10 Gründe warum Ihr Index nicht verwendet wird

Marco Patzwahl
MuniQSoft GmbH
München-Unterhaching

Schlüsselworte

Index Benutzung, Index Tuning

Einleitung

Ein Index auf einer Tabelle sollte ja eigentlich den Zugriff auf die Tabelle beschleunigen. Was macht man jedoch, wenn der Index zwar vorhanden ist, aber nicht verwendet wird. Der Vortrag soll Ihnen einen Leitfaden an die Hand geben, diese Probleme systematisch zu lösen.

Index Tuning Grundlagenregel

Nicht jeder Ausführungsplan wird durch Index Zugriff anstelle eines Full Table Scans schneller

Das einzige, was zählt ist die Ausführungszeit!

Fall 1: Index ist unusable

Sie haben einen Index, nur ist der leider unbrauchbar

Beispiel:

```
ALTER TABLE emp MOVE;

SELECT 1 FROM emp
WHERE empno=7369;
```

OPERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			3
TABLE ACCESS	EMP	FULL	3
Filter Predicates			
EMPNO=7369			

Workaround:

```
ALTER INDEX pk_emp REBUILD;

SELECT 1 FROM emp WHERE empno=7369;
```

OPERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			1
TABLE ACCESS	EMP	BY INDEX ROWID	1
INDEX	PK_EMP	UNIQUE SCAN	0
Access Predicates			
EMPNO=7369			

Fall 2: Es gibt keine Statistiken

Sie haben keine aktuellen Statistiken

Workaround:

```
BEGIN
dbms_stats.gather_index_stats('<OWNER>','<IND_NAME>');
```

```
END;
/
```

Fall 3: Es gibt keine (korrekten) Histogramme

Die Werte in der indizierten Spalte treten ganz unterschiedlich oft auf

Es liegen keine korrekten Histogramminformationen vor

Ermittlung der Verteilungsstatistiken

```
BEGIN
dbms_stats.gather_table_stats(
'<OWNER>','<TAB_NAME>', method_opt=>
  'FOR ALL INDEXED COLUMNS SIZE AUTO);
END;
/
```

Statistiken dazu stehen in `USER_HISTOGRAMS`

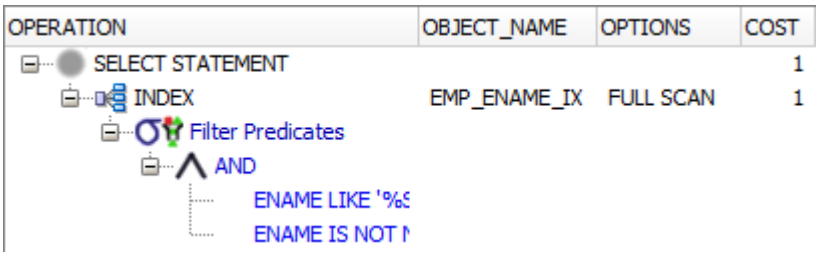
`GATHER_PLAN_STATISTICS`

Fall 4a: Abfrage verwendet LIKE

Sie haben die **VARCHAR2** Spalte (ename) indiziert

```
SELECT * FROM emp
WHERE ename like 'S%'; -- OK

SELECT * FROM emp
WHERE ename like '%S%';
-- 11.2.0.3 OK
```



Workaround für älter Versionen

```
SELECT /*+ INDEX (emp) */ * FROM emp
WHERE ename like '%S%';
```

Fall 4b: Abfrage verwendet LIKE

Sie haben die **NUMBER** Spalte (sal) indiziert

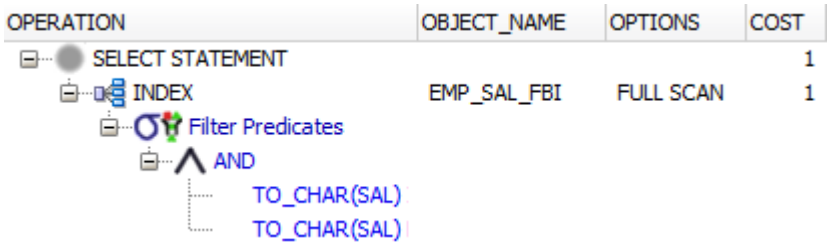
```
SELECT * FROM emp
WHERE sal like '%3%'; -- Geht nicht
```

Workaround:

Function Based Index anlegen

```
CREATE INDEX emp_sal_fbi ON
EMP(to_char(sal));
```

```
select 1 from scott.emp
where sal like '%3%'
```



Fall 4c: Abfrage verwendet LIKE

Sie haben die **DATE** Spalte (hiredate) indiziert

```
SELECT * FROM emp
WHERE hiredate like '%1980%'; -- Geht nicht
```

Workaround:

Function Based Index anlegen

```
CREATE INDEX emp_hiredate_fbi ON
EMP(to_char(hiredate,'DDMMYYYY'));
```

```
select /*+ INDEX(emp) */ *
from scott.emp
where hiredate like '%1980%'
```

Fall 5: Es wird der falsche Index verwendet

Sie haben mehrere Indizes auf der Tabelle, aber der falsche wird verwendet

Workaround:

Index Hint mit richtigem Index verwenden

```
SELECT /*+ INDEX(emp,mein_index) */ *
FROM emp
WHERE sal=3000 and comm=1000;
```

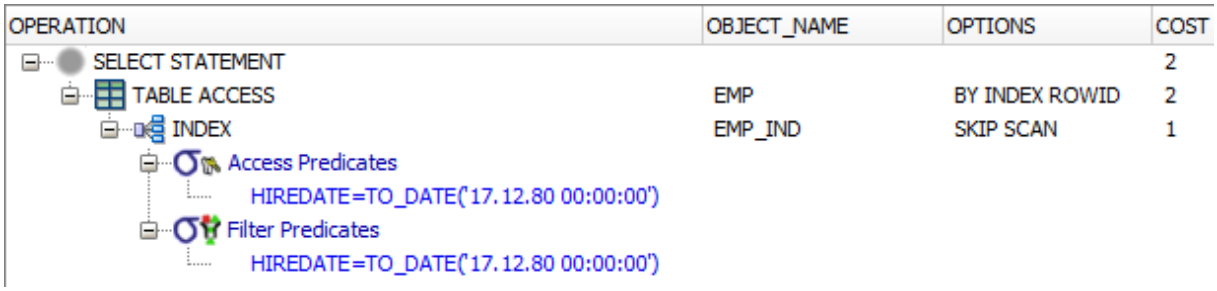
Fall 6: Falsche Reihenfolge indiziert

Sie verwenden eine nicht führende Index-Spalte
(Index auf ename, sal, hiredate)

Workaround:

Reihenfolge ändern oder Index Skip Scan Hint verwenden (Ab 11g kann der Optimizer das automatisch)

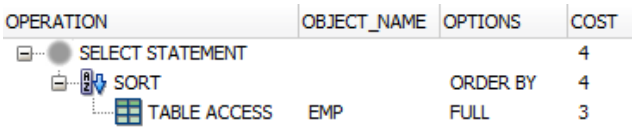
```
SELECT /*+ INDEX_SS(emp) */ FROM emp
WHERE hiredate=to_date('17.12.80 00:00:00');
```



Fall 7: Sortierung soll über Index gehen

Index liegt auf SAL

```
SELECT * FROM emp
ORDER BY sal;
```



Fall 8: Stored Outline / SQL Profil

Es gibt eine Stored Outline oder ein SQL-Profil, das der Optimizer vorzieht

Workaround:

Prüfen Sie, ob Ihre Query als Stored Outline oder SQL Profil existiert

```
SELECT * FROM dba_outlines;
```

```
SELECT * FROM dba_sql_profiles;
```

Löschen Sie ggf. die Stored Outline/ SQL Profil

```
BEGIN DBMS_SQLTUNE.DROP_SQL_PROFILE(name => 'mein_sql_profile');
END;
```

```
DROP OUTLINE <meine_outline>;
```

Fall 9: Initialisierungsparameter

Sie haben Initialisierungsparameter verwendet, die eine Index-Verwendung für den Optimizer ungünstig erscheinen lassen.

Beispiel:

```
db_file_multiblock_read_count (hoch)
```

```
optimizer_index_cost_adj (hoch (>100))
```

```
optimizer_index_caching (niedrig)
```

Fall 10: Sie haben gar keinen Index

Spalte in der WHERE Bedingung ist gar nicht indiziert

Workaround:

Spalte indizieren

```
CREATE INDEX scott.my_ind ON scott.emp(sal)
NOLOGGING ONLINE;
```

Bonustrack: Index Grundlagen

CREATE INDEX Kommando

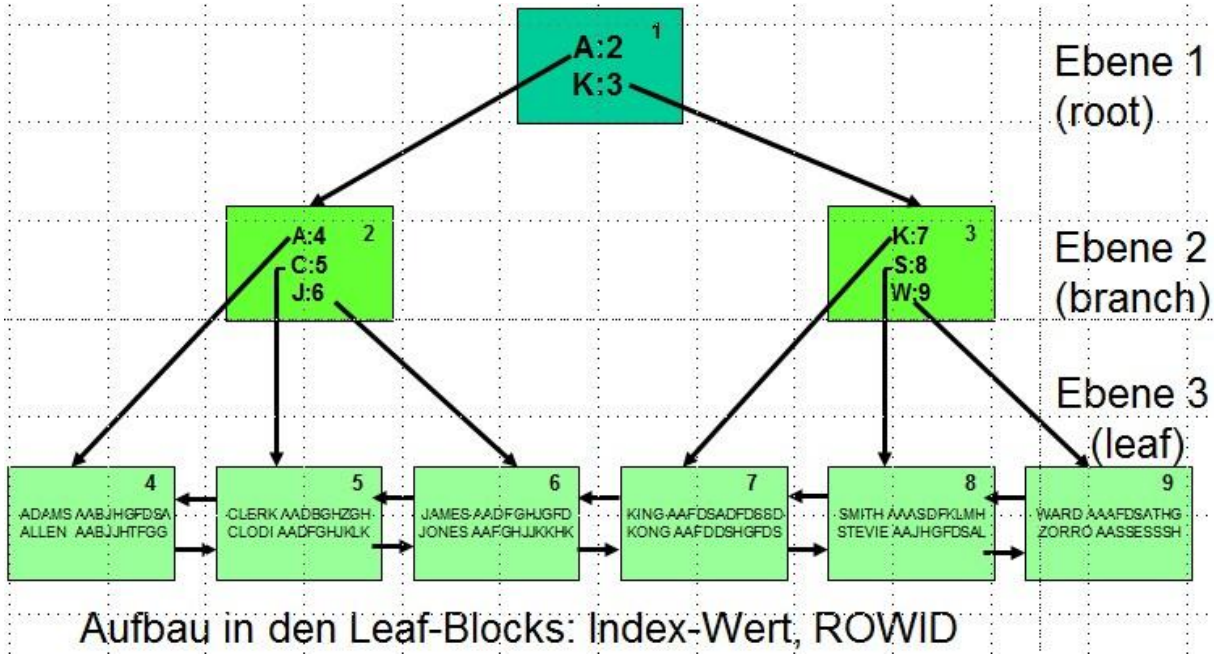
Ein Index kann auf jede beliebige Spalte einer Tabelle (Ausnahme Long und Lob) gesetzt werden

Einspaltige oder mehrspaltige Indizes sind möglich

Syntax:

```
CREATE [UNIQUE] INDEX [<owner>.]<index_name>
ON [<owner>.]<tab_name>
(<spalten_name> [ASC|DESC],
<spalten_name> [ASC|DESC], ...)
[LOGGING|NOLOGGING] [ONLINE]
[TABLESPACE <tablespace_name>]
[COMPRESS <int> | NOCOMPRESS]
[NOSORT]
[PARALLEL <int> |NOPARALLEL]
[PCTFREE <int>] [INITRANS <int>];
```

B-Tree Index Aufbau



Was ist Ihr Index Problem?

Index zu groß ?

Lösung: Prüfen durch DBA_SEGMENTS, DBA_INDEXES, SEGMENT_ADVISOR

Zu viele Indizes ?

Lösung: Prüfen durch DBA_IND_COLUMNS

Index wird nicht benutzt?

Lösung: Prüfen durch Index Monitoring, `V$SEGMENT_STATISTICS, COL_USAGE$`

Zu wenige Indizes ?

Lösung: Prüfen durch `COL_USAGE$`

Index Tuning Tipp: Index Only Access

Hier wird die dazugehörige Tabelle nicht mehr benötigt, weil die gesuchten Daten sich alle im Index befinden

```
CREATE TABLE person (
 id NUMBER PRIMARY KEY
 vorname VARCHAR2(200),
 nachname VARCHAR2(200));

CREATE INDEX person_ix
ON person(nachname);

SELECT nachname FROM person
WHERE nachname like 'A%'
```

Weitere Index-Tuning Tipps:

Index wird bei Primary Key / Unique Key automatisch verwendet

Es ist günstiger, zuerst den Unique Index und dann den Constraint auf diese Spalte anzulegen

```
CREATE TABLE t (id NUMBER);

CREATE UNIQUE INDEX i ON t(id)
TABLESPACE indx_tbs NOLOGGING;

ALTER TABLE t ADD CONSTRAINT pk PRIMARY KEY(id);
```

Alternativ kann wenigstens der Tablespace des Index bei Erzeugen des Constraints angegeben werden:

```
CREATE TABLE t (id NUMBER CONSTRAINT pk
PRIMARY KEY USING INDEX TABLESPACE ts_idx);
```

Index Tuning durch Selektion

Entscheidung, welche Werte alleinig indiziert werden sollen

Nur nach diesen Werten kann dann natürlich gesucht werden

Beispiel:
Nur Münchner Telefonnummern sollen indiziert werden:

```
CREATE INDEX marco.brd_tel_ix ON marco.brd(
CASE WHEN vorwahl='089' THEN vorwahl ELSE NULL END);
```

Originalgröße des Index (ohne Selektion): 2GB

Größe des Index (mit Selektion): 16MB

SELECT dazu (der den Index dann auch benutzt):

```
SELECT * FROM marco.brd WHERE (CASE WHEN vorwahl='089' THEN vorwahl
ELSE NULL END)='089'
```

ONLINE Option (nur EE, PE)

Wird der Index mit der ONLINE Option angelegt, kann während der Index-Erstellung auf der dazugehörigen Tabelle (schreibend) weitergearbeitet werden

Auch bei einem Index Rebuild kann mit der ONLINE Option die Tabelle parallel verändert werden

ONLINE Option steht nur in der Enterprise Edition zur Verfügung

Index Reorg

Der Platz in den Index-Blöcken wird nach einem Delete u.U. nicht mehr verwendet

Indizes wachsen nach vielen Deletes und anschließenden Inserts stark an und sollten häufiger reorganisiert werden

```
ANALYZE INDEX <owner>.<indx> VALIDATE STRUCTURE;

SELECT name, del_lf_rows, lf_rows - del_lf_rows
       lf_rows_used, to_char(del_lf_rows /
      (lf_rows)*100,'999.99999') ratio,
       OPT_CMPR_COUNT,OPT_CMPR_PCTSAVE
FROM index_stats where name = upper('<indx>');
```

Wenn die Anzahl der gelöschten Zeilen 10-15 % beträgt, sollte der Index reorganisiert werden (Metalink Note 30405.1):

```
ALTER INDEX <owner>.<indx> REBUILD ONLINE;
```

Eine andere Möglichkeit ist, das Package dbms_space anzuwenden

Das Package kann die Freelist der Index-Blöcke auslesen und damit ihren Füllpegel in den folgenden Bereichen bestimmen:

<25%

<=50%

<=75%

<=100%

Pipelined Funktion Skript beim Referenten erhältlich.

Möglichkeiten während der Reorganisation eines Index

ALTER INDEX <owner>.<ind>	Bemerkung
REBUILD ONLINE	ONLINE nur bei Enterprise Edition oder PE!
PCTFREE 0	Wenn keine vergrößernden Updates oder Inserts!!! in der Tabelle mehr stattfinden = 0 !
COMPUTE STATISTICS	Index-Segment Statistiken gleich mitberechnen (aber mittels ANALYZE INDEX => Desupported)
COMPRESS 2	Anzahl der führenden Spalten (hier 2 Spalten), die komprimiert werden sollen
TABLESPACE USERS	Index kann bei Reorg auch auf einen anderen TBS verschoben werden
NOLOGGING	Mitprotokollieren des Reorgs in Redologs verhindern

Index Statistiken

Sie sollten Index Statistiken nur mit dem Package DBMS_STATS (.GATHER_INDEX_STATS, .GATHER_SCHEMA_STATS oder .GATHER_DATABASE_STATS) erzeugen

Die folgenden Aufrufe werden nicht mehr empfohlen:

```
ANALYZE INDEX idx COMPUTE STATISTICS;

ANALYZE INDEX idx ESTIMATE STATISTICS …;

CREATE INDEX idx … COMPUTE STATISTICS;

ALTER INDEX idx … REBUILD COMPUTE STATISTICS;
```

Bereits seit 10g werden Statistiken bei einem CREATE INDEX oder ALTER INDEX Befehl automatisch erzeugt!

Nachteile der Index Reorg

Index ist "perfekt" ausbalanciert, weist aber keine Lücken für neue Einträge auf

Neue Spaltenwerte führen zu einem Index Block Split

50/50 Block Split (die Hälfte der Einträge des "alten" Blocks werden in den "neuen" Block kopiert)
90/10 (bzw. 99/1) Block Split (wenn der höchste Wert im Index überschritten wurde)

Index-Tuning durch Spaltensektion

Gedrehte Indizes versuchen die Reihenfolge der indizierten Spalten (c1, c2, c3) zu ändern.
Statt c1, c2, c3 → c3, c2, c1)

```
CREATE INDEX i ON t (c1, c2, c3) COMPRESS 2;

CREATE INDEX i ON t (c3, c2, c1) COMPRESS 2;
```

Achten Sie jedoch darauf, dass nur nach bestimmten Spalten gesucht werden kann

Ausnahme: Index Skip Scan (hier werden führende Index-Spalten übersprungen)

Index Hints (Auswahl)

Positiv Hint	Negativ Hint	Beschreibung
INDEX	NO_INDEX	Index soll (nicht) verwendet werden
INDEX_ASC	NO_INDEX	Index soll aufsteigend verwendet werden
INDEX_DESC	NO_INDEX	Index soll absteigend verwendet werden
INDEX_FFS	NO_INDEX_FFS	Fast Full Scan für Index (nicht) durchführen

INDEX_SS	NO_INDEX_SS	Index Skip Scan (nicht) durchführen
INDEX_SS_ASC	NO_INDEX_SS	Index Skip Scan aufsteigend (nicht) durchführen
INDEX_SS_DESC	NO_INDEX_SS	Index Skip Scan absteigend (nicht) durchführen
INDEX_COMBINE		Mehrere Indizes in Kombination verwenden
INDEX_JOIN		Zwei Indizes miteinander Joinen, Verknüpfungsspalte ist die ROWID
PARALLEL_INDEX	NO_PARALLEL_INDEX	Indexabfrage (nicht) parallel durchführen
INDEX-RRS		Parallel Index Fast Full Scan (undokumentiert)

Kontaktadresse:

Marco Patzwahl

MuniQSoft GmbH

Grünwalder Weg 13 a

D-82008 Unterhaching

Telefon: +49 (0) 89-67 90 90 40

Fax: +49 (0) 89-67 90 90 50

E-Mail marco.patzwahl@muniqsoft.de

Internet: www.muniqsoft.de