zugegriffen wird. Die Prozedur, die von Oracle zur Ausführung einer Anweisung verwendet wird, kann sich stark auf die Geschwindigkeit der Anweisungsausführung auswirken.

Was lernen Sie hier?

In diesem Kapitel lernen Sie die verschieden Verfahren für den Zugriff auf die Daten in den Tabellen kennen. Sie werden sehen, mit welcher Strategie der Optimizer aus diesen Verfahren auswählt.

Lernziele

Nach diesem Kapitel sollten Sie wissen,

- wie das Datenbanksystem effizient auf die Daten zugreifen kann
- was die Funktionsweise des Optimizers ist
- wie Datenbankabfragen vom Datenbanksystem ausgewertet werden
- wann die zwei verschiedenen Optimizer-Modi von Oracle angewendet werden

Wie werden Queries ausgewertet?

Als Beispiel betrachten wir folgende (einfache) Abfrage:

```
SELECT NAME, SALARY
FROM EMPLOYEE E, DEPARTEMENT D
WHERE D.DNO = E.DNO AND SALARY > 150000;
ORDER BY SALARY
```

Wenn Sie diese Query an das Datenbanksystem absetzen, muss sie schlussendlich in einzelne Lese-Zugriffe auf die Harddisk übersetzt werden. Wie wird das erreicht?

Figure 1–1 SQL Processing Overview

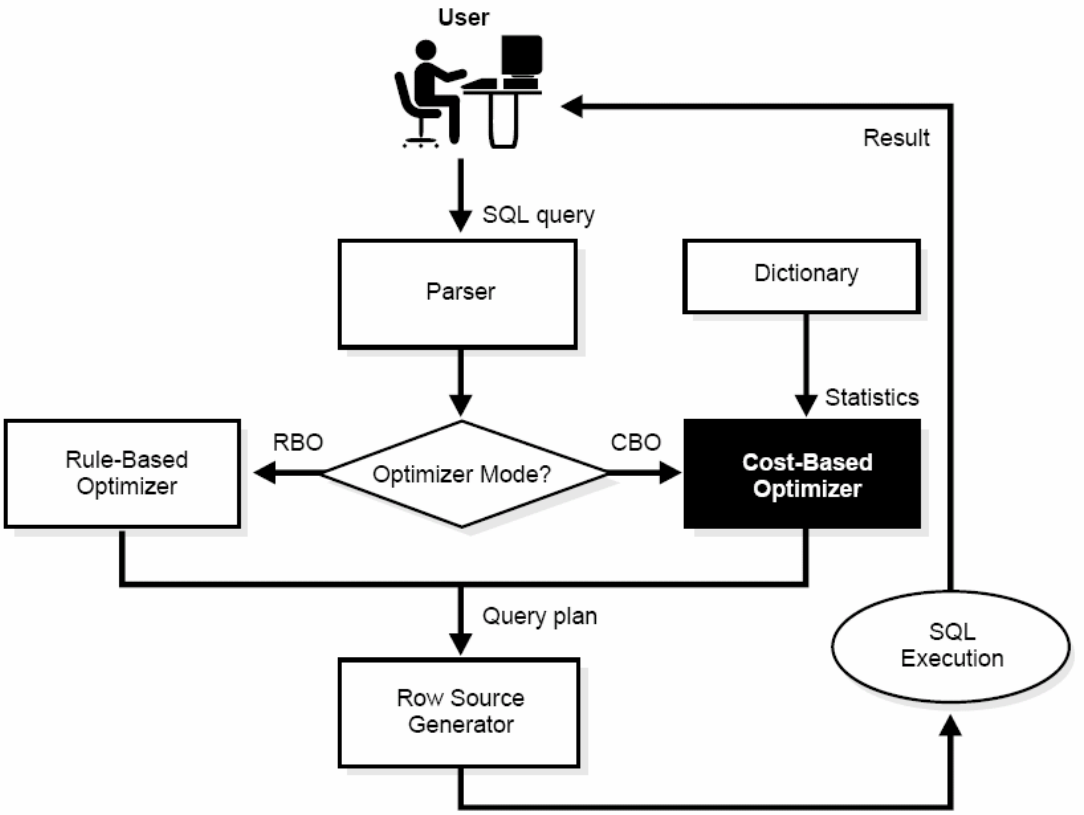

Abbildung 1 Wie werden Queries ausgewertet [Green] S.42

Eine SQL-Query wird wie folgt von der Datenbank ausgewertet:

- Der Parser analysiert die Query und prüft, ob sie syntaktisch und semantisch korrekt ist
- Aus der Query wird durch den Optimizer[1] der optimale Zugriffsplan (engl. Executionplan) berechnet. Dies kann auf zwei Arten geschehen:
  - Der RBO (Rule-based optimizer) basiert auf einer Menge von internen Regeln um den optimalen Zugriffspfad für die Auswertung der Query zu berechnen.
  - Der CBO (Cost-based optimizer) berechnet die Zugriffskosten an Hand von Statistiken zu den einzelnen Tabellen und Indices. Dies liefert im Allgemeinen die besseren Resultate.
- Der Row Source Generator erhält den Zugriffsplan vom Optimizer und erzeugt einen Ausführungsplan für die physischen Datenbankzugriffe
- Die SQL Execution Engine erzeugt mit Hilfe des optimalen Zugriffplans (execution plan) die Resultatmenge

Aufgabe 1

a) Studieren Sie nochmals die Abbildung 1 und überlegen Sie sich, wieso Oracle zwei verschiedene Optimizer verwendet. Was könnte der Grund sein?

b) Was könnte der Unterschied zwischen dem RBO und dem CBO sein.

c) Was ist die Voraussetzung, damit Sie den CBO verwenden können?

Ziele des kostenbasierten Verfahrens

Das Ziel des kostenbasierten Verfahrens ist standardmässig der beste Durchsatz oder die geringste Ressourcennutzung, die für die Verarbeitung aller Zeilen erforderlich ist, auf die die Anweisung zugreift.

Oracle kann eine Anweisung auch im Hinblick auf die kürzeste Antwortzeit oder die geringste Ressourcennutzung optimieren, die für die Verarbeitung der ersten Zeile erforderlich ist, auf die die SQL-Anweisung zugreift.

d) Bei welchen Computerprogrammen ist es wichtig, dass Sie schnell die erste (oder die ersten paar) Zeilen bekommen? Bei welchen, dass Sie schnell alle Zeilen bekommen? Geben Sie einige Beispiele an.

Statistiken für das kostenbasierte Verfahren

Beim kostenbasierten Verfahren werden Statistiken zur Schätzung der Kosten jedes Ausführungsplans verwendet. In diesen Statistiken werden die Datenverteilung und die Speichereigenschaften von Tabellen, Spalten, Indizes und Partitionen gemessen. Sie können diese Statistiken mit dem Befehl ANALYZE[2] generieren. Der Optimizer verwendet diese Statistiken, um zu schätzen, wieviel E/A, CPU-Zeit und Speicher für

[1] Auf die Funktionsweise des Optimizer wird später genauer eingegangen.

[2] ANALYZE ist ein Tool des Oracle Enterprise Managers und kann von dort gestartet werden. Mehr dazu später.

die Ausführung einer SQL-Anweisung mit einem bestimmten Ausführungsplan erforderlich ist.

Wann wird das kostenbasierte Verfahren verwendet?

Im Allgemeinen sollten Sie das kostenbasierte Verfahren für Ihre Anwendungen verwenden. Das regelbasierte Verfahren dient Anwendungen, die geschrieben wurden, bevor die kostenbasierte Optimierung zur Verfügung stand. Die kostenbasierte Optimierung kann effektiver sein, da ihre Auswertung auf den neuesten Informationen über die Leistung Ihrer Datenbank basiert. Sie wählt im Allgemeinen einen Ausführungsplan, der gleich gut oder besser als der beim regelbasierten Verfahren gewählte Plan ist. Dies gilt insbesondere bei umfangreichen Abfragen mit mehreren Joins oder Indizes.

Da der kostenbasierte Optimizer jedoch stark von Objektstatistiken abhängt, wird dringend empfohlen, diese Statistiken in regelmässigen Abständen mit dem Befehl ANALYZE zu aktualisieren.

Aufgabe 2

Studieren Sie im *Oracle9i Database Performance Tuning Guide and Reference* das Kapitel 1 *Introduction to the Optimizer*. (Der Guide ist als PDF diesem Leitprogramm beigelegt.)

Verfahren für den Datenbankzugriff

In diesem Abschnitt lernen Sie, wie eine Datenbank die gesuchten Informationen aus den Tabellen liest, Im Allgemeinen werden in Datenbanksystemen nur folgende Verfahren für den Tabellenzugriff eingesetzt [Ault, S.688]:

- **Vollständiger Tablescan**: (engl. Full tablescan) Wird üblicherweise verwendet, wenn es bei einer Tabelle keine oder nur ungeeignete Indices hat. Oder falls beim Zugriff mehr als 20 Prozent der Tabellendaten gelesen werden müssen.
- **Per RowID**: Überlicherweise als Ergebnis eines Indexscans. Dies ist die schnellste Zugriffsmethode auf Daten, die in einer Tabelle enthalten sind. Eine RowID verweist direkt auf eine bestimmte Zeile in einer Tabelle.
- **Clusterscan**: Nur im Zusammenhang mit Clustern einer Datenbank. Cluster werden (mit Ausnahme des Data Dictionary) nicht sehr häufig eingesetzt, sodass diese Zugriffsmethode im Allgemeinen nicht zur Anwendung kommt.
- **Hash-Scan**: Üblicherweise beim Fehlen von Indices, bei ungeeigneten Indices oder beim Zugriff auf mehr als 20 Prozent der Tabellendaten. Bevorzugt bei hohen Werten für SORT_AREA_SIZE[3], da die Grösse der Hash-Bereiche von der Grösse des Sortierbereichs abgeleitet wird.

Aufgabe 3

Welches Verfahren für den Datenbankzugriff verwendet Oracle bei Tabellen ohne Index?

[3] SORT_AREA_SIZE ist einer der unzähligen Parameter, mit welchem die ORACLE Datenbank auf spezifische Bedürfnisse angepasst werden kann.

Verfahren für den Indexscan

Indices sind die bevorzugte Vorgehensweise, um an Daten zu gelangen, wenn Sie weniger als 20 Prozent der Tabellendaten zurückliefern oder wenn Tabellenstatistiken darauf hinweisen, dass ein Index das geeignete Verfahren zum Nachschlagen oder Abrufen von Daten darstellt. Oracle verwendet folgende Verfahren für den Indexscan: [Ault, S.688]

- **Eindeutig**: Wird bei eindeutigen Indices zum Nachschlagen einzelner Zeilen eingesetzt.
- **Bereich**: (engl. Range indexscan) Wird eingesetzt, wenn Daten eines Wertebereichs angefordert werden.
- **Schnelles vollständiges Durchsuchen**: Kann an Stelle eines vollständigen Tablescans verwendet werden, falls die benötigten Daten im Index gespeichert sind.
- **Vollständiger Indexscan**: Kann verwendet werden, wenn sich die in einem Vergleich verwendete Spalte im Index befindet.
- **Bitmap-Scan**: Wird im Zusammenhang mit Bitmap-Indices verwendet.
- **Skip-Scan**: Ermöglicht das Durchsuchen der zweiten Schicht eines mehrstufigen Index.
- **Bitmap-Join**: Ermöglicht es, Tabellen mithilfe eines Bitmap-Index über die Join-Spalten im Vorfeld zu verknüpfen.

Aufgabe 4

Sie haben sieben verschiedene Verfahren für den Indexscan kennen gelernt. Schauen Sie sich den eindeutigen Index und das Verfahren „Eindeutig“ genauer an.

a) Was ist ein eindeutiger Index?

b) Welche Möglichkeiten gibt es, um einen eindeutigen Index zu erstellen?

Verfahren für Tabellen-Joins

Oracle greift auf zahlreiche Verfahren zum Verknüpfen von Tabellen (engl. Join) zurück, um den vielen Verknüpfungsbedingungen gerecht zu werden.

Diese Verfahren sind im Einzelnen [Ault, S.689]:

- **Verschachtelte Schleifen**: (engl. Nested Loops) Eine Tabelle wird als vorgebende äussere Tabelle bestimmt; die andere wird zur inneren Tabelle. Die mit der Join-Bedingung übereinstimmenden Werte werden in der äusseren Tabelle selektiert; die innere Tabelle wird dann nach diesen Werten durchsucht. Dies ist ein äusserst effektives Verfahren, wenn die äussere Tabelle um ein Vielfaches kleiner ist als die innere Tabelle.
- **Sort-Merge-Join**: (Sortierung und Zusammenfassung) Beide Tabellen werden nach den Zeilen durchsucht, die mit der Join-Bedingung übereinstimmen; die Ergebnisse werden sortiert und anschliessend ausgegeben.
- **Cluster-Join**: Wird im Zusammenhang mit Clustertabellen verwendet.
- **Hash-Join**: Die kleinere Tabelle wird in eine Hash-Tabelle konvertiert; wobei der Hash-Wert auf der Join-Kriterium basiert. Die Werte aus der Hash-Tabelle werden dann verwendet, um nach gehashten Stücken von Werten aus der grösseren Tabelle zu suchen. Die Tabellen werden in Stücke der Grösse von HASH_AREA_SIZE unterteilt. (HASH_AREA_SIZE ist üblicherweise doppelt so gross wie SORT_AREA_SIZE). Dies ist ein äusserst effektives Vorgehen, wenn die Tabellen ungefähr die gleiche Grösse haben und sehr gross sind.
- **Kartesisches Produkt**: Wenn nicht genügend Join-Bedingungen angegeben sind (bei einem Join von n Tabellen muss es n-1 Join-Bedingungen haben), werden die gesamten Inhalte der nicht richtig verknüpften Tabelle(n) als Join mit der Ergebnismenge verknüpft.
- **Index-Join**: Dieses Verfahren ermöglicht es, Indices statt Tabellen miteinander zu verknüpfen, falls die in der Abfrage vorkommenden Daten in den Indices enthalten sind. Bei den an diesem Join beteiligten Indices muss es sich um einspaltige Indices handeln (= sie dürfen nicht zusammengesetzt sein).

Aufgabe 5

a) Sie wollen zwei Tabellen „joinen“. Überlegen Sie sich, welche Tabelle beim Nested Loop die kleinere sein sollte? Begründen Sie Ihre Antwort.

b) Wieso erzielt der CBO bei diesem Verfahren typischerweise die besseren Resultate als der RBO? Begründen Sie Ihre Antwort.

EXPLAIN-PLANS

In Explain-Plans werden die Schritte beschrieben, die der Optimizer wählt, um die SQL-Anweisung auszuführen. Im Explain-Plan werden bei jedem Schritt entweder Zeilen aus der Datenbank abgerufen oder Zeilen als Eingabe von einem oder mehreren weiteren Schritten akzeptiert. Diese Reihe von Schritten, auch Ausführungspfad oder Zugriffspfad genannt, wird beeinflusst durch:

- Die Syntax der SQL-Anweisung
- Den Optimizer-Modus (Regel- oder Kostenbasiert)
- Verfügbare Indizes/Partitionen

Anzeigen des Ausführungspfads

Der zu einer Query gehörende Ausführungsplan kann einfach im OEM (Oracle Enterprise Manager) angeschaut werden. Dazu gehen Sie wie folgt vor:

Aufgabe

- Starten Sie den OEM und loggen sich als `system` mit der Rolle `sysdba` ein.
- Nach erfolgreicher Anmeldung im OEM öffnen Sie den *SQL Scratchpad*: (s.Abbbildung 2)

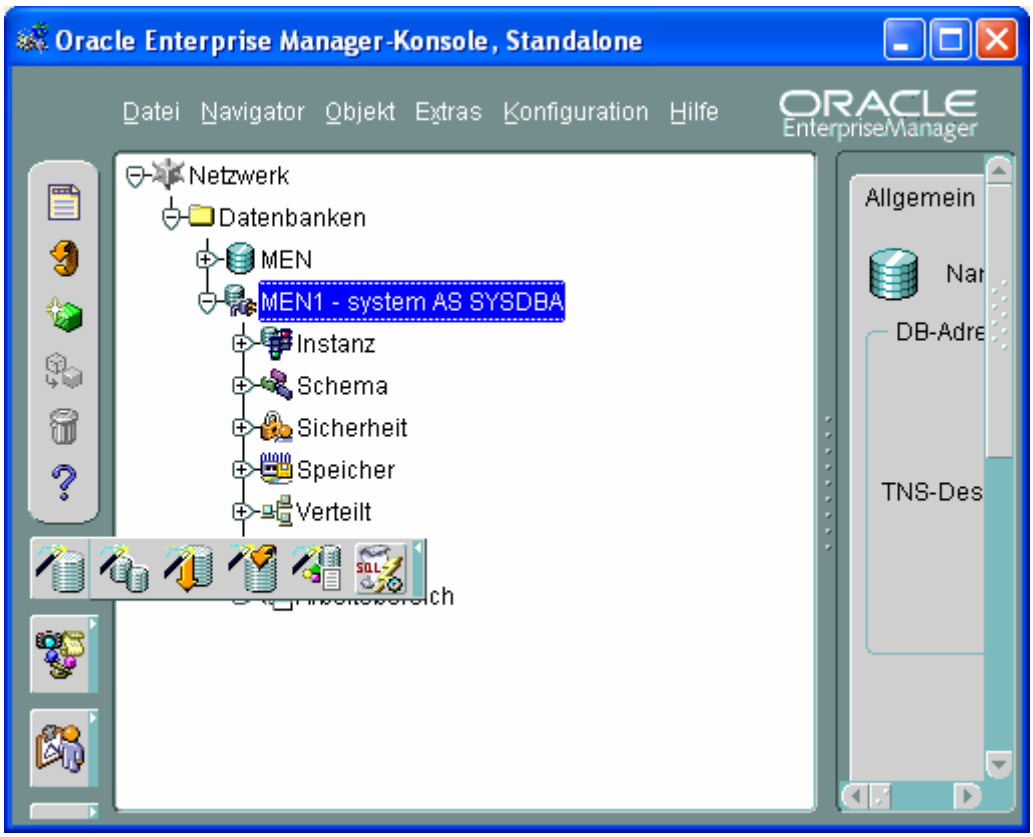

Abbildung 2 Oracle Enterprise Manager (OEM)

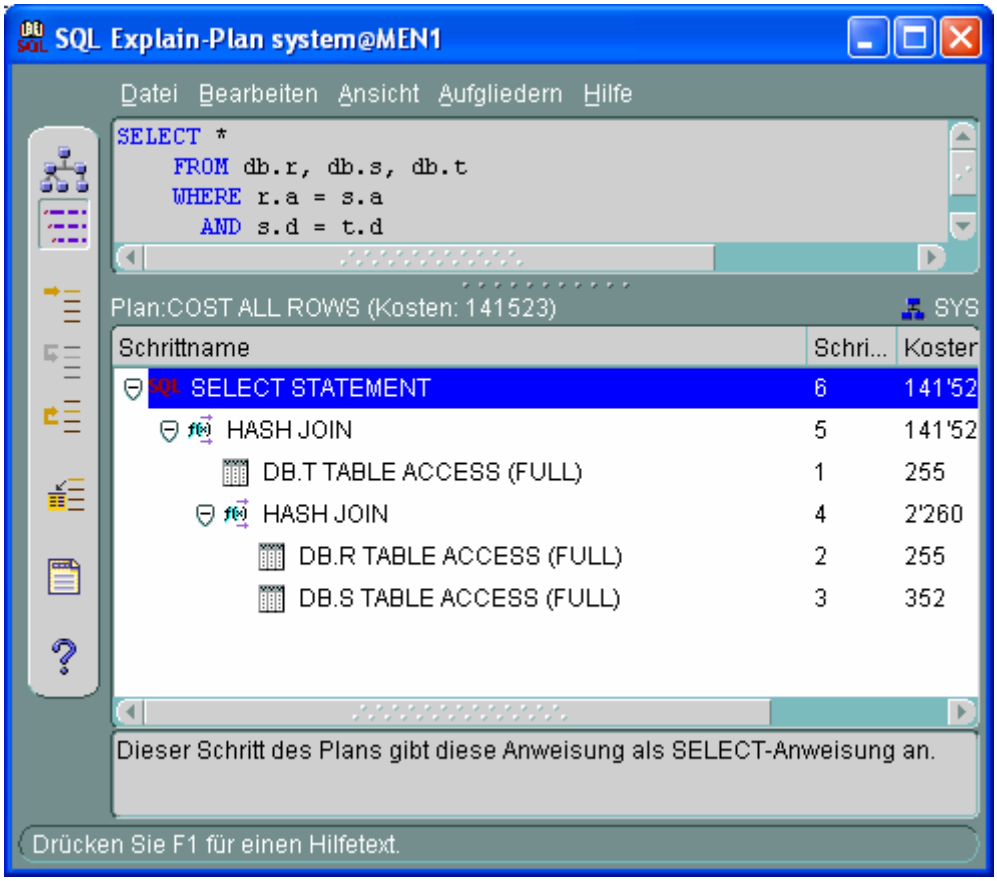

Abbildung 3 SQL Explain-Plan

Im *SQL Scratchpad* geben Sie die zu analysierende Query ein. Anschliessend können Sie sich unter der Option *Explain-Plan* den Ausführungsplan anzeigen lassen:

Aufgabe

- Geben Sie eine eigene Abfrage im *SQL Scratchpad* ein. (Tip: Oft sind in Oracle die Tabellen des Standardbeispiels installiert. Dann können Sie die Query vom Kapitelanfang verwenden)
- Zeigen Sie den Ausführungsplan Ihrer Query an.

Das Tool *SQL Explain-Plan* besteht im Wesentlichen aus drei Fenstern. Im oberen Fenster sehen Sie die Query und im mittleren Fenster den detaillierten Ausführungsplan. Im untersten Fenster steht eine Erklärung zum ausgewählten Schritt.

Werfen wir nun einen Blick auf den Ausführungsplan folgender Abfrage (Abbildung 4)

```
SELECT * FROM
DB.R, DB.S, DB.T
WHERE R.A = S.A AND S.D = T.D;
```

Der Ausführungsplan besteht aus insgesamt 6 Schritten. Die Reihenfolge der einzelnen Schritte wird in der Spalte *Schritt-Nr.* angegeben. Schritt 1 bis 3 besteht aus je einem vollständigen Tablescans pro Tabelle: TABLE ACCESS (FULL), d.h. es werden alle Tabellen komplett gelesen.

Wie Sie wissen, muss Oracle einen vollständigen Tablescan durchführen, wenn es keine Indices für die Tabellen gibt.

Schritt 4 und 5 verbindet die Tabellen mit je einem HASH JOIN. Die Tabellen im Beispiel sind alle gleich gross und haben je 1'000'000 Zeilen. In der Spalte *Zeilen* sind die zu erwartenden Zeilen der Operation angegeben. Oracle kann vor der Ausführung nicht wissen, wie viele Zeilen der Hash-Join zurückliefert. Darum werden die Zeilen vom Optimizer an Hand von Statistiken geschätzt[4].

Schritt 6 gibt das Resultat der Abfrage zurück.

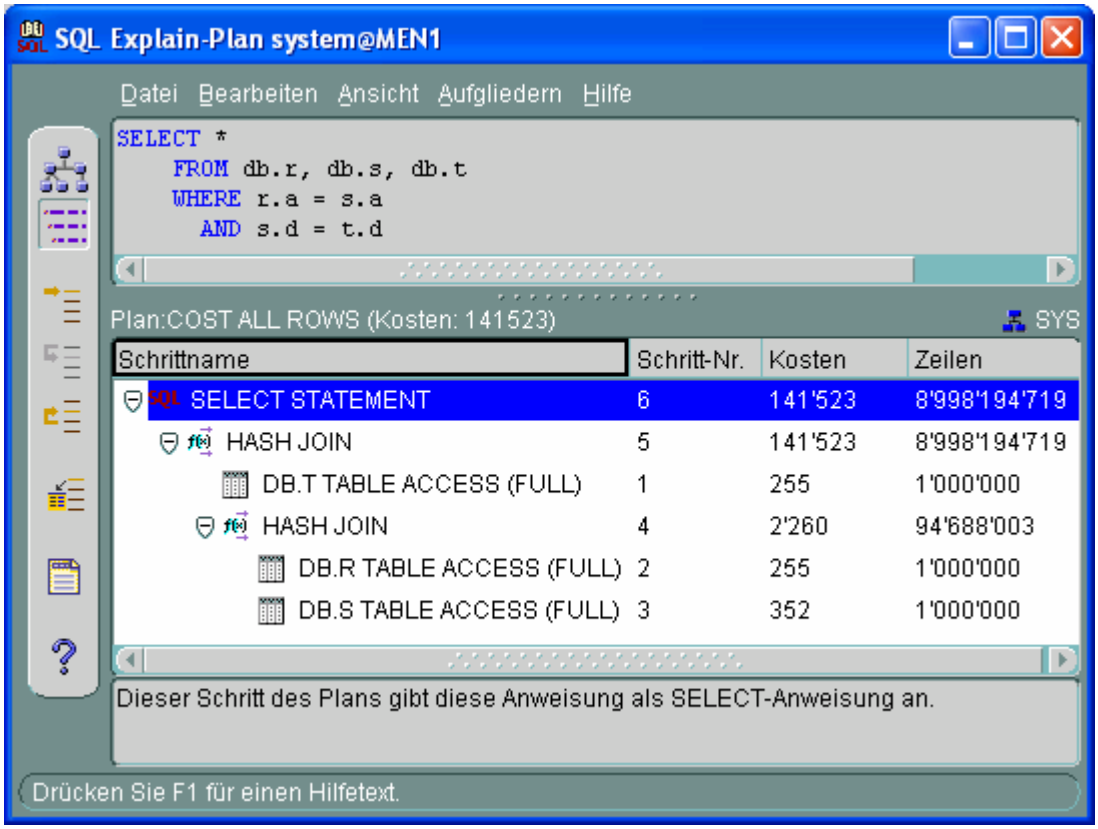

Abbildung 4 Ausführungsplan zu Beispiel-Query

Aufgabe

- Studieren Sie den Ausführungsplan Ihrer Query und vergleichen Sie ihn mit dem Beispiel.
- Experimentieren Sie mit den verschiedenen Möglichkeiten dieses Tools.

[4] Hinweis: Der kostenbasierte Oracle-Optimizer generiert die Hash-Tabelle mit der seiner Meinung nach kleineren der beiden Tabellen. Er bestimmt anhand der Statistiken, welche Tabelle kleiner ist. Veraltete Statistiken könnten also zu einer falschen Auswahl des Optimizers führen. Falls es keine Statistiken für die Tabellen gibt, wird der Rule-based Optimizer verwendet. Dieser Optimizer erzeugt einen Ausführungsplan nur anhand von Regeln.

Der Explain-Plan kann alternativ auch graphisch dargestellt werden:

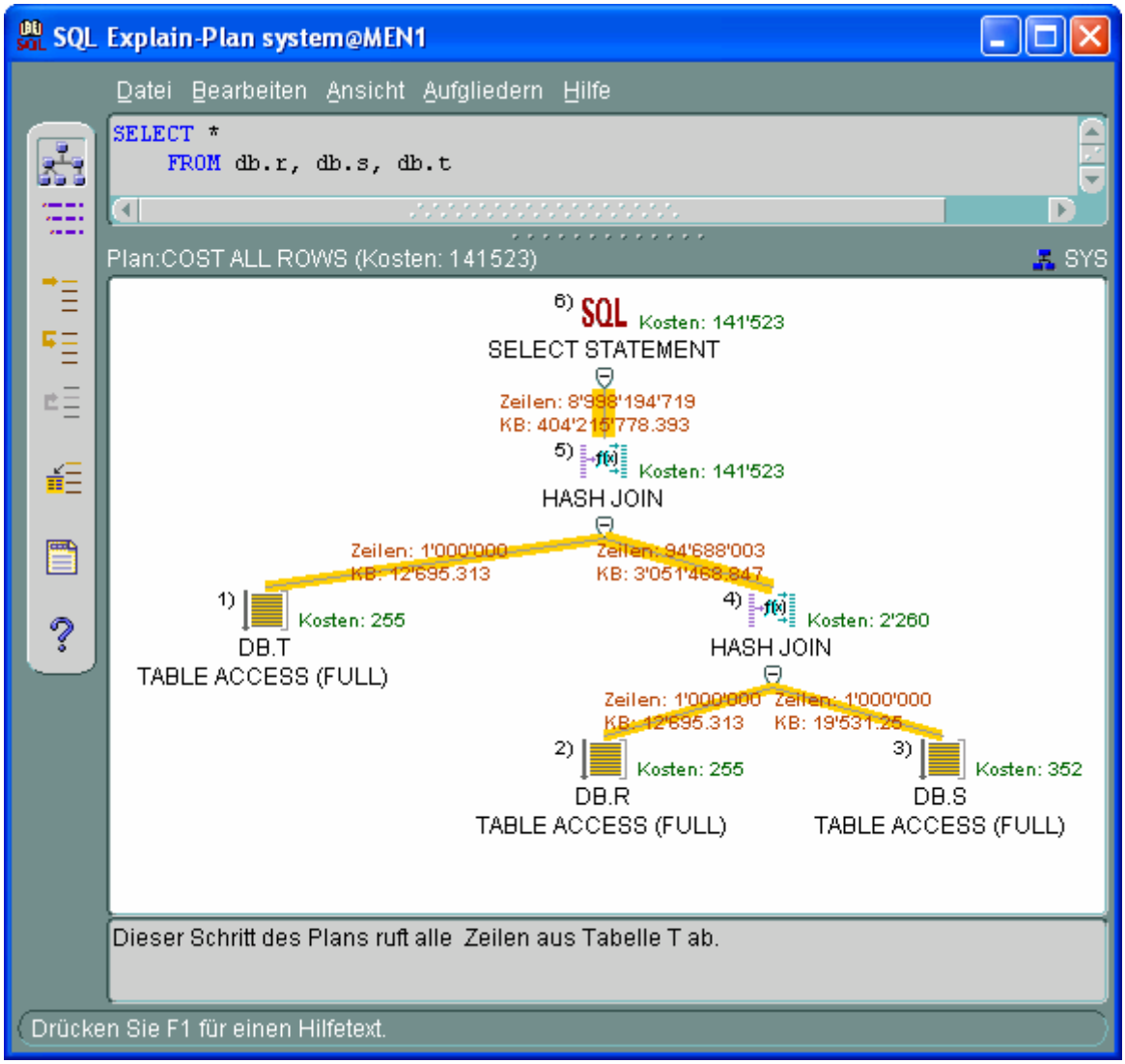

Abbildung 5 Graphische Ansicht des Explain-Plans

Zusammenfassung der Möglichkeiten von SQL Explain-Plan:

- Der Explain-Plan zeigt: die relativen Kosten in der Spalte *Kosten* für die Ausführung der Anweisung (bei Verwendung des kostenbasierten Optimizers). Diese Kosten resultieren aus der Erfassung mehrerer Faktoren, einschliesslich der Menge der Rechner-Ressourcen (beispielsweise die E/A- und CPU-Auslastung) sowie der Zeit zum Abschliessen der Ausführung der Anweisung (bei Verwendung des kostenbasierten Optimizers).
- den vom Optimizer gewählten Ausführungspfad
- die verwendeten Indizes/Partitionen
- die verwendeten Join-Methoden
- die Reihenfolge, in der Joins ausgeführt werden

2 Query Optimierung

Übersicht

Was lernen Sie hier?

Nach dem das erste Kapitel eher theoretisch war, bekommen Sie in diesem Kapitel die Gelegenheit, an einem interessanten Beispiel das Tuning von SQL-Queries selber zu üben.

Am Schluss sollten Sie ein Gefühl für die Möglichkeiten und Grenzen dieses Teilgebietes des Datenbank Tunings bekommen haben. Sie werden sehen, dass zum Teil massive Performanceverbesserungen erzielt werden können.

Was tun Sie?

Sie werden verschiedene Tabellen in Oracle anlegen und diese mit Daten füllen. Auf diesen Tabellen werden wir einige Abfragen durchführen und die Ausführungszeiten messen. Diese Abfragen werden Sie Schritt für Schritt zu optimieren versuchen, so dass die Ausführungszeiten möglichst klein werden.

Lernziele

Nachdem Sie dieses Kapitel durchgearbeitet haben,

- kennen Sie verschiedene Möglichkeiten, wie man eine Query tunen kann
- wissen Sie, was man bei Indices beachten muss
- wissen Sie, wieso Statistiken von den einzelnen Tabellen in Oracle wichtig sind
- können Sie die Statistiken selber erzeugen
- verstehen Sie was Optimizer Hints sind und wie Sie diese einsetzen können

Vorbereitung

Bevor Sie mit den Abfragen anfangen können, müssen wir zuerst die Tabellen kreieren und die Daten laden. (Eine Übersicht zu den verschiedenen Dateien finden Sie in Anhang B.)

Gehen Sie wie folgt vor:

- Loggen Sie sich im OEM als User `system` mit der Rolle `sysdba` ein.
- Erstellen Sie ein neues Schema `DB`
- Starten Sie SQL*Plus und erzeugen Sie mit untenstehendem Skript `CREATE_TABLES_DB.txt` die drei Tabellen `Kunden`, `Bestellungen` und `Details` (ev. müssen Sie noch den Name des Tablespace und des Schemas anpassen):

```
CREATE TABLE "DB"."KUNDEN"
    ("A" NUMBER(10) NOT NULL,
     "B" NUMBER(10) NOT NULL,
    PRIMARY KEY("A"))
    TABLESPACE "DATA01"
;
CREATE TABLE "DB"."BESTELLUNGEN"
    ("A" NUMBER(10) NOT NULL,
     "D" NUMBER(10) NOT NULL,
     "E" VARCHAR2(10) NOT NULL,
    PRIMARY KEY("D", "E"),
    CONSTRAINT "FK_KUNDEN" FOREIGN KEY("A")
    REFERENCES "DB"."KUNDEN"("A"))
    TABLESPACE "DATA01"
;
CREATE TABLE "DB"."DETAILS"
    ("D" NUMBER(10) NOT NULL,
     "E" NUMBER(10) NOT NULL,
     "F" NUMBER(10) NOT NULL)
    TABLESPACE "DATA01"
;
```

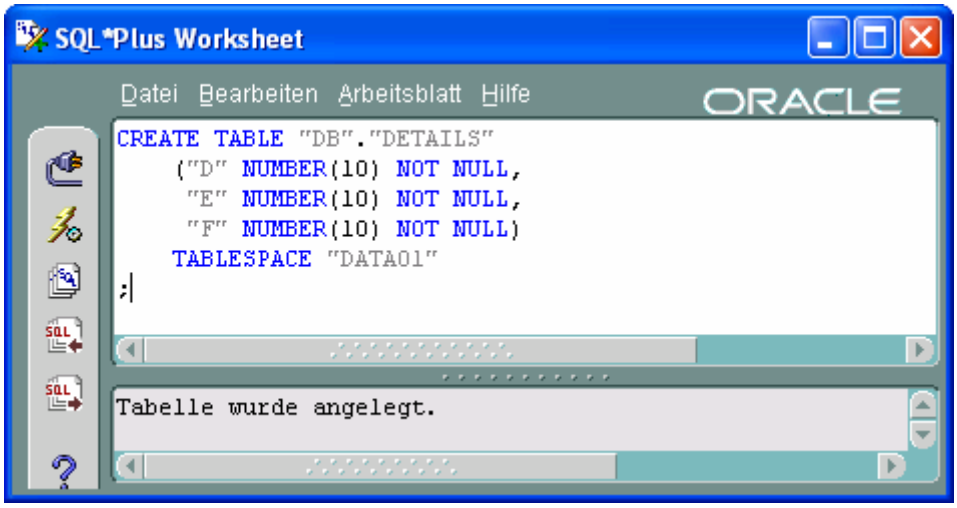

Abbildung 6 Anlegen von Tabellen mit SQL*Plus

Vorbereitung

Jetzt sind wir bereit, um die Daten in die Tabellen zu laden. Sie finden drei Dateien `KUNDEN.tabtxt`, `BESTELLUNGEN.tabtxt` und `DETAILS.tabtxt` im Verzeichnis zum Leitprogramm. Schauen Sie sich den Inhalt an. Die einzelnen Datensätze sind durch einen Tabulator getrennt.

Zum Laden der Daten benutzen wir das Oracle Tool *SQL*Loader*. Der SQL*Loader liest eine Menge von Daten aus einer Datei, generiert die notwendigen INSERT Befehle und gibt sie dem Oracle Kernel weiter. Mit dieser Methode können grosse Mengen von Rohdaten äusserst schnell geladen werden (engl. Bulk load)

Zum Laden der Daten gehen Sie wie folgt vor:

- Von der Kommandozeile geben Sie folgenden Befehl ein:

  ```
  sqlldr <user>/<passwor>@<db> control=<filename>
  ```

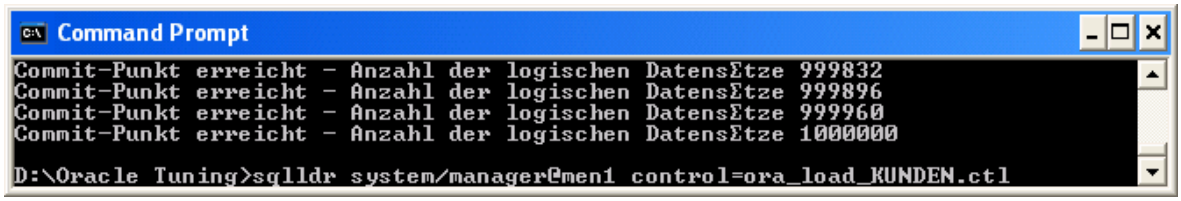

Abbildung 7 Laden der Daten mit SQLLDR

- Laden Sie alle drei Rohdaten-Dateien in die Tabellen. (ora_load_<Tablename>.ctl ist das Controlfile für den SQL*LOADER). Ev. müssen Sie das Schema in den Controlfiles anpassen.

Aufgabe 1

Überprüfen Sie die Anzahl Datensätze in SQL*Plus. Schreiben Sie dazu eine Query.

Notieren Sie sich die Tabellengrössen für die nächsten Aufgaben.

Aufgabe 2

Nachdem die Datenbank für das Tuning bereit ist, legen wir los.

Hier und in den folgenden Aufgaben geht es um die Optimierung von dieser Query:

```
SELECT * FROM
DB.KUNDEN K, DB.BESTELLUNGEN B, DB.DETAILS D
WHERE K.A = B.A AND B.D = D.D;
```

- Geben Sie die Query im SQL Scratchpad ein und lassen Sie sich den Explain-Plan anzeigen.
- Führen Sie die Query aus. Notieren Sie sich die Ausführungszeit.

Aufgabe 3

Nehmen Sie die gleiche Query wie bei der letzten Aufgabe und vertauschen Sie die Reihenfolge der Tabellen `DB.KUNDEN` und `DB.DETAILS` in der FROM-Clause.

```
SELECT * FROM
DB.DETAILS D, DB.BESTELLUNGEN B, DB.KUNDEN K
WHERE K.A = B.A AND B.D = D.D;
```

- Geben Sie die Query im SQL Scratchpad ein und lassen Sie sich den Explain-Plan anzeigen.
- Führen Sie auch diese Query aus. Notieren Sie sich die Ausführungszeit. Was stellen Sie fest? Haben Sie eine Erklärung dafür?

Verwendung des Cost-based Optimizer (CBO)

Als nächstes wollen wir den Cost-based Optimizer verwenden. Dazu müssen wir zuerst die Tabellen analysieren. Wenn wir Tabellen und Indices analysieren, erstellt Oracle eine Statistik und speichert sie im Data Dictonary. Diese Statistiken sollten in regelmässigen Abständen oder nach grösseren Änderungen aktualisiert werden.

Übersicht CBO

Das kostenbasierte Verfahren setzt sich aus folgenden Schritten zusammen:

1. Der Optimizer generiert, basierend auf den verfügbaren Zugriffspfaden und Hints, eine Reihe von potentiellen Ausführungsplänen für die Anweisung.

2. Der Optimizer schätzt die Kosten für jeden Ausführungsplan basierend auf der Datenverteilung und der Speichereigenschaften-Statistik[5] für die Tabellen, Cluster und Indizes im Data Dictionary.
3. Die Kosten stellen einen geschätzten Wert dar, der sich proportional zur erwarteten Ressourcennutzung verhält, die für die Anweisungsausführung mit dem Ausführungsplan benötigt wird. Der Optimizer berechnet die Kosten basierend auf den geschätzten Rechner-Ressourcen, die für die Ausführung der Anweisung mit dem Plan erforderlich sind, einschliesslich (jedoch nicht beschränkt auf) E/A, CPU-Zeit und -Speicher. Die Ausführung serieller Ausführungspläne mit höheren Kosten nimmt mehr Zeit in Anspruch als solche mit geringeren Kosten. Bei Verwendung eines parallelen Ausführungsplans[6] steht die Ressourcennutzung nicht direkt in Bezug zur abgelaufenen Zeit.
4. Der Optimizer vergleicht die Kosten der Ausführungspläne und wählt den kostengünstigsten Plan.

Vorbereitung

Für die nächste Aufgabe müssen Sie zuerst Ihre drei Tabellen analysieren. Dazu gehen Sie wie folgt vor:

- Im OEM wählen Sie *Objekt→Analysieren* und bei den Standardoptionen *weiter*.
- In Objektauswahl wählen Sie die drei Tabellen aus. (s.Abbildung)
- Drücken Sie *Weiter* und akzeptieren Sie die Standardeinstellungen auf den nächsten Bildschirmen. Starten Sie die Analyse mit *Beenden*.

[5] Sie müssen die Schemaobjekte Ihrer Datenbank mit dem Befehl ANALYZE analysieren, um die Statistiken auf einem aktuellen Stand zu halten. Wenn das Datenbankschema nicht regelmässig analysiert wird, verwendet der Optimizer möglicherweise ungültige Statistiken und wählt nicht die effektivste Zugriffsmethode.

[6] Für die parallele Ausführung brauchen Sie natürlich ein Mehrprozessor-System mit 2 oder mehr Prozessoren

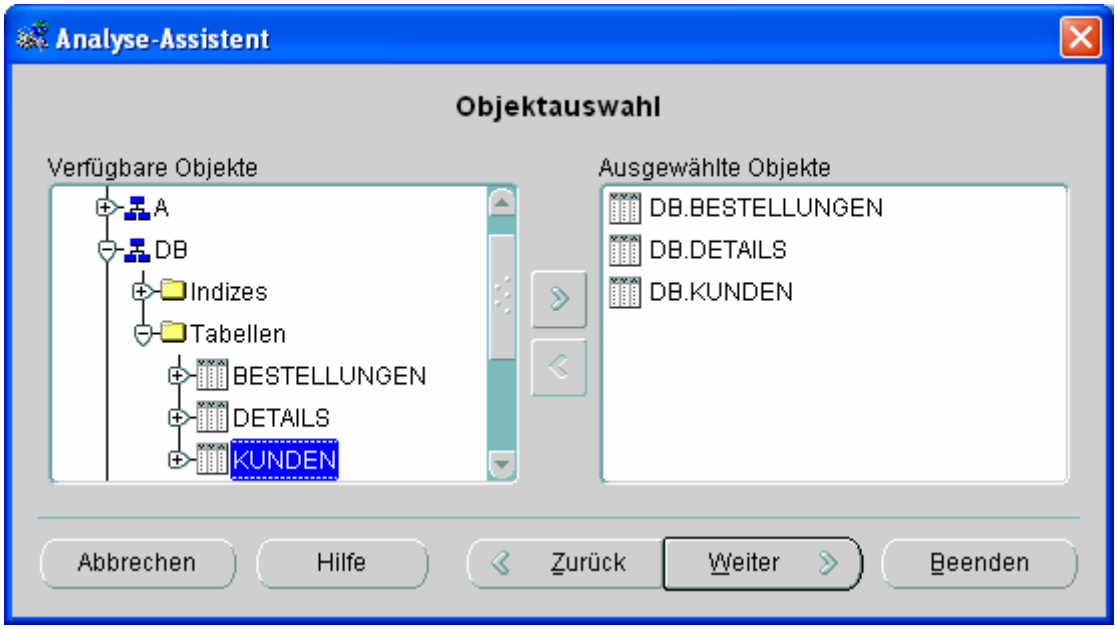

Abbildung 8 Berechnung der Statistiken

Aufgabe 4

- Nachdem die Tabellen erfolgreich analysiert wurden, führen Sie die gleiche Query nochmals aus:

```
SELECT * FROM
DB.KUNDEN K, DB.BESTELLUNGEN B, DB.DETAILS D
WHERE K.A = B.A AND B.D = D.D;
```

- Geben Sie die Query im *SQL Scratchpad* ein und lassen Sie sich den *Explain-Plan* anzeigen.
- Vergleichen Sie den Ausführungspfad mit demjenigen von Aufgabe 3. Was stellen Sie fest?
- Führen Sie die Query aus. Notieren Sie sich wieder die Ausführungszeit.

Verwendung von Indices

Bis jetzt haben wir noch keine Indices verwendet. Wie Sie sicher wissen, können mit Hilfe von Indices die Abfragen beschleunigt werden.

Falls Sie keinen Index zu einer Tabelle haben, muss die Datenbank jedes Mal die ganze Tabelle durchsuchen (Full tablescan)

Hinweis: Falls Sie ein *Primary Key Constraint* definiert haben, legt Oracle automatisch einen Index auf den Primary Key an.

Aufgabe 5

In dieser Aufgabe legen wir zusätzlich Indices an. Was erwarten Sie von der Ausführungszeit?

- Erstellen Sie die Indices in SQL*Plus mit folgenden Befehlen (ev. müssen Sie noch den Tablespace anpassen):

```
CREATE INDEX BEST_A_IDX ON DB.BESTELLUNGEN(A)
  TABLESPACE "INDX";
CREATE INDEX BEST_D_IDX ON DB.BESTELLUNGEN(D)
  TABLESPACE "INDX";
CREATE INDEX KUND_A_IDX ON DB.KUNDEN(A)
  TABLESPACE "INDX";
CREATE INDEX DET_D_IDX ON DB.DETAILS(D)
  TABLESPACE "INDX";
```

- Geben Sie wieder die Query im *SQL Scratchpad* ein und lassen Sie sich den *Explain-Plan* anzeigen. Was stellen Sie fest?
- Vergleichen Sie den Ausführungspfad mit dem aus Aufgabe 3. Was stellen Sie fest?
- Führen Sie die Query aus. Notieren Sie sich die Ausführungszeit.

Verwendung von *Optimizer Hints*

Ein wichtiges Merkmal von Oracle besteht in der Möglichkeit zum Angeben von Optimizer Hints (Hinweisen an den Optimizer). Mit diesen Hinweisen können Sie den Optimizer veranlassen, einen von Ihnen bestimmten Ausführungspfad für die Abfrage zu verwenden.

Die Hinweise werden in den Kommentaren zu den SQL-Befehlen SELECT, INSERT und UPDATE angegeben: Ein Hint beginnt mit **`/*+`** und endet mit **`*/:`**

```
SELECT /*+ Hint */
<Anweisungsrumpf>
```

In Oracle gibt es Hints für die Wahl von

- Zugriffsmethoden
- Join-Reihenfolgen und Umwandlungen
- Parallelen Operationen (nur auf Mehrprozessor-Rechnern verfügbar)
- Sonstige

In der Tabelle sind einige Hints in Oracle mit ihrer Bedeutung aufgeführt [Ault, S.720]:

Hint	Bedeutung
+	Muss direkt auf dem Kommentarindikator folgen. Teilt Oracle mit, dass es sich um einen Hint handelt.
CHOOSE	Standartwert; falls statistische Werte zur Verfügung stehen, wird der statistikorientierte (CBO), sonst der regelbasierte (RBO) Optimizer verwendet.
RULE	Verwendet den RBO
Hints für Zugriffsmethoden	
CLUSTER(TABLE)	Teilt dem Optimizer mit, für den Zugriff auf die Tabelle einen Clusterscan vorzunehmen.
FULL(TABLE)	Teilt dem Optimizer mit, die angegebene Tabelle vollständig zu durchsuchen (full table-scan)
HASH(TABLE)	Teilt dem Optimizer mit, für den Zugriff auf die Tabelle explizit das Hash-Verfahren anzuwenden.
HASH_AJ(TABLE)	Wandelt eine NOT IN-Abfrage in einen Hash-Antijoin um.
ROWID(TABLE)	Erzwingt das Durchsuchen der ROWID für die angegebene Tabelle.
INDEX(TABLE) [INDEX]	Erzwingt für die angegebene Tabelle einen Indexscan mit den angegebenen Indices.
Hints für Join-Operationen und Umwandlungen	
ORDERED	Erzwingt die Verknüpfung von Tabellen in der angegebenen Reihenfolge. Wenn Sie wissen, dass Tabelle X weniger Zeilen enthält, kann deren Platzierung in der ersten Position die Ausführung bei einem Join beschleunigen.
STAR	Erzwingt, dass die grösste Tabelle bei einem Join zuletzt berücksichtigt wird, wobei für den Index ein Join für verschachtelte Schleifen verwendet wird.
PUSH_SUBQ	Sorgt dafür, dass nicht zusammengefasste Subqueries zum frühestmöglichen Zeitpunkt im Ausführungsplan ausgewertet werden.
Hints für Join-Operationen	
USE_HASH(TABLE)	Veranlasst Oracle dazu, jede angegebene Tabelle über einen Hash-Join mit einer anderen Zeilenquelle zu verknüpfen.
USE_NL(TABLE)	Zwingt eine verschachtelte Schleife dazu, die angegebene

	Tabelle als äussere Tabelle zu verwenden.
USE_MERGE(TABLE. [,TABLE])	Erzwingt die Verknüpfung der angegebenen Tabellen über einen Sort-Merge Join.

Aufgabe 6

In dieser Aufgabe wollen wir den Optimizer anweisen, dass er die Indices verwenden soll. Dazu spezifizieren wir die folgenden Hints:

```
SELECT  /*+ ORDERED INDEX(K) INDEX(K) INDEX(D)
USE_NL(B D)*/
* FROM
DB.KUNDEN K,  DB.BESTELLUNGEN B, DB.DETAILS D
WHERE K.A = B.A AND B.D = D.D;WHERE K.A = B.A
AND B.D = D.D;
```

- Schauen Sie in der Oracle Dokumentation nach, was die einzelnen Hints bedeuten.
- Fügen Sie der Query den Hint zu und studieren Sie den *Explain-Plan*. Werden die Indices nun verwendet?
- Führen Sie die Query aus. Notieren Sie sich die Ausführungszeit.

Aufgabe 7

Wie gut ist der CBO im Vergleich zum RBO? Nachdem wir die Indices zu den Tabellen angelegt haben, geben wir dem Optimizer einen Hint, das er den RBO verwenden soll.

```
SELECT  /*+ RULE */
* FROM
DB.KUNDEN K, DB.BESTELLUNGEN B, DB.DETAILS D
WHERE K.A = B.A AND B.D = D.D;
```

- Fügen Sie der Query den Hint zu und vergleichen Sie den *Explain-Plan* mit den vorherigen. Was stellen Sie fest?
- Führen Sie die Query aus. Notieren Sie sich die Ausführungszeit.

Aufgabe 8

Diese Liste der Hints ist nicht vollständig. Studieren Sie deshalb im *Oracle9i Database Performance Tuning Guide and Reference* das Kapitel 5 *Optimizer Hints.*.

Hinweis: Wie Sie gesehen haben, sind die Hints ein mächtiges Werkzeug für das Datenbank Tuning., Für die Verwendung von Hints ist es aber äusserst wichtig, dass Sie sich sehr gut mit der zu optimierenden Applikation auskennen!

Aufgabe 9 (optional)

Überlegen Sie sich eine komplexere Query mit einer oder mehreren Subqueries und schauen Sie sich die Ausführungspläne an. Können Sie diese mit Hints optimieren? Experimentieren Sie mit weiteren Queries!

3 Lösungen zu den Aufgaben

Lösungen zu Kapitel 1: Der Optimizer

Lösungen Aufgabe 1

a) Der Grund ist historischer Art. Oracle hat zuerst den einfacheren RBO entwickelt. Der RBO erzielt im Allgemeinen gut Ergebnisse, berücksichtig aber nicht die Grösse der Tabellen und die Verteilung der Daten. Dies ist aber wichtig, um die effizienteste Reihenfolge zu bestimmen, in der die Tabellen „gejoint“ werden sollen.

Deshalb hat Oracle den CBO entwickelt, der die Grösse der Tabellen und die Verteilung der Daten berücksichtigt. Oracle empfiehlt, heute nur noch den CBO zu verwenden.

b)

Das regelbasierte Verfahren (RBO)

Beim regelbasierten Verfahren wählt der Optimizer einen Ausführungsplan, der auf den verfügbaren Zugriffspfaden sowie den Rangfolgen dieser Zugriffspfade basiert. Die regelbasierte Optimierung kann für den Zugriff auf relationale Daten und Objekttypen verwendet werden. Die Rangfolge, die Oracle für die Zugriffspfade verwendet, ist heuristisch. Falls es mehrere Möglichkeiten zur Ausführung einer SQL-Anweisung gibt, wird beim regelbasierten Verfahren stets der Vorgang mit der niedrigeren Rangfolge verwendet. Vorgänge mit einer niedrigeren Rangfolge werden meist schneller ausgeführt als Vorgänge, denen Konstrukte mit höheren Rangfolgen zugeordnet sind.

Das kostenbasierte Verfahren (CBO)

Beim kostenbasierten Verfahren legt der Optimizer den effizientesten Ausführungsplan fest, indem er die verfügbaren Zugriffspfade (= vorhandene Indices) berücksichtigt und Informationen mit einbezieht, die auf Statistiken im Data Dictionary für Schemaobjekte (= Tabellen, Cluster, Indizes und Partitionen) basieren, auf die die Anweisung zugreift. Beim kostenbasierten Verfahren werden ebenfalls Hints[7] oder Optimierungsvorschläge berücksichtigt, die in einem Kommentar in der Anweisung gespeichert werden.

c) Um den CBO verwenden zu können, müssen Sie zuerst Statistiken zu den Tabellen und Indices erstellen. Wie das geht, werden Sie im nächsten Kapitel lernen.

d) erste Zeilen: Interaktives GUI, etc.

alle Zeilen: Batchprogramme, Statistiken, etc.

[7] Hints werden durch die Software-Entwickler in SQL-Queries eingefügt. Sie geben dem Optimizer Hinweise wie die Query ausgeführt werden soll. Hints werden im nächsten Kapitel behandelt.

Hinweis Aufgabe 2

Oracle hat eine sehr umfangreiche Dokumentation, verteilt auf mehrere Online-Bücher im HTML oder PDF-Format. Falls Sie mit Oracle arbeiten möchten, ist wichtig, dass Sie sich mit diesen Büchern vertraut machen. Weitere Informationen finden Sie unter `www.oracle.com`.

Lösung Aufgabe 3

Oracle führt in diesem Fall im Allgemeinen einen „Full Tablescan“ durch. Falls die Tabelle gross ist, dauert das lange.

Lösung Aufgabe 4

a) Jeder Wert kommt im Index genau ein Mal vor.

b) Wenn Sie einen Primärschlüssel auf eine Tabelle erstellen, erzeugt Oracle implizit einen eindeutigen Index. Explizit können Sie einen eindeutigen Index durch Angabe von UNIQUE im CREATE INDEX Statement erzeugen.

Lösung Aufgabe 5

a) Die äussere Tabelle muss kleiner als die innere sein.

b) Der CBO weiss auf Grund der Statistiken, welche Tabelle die kleinere ist und kann deshalb immer die kleinere Tabelle als äussere wählen.

Der RBO hat keine Angaben über die Tabellengrössen!

Lösungen zu Kapitel 2: Query Optimierung

Lösungen Aufgabe 1

Mit `count(*)` können Sie die Datensätze zählen:

Tabelle `Kunden` hat 10'000,

Tabelle `Bestellungen` 100'000 und

Tabelle `Details` 1 Million Datensätze: (s.unten)

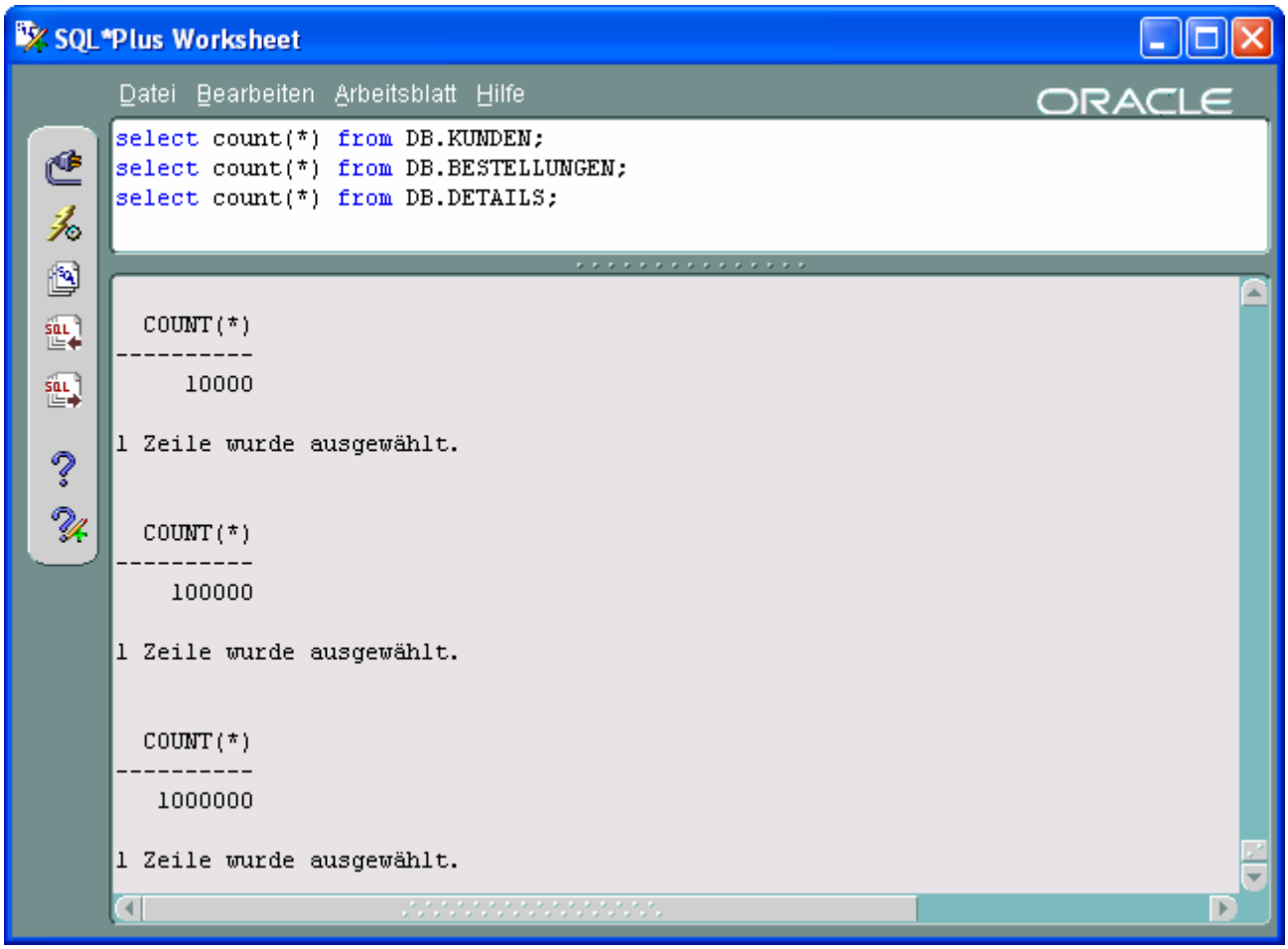

Abbildung 9 Lösung zu Aufgabe 1

Lösung Aufgabe 2

Oracle verwendet den RBO (Rule-based Optimizer), da noch keine Statistiken zu den Tabellen erzeugt wurden.

Die Ausführungszeit ist ca. 52 Sek. (Abhängig vom Rechner und der aktuellen Systembelastung)

(Hinweis: Im untersten Fenster des SQL Explain-Plan finden Sie jeweils eine Beschreibung zu den einzelnen Schritten des Ausführungspfades)

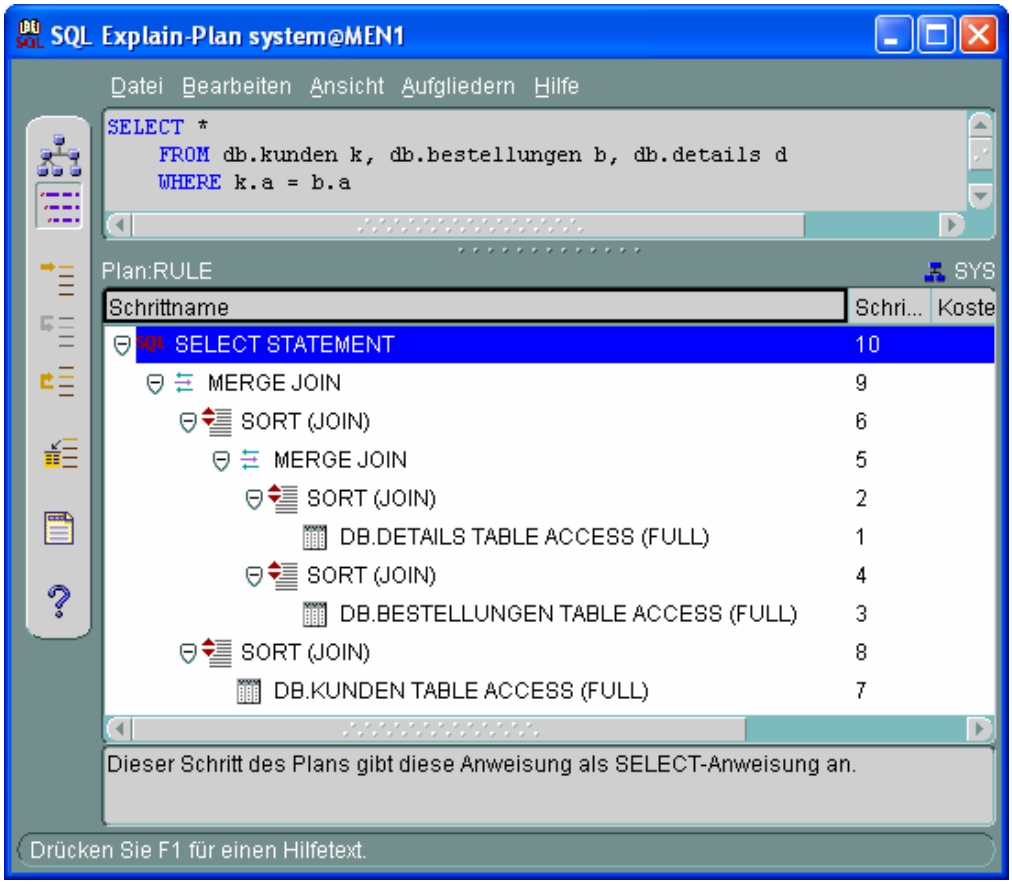

Abbildung 10 Lösung zu Aufgabe 2

Lösung Aufgabe 3

Die Ausführungszeit ist ca. 64 Sek., d.h. grösser als bei der identischen Query aus Aufgabe 3!

Wie erklären Sie sich diesen Unterschied?

Der RBO hat keine Angabe über die Grösse der Tabellen und bestimmt deshalb die Reihenfolge der Schritte an Hand von Regeln (engl. Rules, darum RBO: rule-based optimizer)

Vergleichen Sie die Reihenfolge der Schritte von Aufgabe 2 und Aufgabe 3. Was stellen Sie fest?

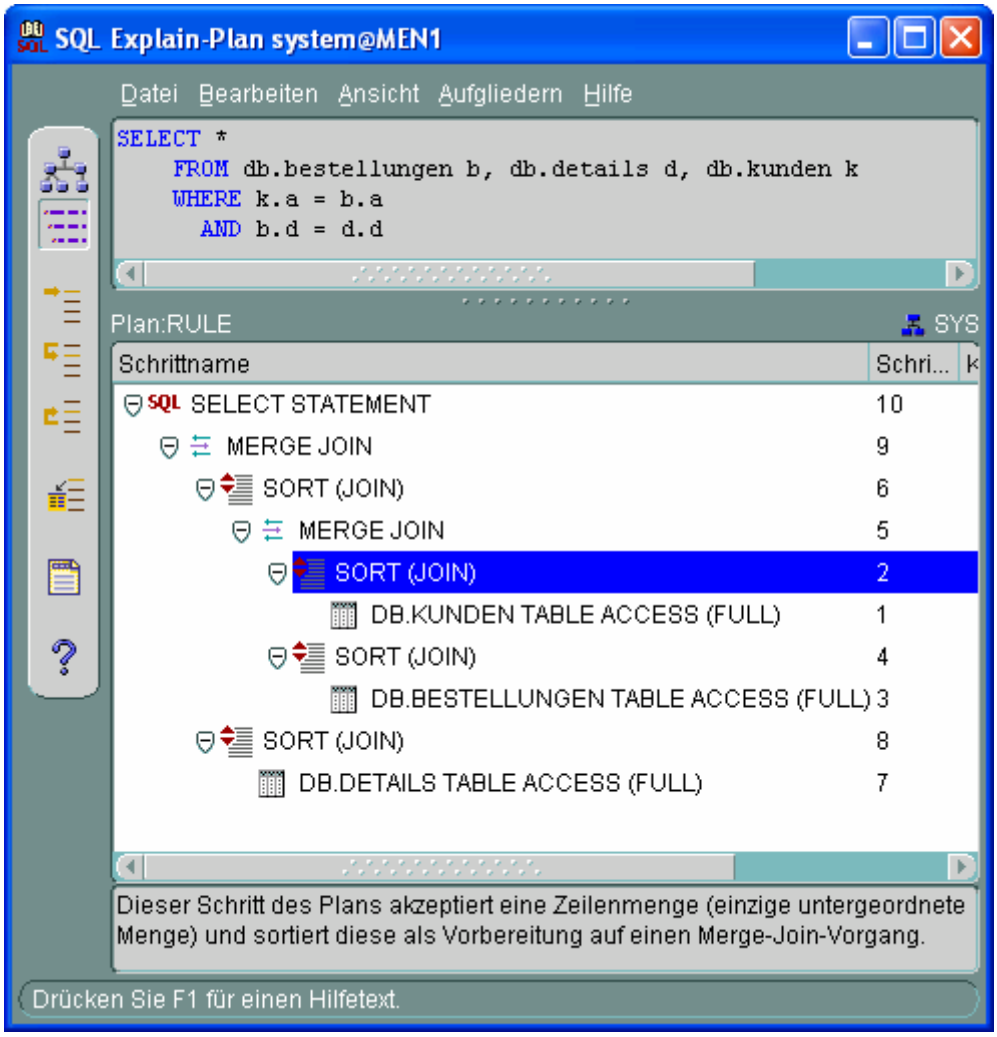

Abbildung 11 Lösung zu Aufgabe 3

Lösung Aufgabe 4

Nachdem die drei Tabellen analysiert wurden, verwendet Oracle den CBO. (Im Explain-Plan werden jetzt auch die zu erwartenden *Kosten*, *Zeilen* und *Kilobytes* angezeigt.)

Die Ausführungszeit ist ca.5 Sek. Gegenüber der vorherigen Aufgabe mit dem RBO ist das eine Verbesserung **von Faktor 10!!**

Oracle empfiehlt den CBO zu benutzen. Dazu müssen Sie aber aktuelle Statistiken haben. Am Besten lassen Sie Ihre Datenbank mit einem Skript automatisch analysieren; z.B. jedes Wochenende oder in jeder Nacht.

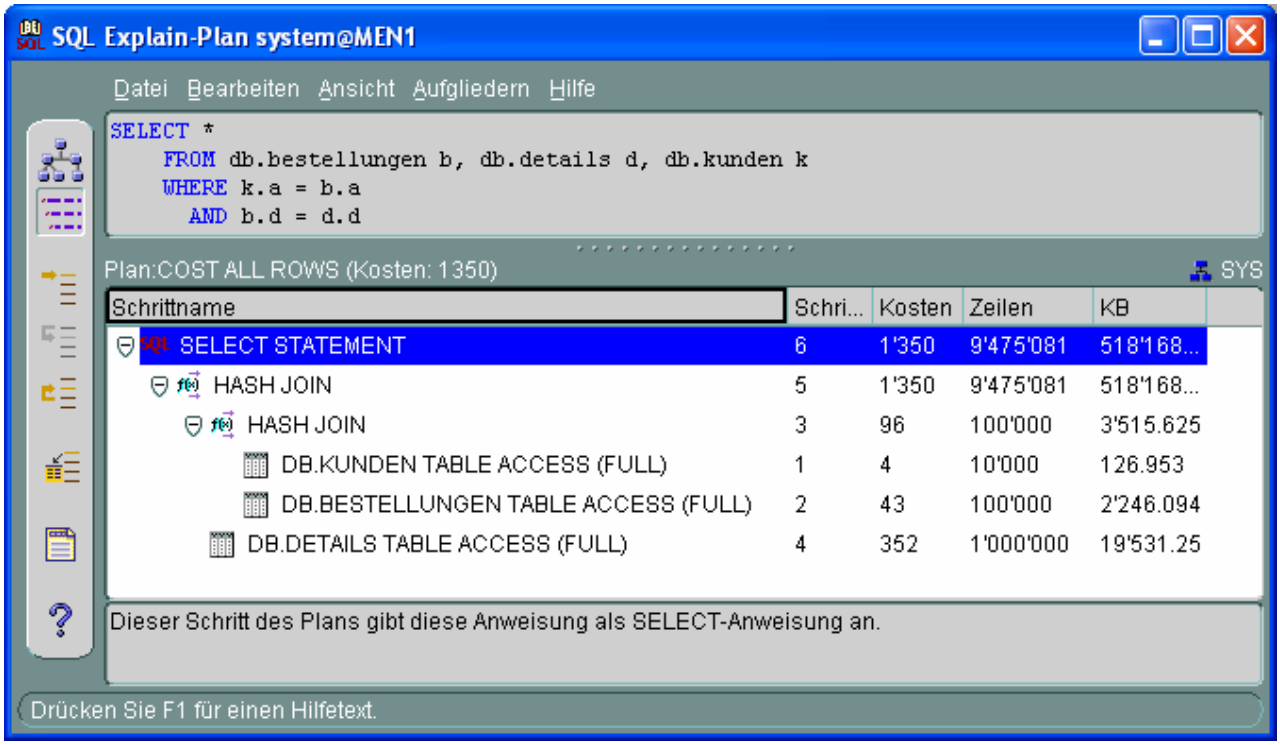

Abbildung 12 Lösung zu Aufgabe 4

Lösung Aufgabe 5

Der Optimizer braucht die Indices **nicht**, obwohl wir sie extra angelegt haben! Wieso nicht?

Indices werden von Oracle nur gebraucht um an Daten zu gelangen, wenn sie *weniger als 20%* der Tabellendaten zurückliefern. Sobald der Optimizer anhand der Statistiken berechnet hat, dass er auf mehr als 20% der Tabellendaten zugreifen muss, wählt er einen Tablescan.

Der Explain-Plan und damit auch die Ausführungszeit sind gleich wie in Aufgabe 4.

Lösung Aufgabe 6

Der Hint weist Oracle an, anstelle von Tablescans die Indices in der angegebenen Reihenfolge zu benutzen und die zurückgegebenen Datensätze mit Nested-Loop zu verknüpfen.

Die Ausführungszeit beträgt weniger als 1 Sekunde! Wie Sie sehen, ist es möglich mit Hints die Abfrage bedeutend zu beschleunigen.

Nested-Loop: Dieser Schritt des Plans verknüpft zwei Zeilenmengen, indem eine Iteration über der steuernden oder äusseren Zeilenmenge (der ersten untergeordneten Menge des Joins) durchgeführt wird, und indem für jede Zeile die Schritte der inneren Zeilenmenge (der zweiten untergeordneten Menge) ausgeführt werden. Entsprechende Zeilenpaare werden gegen die Join-Bedingung getestet, die in der WHERE-Klausel der Abfrage angegeben wird. (Text übernommen aus der OEM Online-Hilfe)

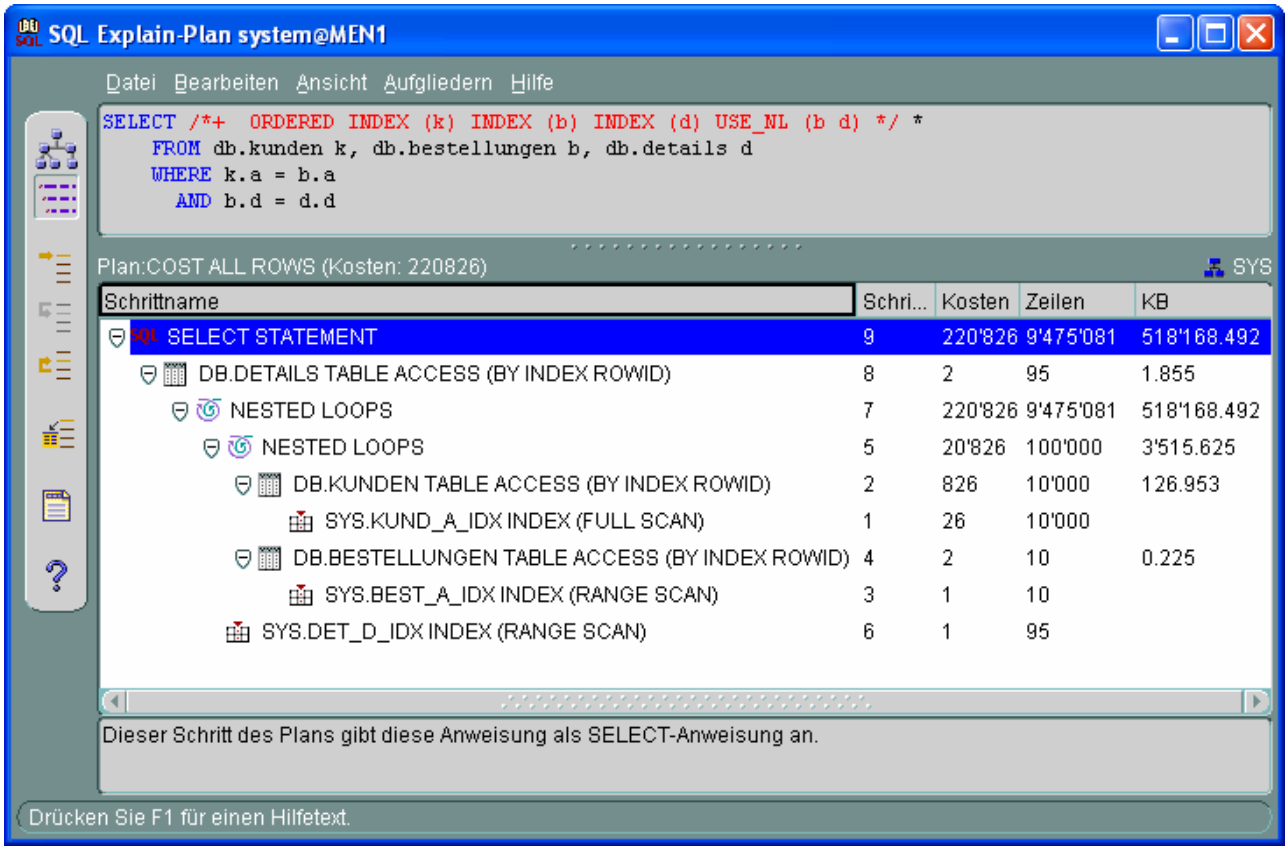

Abbildung 13 Lösung zu Aufgabe 6

Lösung Aufgabe 7

Der RBO greift auf die Indices zu.

Die Ausführungszeit liegt bei ca. 1.2 Sek. Das ist die gleiche Grössenordnung wie bei der von Hand optimierten Query aus der vorhergegangen Aufgabe.

Dieses Ergebnis bedeutet aber nicht, dass der RBO besser ist als der CBO! Im Allgemeinen erzielt der CBO die besseren Resultate. Oracle empfiehlt deshalb, den CBO zu benutzen.

Was bedeutet das nun für Sie als Entwicklerin oder Datenbankadministrator?

Identifizieren Sie die Abfragen, welche zu lange Ausführungszeiten haben. Studieren Sie den Explain-Plan. Werden die Indices verwendet wie geplant? Fehlen Indices? Fügen sie zusätzliche Indices ein. Probieren Sie den RBO und CBO aus. Welcher liefert die besseren Resultate?

Falls Sie die Applikation und die Daten in den Tabellen **gut** kennen, können Sie im Allgemeinen durch Tuning massive Performanceverbesserungen erzielen!

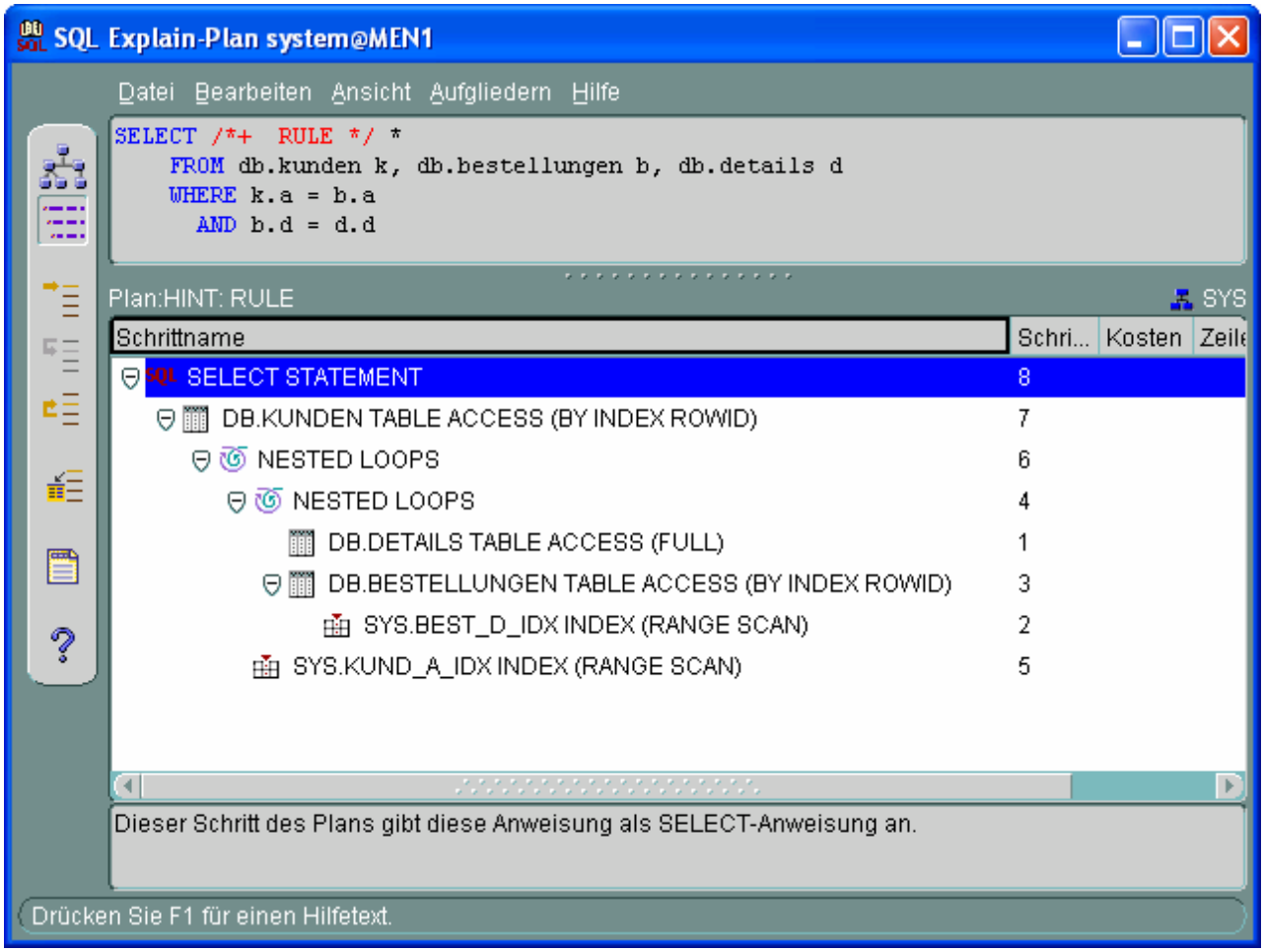

Abbildung 14 Lösung zu Aufgabe 7

4 Anhang A: Quellenverzeichnis

[Ault]

Ault Michael: *Oracle-Datenbanken Administration und Management*, Bonn 2003, 1.Auflage (MITP) ISBN 3-8266-1316-3

[Buff]

Buff Hanswalter: *Datenbanktheorie*, Zürich 2003 (Norderstedt), ISBN 3-0344-0201-5

[Loney]

Loney Kevin, Theriault Marlene: *Oracle 9i DBA Handbook*, Emeyville 2002, Oracle Press, (McGraw-Hill), ISBN 0-07-219374-3

Folgendes Oracle Manual ist dem Leitprogramm beigelegt:

[Green]

Green Connie Dialeris: Oracle9 *i Database Performance Tuning Guide and Reference*, Release 2 (9.2) 2002, Part No. A96533-02

5 Anhang B: Skripts und Daten für die Studierenden

Für die Aufgaben in Kapitel 2 werden folgende Dateien gebraucht:

`CREATE_TABLES_DB.txt`	Script zum erzeugen der Tabellen
`KUNDEN.tabtxt`	Zufällig erzeugte Kundendaten
`BESTELLUNGEN.tabtxt`	Zufällig erzeugte Bestelldaten
`DETAILS.tabtxt`	Zufällig erzeugte Details zu Bestellungen
`ora_load_Kunden.ctl`	Controlfile zum Laden der Kundendaten
`ora_load_Bestellungen.ctl`	Controlfile zum Laden der Bestelldaten
`ora_load_Details.ctl`	Controlfile zum Laden der Detaildaten

Oracle Manual zum Datenbanktuning:

`Performance Tuning.pdf`	Database Performance Tuning Guide → [Green]