DOAG Nürnberg 2010

Oracle
Index Tuning & Administration

Marco Patzwahl

MuniQSoft GmbH

- Gegründet 1998
- Tätigkeitsbereiche:
  - Oracle IT Consulting & Services
  - Oracle Schulungen (SQL, PL/SQL, DBA, APEX, B&R, …)
  - Software-Lösungen
  - Oracle Lizenzen

MuniQSoft GmbH
Grünwalder Weg 13 a
D-82008 Unterhaching
www.muniqsoft.de
+49 89 6228 6789-0

Was ist ein Index?

- Ein Index indiziert eine / mehrere Spalten einer Tabelle
- Bei Abfragen auf indizierte Spalten werden die dazu passenden Zeilen schneller gefunden
- Bei Inserts/Updates/Deletes auf indizierten Spalten wird die Transaktion jedoch verlangsamt!
- Dieser Vortrag behandelt folgende Index-Typen nicht:
  - XML Index
  - Context Index
  - Spatial Index
  - Partitionierter Index

Index Vorteile / Nachteile

- u.U. schneller Zugriff bei Select, Update und Delete
- Sortierungen schneller
- COUNT(*) schneller bei Bitmap Index oder indizierten NOT NULL Spalten

- Längere Verarbeitungszeit bei Insert/Update/Delete
- Höherer Pflegeaufwand durch Index (Ressourcenverbrauch)
- Index-Blöcke können Cache dicht machen (und Tabellen-Blöcke aus dem Cache vertreiben)

CREATE INDEX Kommando

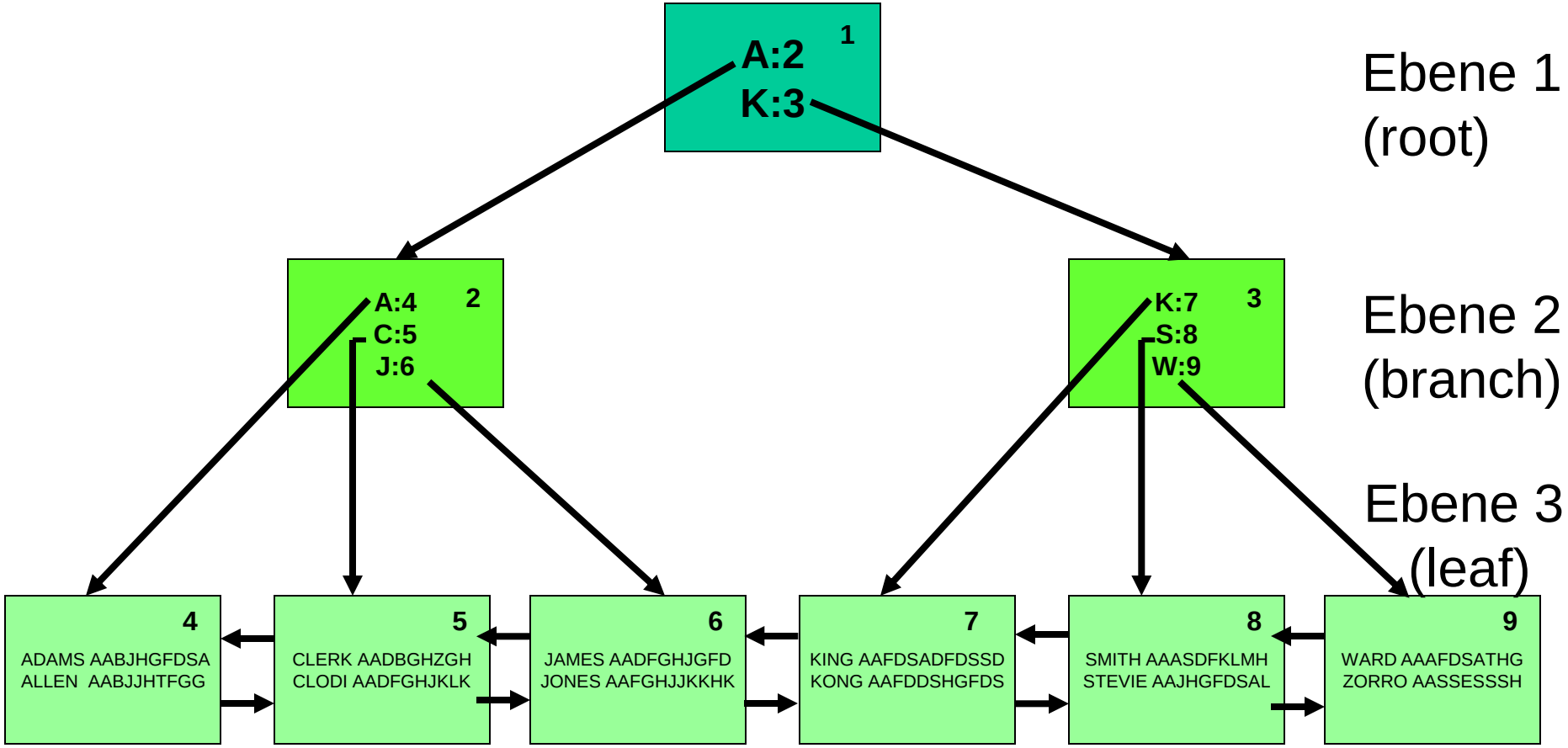
- Ein Index kann auf jede beliebige Spalte einer Tabelle (Ausnahme Long und Lob) gesetzt werden
- Einspaltige oder mehrspaltige Indizes sind möglich
- Syntax:

```
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>];
```

Besonderheiten im Vergleich zur Tabelle

- **Es gibt keinen Default Index Tablespace. Ein Index landet (ohne Angabe) im Default Tablespace des Benutzers**
- **PCTFREE hält bei Indizes den Platz für Updates und INSERTS frei!**
- **Es gibt keinen internen Update auf einen Index-Wert. Stattdessen wird der Wert gelöscht und neu eingefügt**
- **Ein Indexeintrag darf maximal 78% eines Blocks betragen sonst bekommt man den Fehler (8k Blockgröße)**
  - `SQL-Fehler: ORA-01450: Maximale Schlüssellänge (6398) überschritten`
- **Maximal 32 Spalten (Bitmap Index: 30) lassen sich von einem Index erfassen**

B-Tree Index Aufbau



Aufbau in den Leaf-Blocks: Index-Wert, ROWID

Index Tablespace

- **Nach Möglichkeit sollten ein oder mehrere Tablespaces dediziert für Indizes verwendet werden**
- **Vorteile:**
  - **Saubere Trennung von Tabellen und Indizes**
  - **Im Crashfall des Tablespace u.U. Neuaufbau der Indizes mit Skripten**
  - **Bei geeigneter Hardware: Eigene Platten für Indizes (Ideal: Solid State Disks)**

Bitmap Index

- **Voraussetzungen:**
  - **Enterprise oder Personal Edition**
  - **Bei Tabellen mit geringer Insert und Update Aktivität auf indizierte Spalten (Data Warehouse)**
- **Vorteile:**
  - **Bei größeren Datenmengen mit geringer Kardinalität (z. B. 80 Mio. Einwohner, Bitmap Index auf Spalte Bundesländer (16))**
  - **Schneller, wenn Abfragen aus mehreren WHERE- Bedingungen bestehen, die mit OR oder AND verknüpft sind**
  - **Bei Read-Only Tabellen schneller**
  - **Benötigen weniger Speicherplatz als B*Indizes, wenn Kardinalität klein (teilweise nur 2% von B*Indizes)**
  - **Count Funktion ist schneller**

Bitmap Index (f)

- **Syntax:**
  - ```
    CREATE BITMAP INDEX [<owner>.]<index_name>
      ON [<owner>.]<tab_name>
      (<col_name> [ASC|DESC], <col_name> [ASC|DESC] ...)
      [TABLESPACE <tbs_name>];
    ```
- **Nachteile:**
  - **INSERT, UPDATE und DELETE dauern länger. Da beim Ändern eines Wertes alle Zeilen mit diesem Wert gesperrt werden, kann es leichter zu Deadlocks kommen**
  - **Index-Werte werden komprimiert gespeichert. Bei Änderungen muss zuerst dekomprimiert, danach manipuliert, der Index-Zweig neu aufgebaut und wieder komprimiert werden**
- **Hinweis:**
  - **NULL Werte werden indiziert**
    ```

Bitmap Index (ff)

- **Beispiel:**
  - ```
    CREATE TABLE bm_test AS SELECT rownum
    nr,decode(floor(DBMS_RANDOM.value(0,4)),0,
    'Nord',1,'SÜd',2,'Ost',3,'West') region
     FROM dba_objects; -- auf 1 Mio Zeilen
    ```
  - ```
    CREATE BITMAP INDEX bm_ind
    ON bm_test(region) TABLESPACE indx;
    ```
  - ```
    EXEC dbms_stats.gather_index_stats(
    ownname=>'SCOTT',indname=>'BM_IND');
    ```
  - ```
    SELECT blevel, leaf_blocks, distinct_keys,
           avg_leaf_blocks_per_key FROM DBA_INDEXES
    WHERE index_name='BM_IND';
    => 1,170,4,42     (Bitmap Index)
    => 2,4440,4,1110 (Für Nonuniqe Index, >Faktor 26 )
    ```

Reverse Key Index

- **Beschreibung**
  - **Ein Reverse Key Index indiziert die Werte rückwärts**
  - **So wird z.B. aus 123 der Wert 321**
- **Vorteile:**
  - **Bei Primary Key Erzeugung durch Sequenzen sind Inserts schneller**
  - **Verteilt die Index-Einträge auf verschiedene Index-Blöcke → weniger Sperrkonflikte**
  - **Vorteilhaft in einer RAC-Umgebung**
- **Nachteile:**
  - **Keine Index Range Scans möglich**
- **Beispiele:**

```
CREATE INDEX "TELE"."BRD_REV_IDX"
ON "TELE"."BRDP"  ("PLZ")
TABLESPACE "BRD_IND_TBS" REVERSE;
```

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%'
    ```
    ```

Index-Tuning

- 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

- **Sie können entscheiden, 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'
    ```

COMPRESS Option

- **Mit der COMPRESS Option können führende Index-Spalten, die viele gleiche Werte besitzen, platzsparend im Index untergebracht werden**
  - ```
    CREATE INDEX scott.i
    ON scott.emp(deptno,job) COMPRESS 2;
    ```
- **Sie können vom bestehenden Index eine Analyse erzeugen, um zu sehen, wie viel Prozent ein COMPRESS einsparen würde**
  - ```
    ANALYZE INDEX scott.i VALIDATE STRUCTURE;
    ```
  - ```
    SELECT opt_cmpr_count,opt_cmpr_pctsave
    FROM index_stats;
    ```
    ```

Nutzen der Key Compression

- **Hier werden gleiche Index-Einträge komprimiert.**
  - `CREATE TABLE t (x VARCHAR2(100),y VARCHAR2(100));`
  - `INSERT INTO t SELECT object_type,status FROM dba_objects;`
  - `CREATE INDEX i ON t(x,y) COMPRESS 2;`
- **Folgende Indizes wurden erzeugt:**
  - **i(x,y)** — **100 Leaf Blocks**
  - **i(x)+i(y)** — **77+66 = 143 Leaf Blocks**
  - **i(x,y) compress 2** — **43 Leaf Blocks**
  - **i(x,y) compress 1** — **67 Leaf Blocks**

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**
- **Hinweis:**
  - **ONLINE Option steht nur in der Enterprise Edition zur Verfügung**

NOLOGGING Option

- Durch NOLOGGING wird beim Aufbau bzw. Reorg des Index VIEL weniger in den Redolog-Dateien gespeichert
- Vorteil
  - Schnelle Durchführung des Befehls
- Nachteil
  - Bis zum nächsten Backup des Index-Tablespace kann der Index bei einem Crash mit Hilfe der Redologs nicht mehr aufgebaut werden (eigene Skripten notwendig)

Index Größe abschätzen

- Mit folgendem Skript lässt sich die Größe (hier in MB) abschätzen, die ein Index bei einer Erstellung benötigen würde
- Hinweis: Compress wird bei Indexberechnung nicht berücksichtigt
-

```
DECLARE
 ub  NUMBER;
 ab  NUMBER;
 s CLOB;
BEGIN
  s:='CREATE INDEX x.x ON scott.bug ("TEXTID", "LOAD", "GEO")
  PCTFREE 30 TABLESPACE USERS';
  dbms_space.create_index_cost(s, ub, ab);
  dbms_output.put_line('Used MB: ' ||round(ub/1024/1024,2);
  dbms_output.put_line('Alloc MB:' ||round(ab/1024/1024,2);
END;
/
```

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;`
    ```

Index Reorg

- 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 ☺

Index Reorg

- **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

Rebuild versus Drop & Create

- **Ein Rebuild kann einen bestehenden SORTIERTEN Index klonen**
- **Ein Drop Index & Create Index muss den Index komplett neu sortieren**

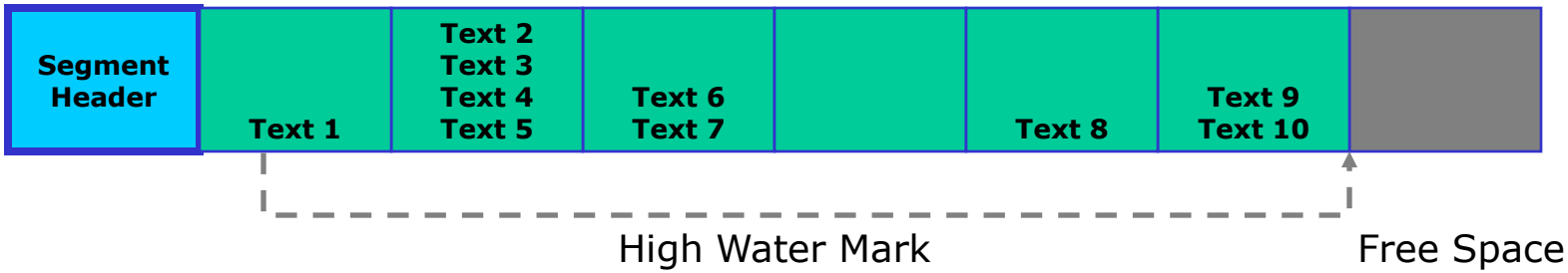
	REBUILD	DROP & CREATE
1. Test	2m30s	3m39s
2. Test	2m22s	3m26s
3. Test	2m26s	3m30s

Index Statistiken

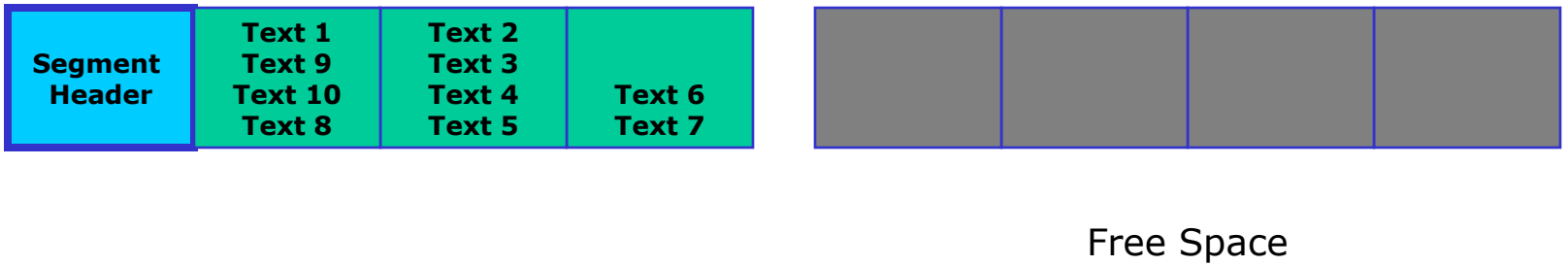
- **Sie sollten Index Statistiken nur mit dem Package DBMS_STATS (.GATHER_INDEX_STATS, .GATHER_SCHEMA_STATS oder .GATHER_DATABASE_STATS) erzeugen**
- **Die folgenden Aufrufe werden nicht mehr empfohlen:**
  - `ANALYZE INDEX idx COMPUTE STATISTICS;`
  - `ANALYZE INDEX idx ESTIMATE STATISTICS …;`
  - `CREATE 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!**

Index Online Segment Shrink (ab 10g)

- **Kann durch den Segment Advisor überprüft werden**

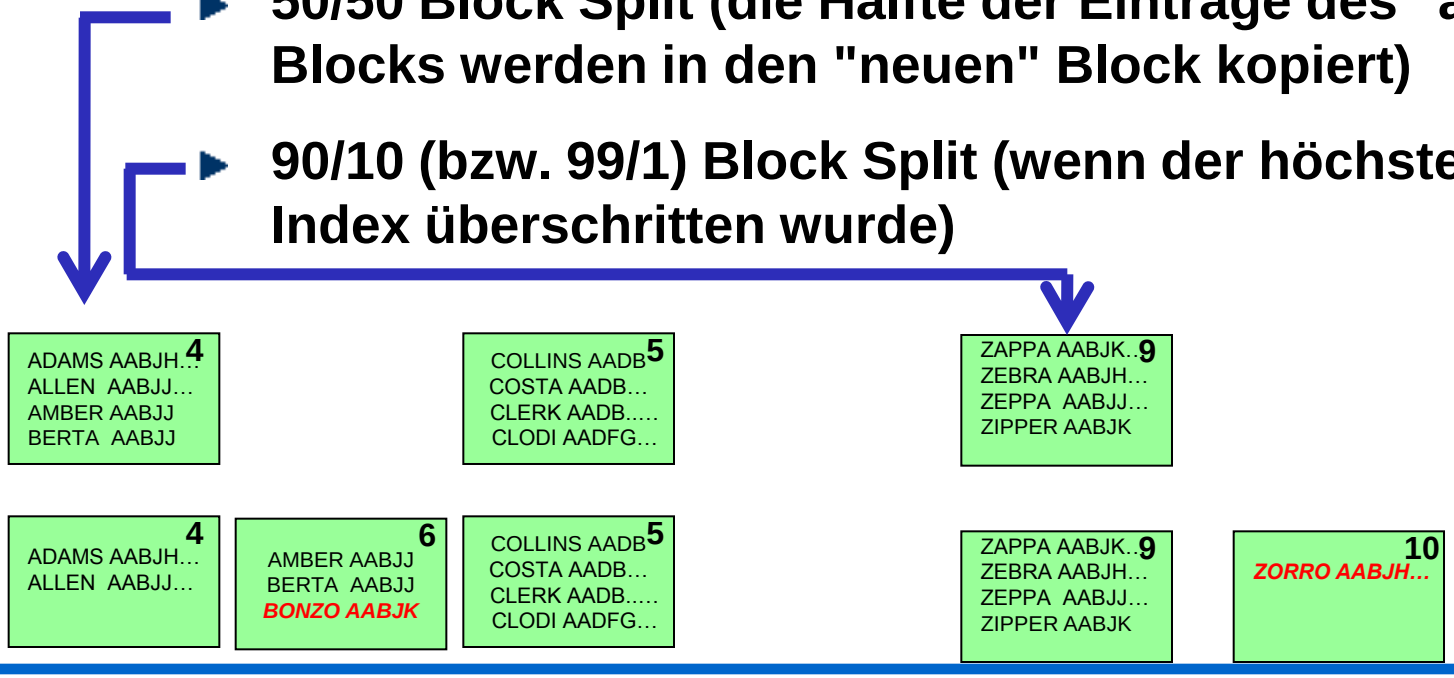


- **`ALTER INDEX scott.i SHRINK SPACE CASCADE;`**



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

- **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)
- **Bei geringer Kardinalität den Bitmap Index verwenden**
  - `CREATE BITMAP INDEX i …`

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)

Indexübersicht incl. möglicher Optionen

Indextyp Funktionalität	B-Tree Unique Index	B-Tree Non-Unique Index	Bitmap Index	Function Based Index	Reverse Key Index
Null Werte indizierbar?	Nein	Nein	Ja	Nein	Nein
Komprimierbar?	Ja	Ja	Nein	Ja	Ja
Nologging?	Ja	Ja	Ja	Ja	Ja
Parallel?	Ja	Ja	Ja	Ja	Ja
Online Reorg?	Ja	Ja	Ja	Ja	Ja
Blocksperren	Nein	Nein	Ja	Nein	Nein

Performance Messung: CREATE INDEX

NOARCHIVELOG	Non Unique (528 MB)	Unique (3 Spalten, 928 MB)	Bitmap (104 MB)	Function based
NORMAL	2m50s	4m22s	1m25s	2m24s
NOLOGGING	2m30s	4m23s	1m27s	2m29s
COMPRESS	2m36s	4m35s	na	2m31s
PARALLEL 4	2m18s	2m52s	1m14s	2m13s
ONLINE	2m50s	4m39s	1m8s	2m34s
ARCHIVELOG				
NORMAL	4m12s	5m46s	1m46s	3m09s
NOLOGGING	2m46s	4m18s	1m29s	2m22s
COMPRESS	3m20s	6m29s	na	2m47s
PARALLEL 4	4m04s	4m34s	1m30s	3m13s
ONLINE	3m45s	6m11s	1m24s	3m26s

Oracle 11.2.0.1 auf Red Hat 5,CPU: 4, MEMORY_TARGET=1G,
Tab-Größe: 2GB (27 Mio Zeilen)
STARTUP FORCE nach jeder Messung. Mittelwert aus 5 Messungen

DML Operationen mit Index

NOARCHIVELOG	UPDATE/ Rollback	DELETE/ Rollback	INSERT/ Rollback
Ohne Index	03,20s / 3,10s	03,72s / 2,49s	02,55s / 1,33s
NORMAL	11,03s / 5,66s	12,11s / 35,2s	43,00s / 1,51s
NOLOGGING	12,56s / 5,70s	11,78s / 39,8s	42,17s / 1,47s
COMPRESS	10,05s / 4,96s	11,41s / 45,55s	40,86s / 1,53s
PARALLEL 4	09,58s / 5,68s	11,34s / 44,64s	41,39s / 1,59s
ONLINE	09,64s / 5,34s	12,00s / 39,35s	41,56s / 1,44s
ARCHIVELOG	**UPDATE/ Rollback**	**DELETE/ Rollback**	**INSERT/ Rollback**
Ohne Index	02,2s / 02,21s	03,58s / 02,96s	02,27s / 1,52s
NORMAL	09,7s / 06,13s	15,07s / 34,42s	45,76s / 1,54s
NOLOGGING	17,40s / 6,85s	14,67s / 35,05s	42,22s / 1,56s
COMPRESS	9,30s / 4,30s	14,05s / 36,78s	61,18s / 1,79s
PARALLEL 4	11,67s / 5,25s	14,55s / 34,73s	50,07s / 1,47s
ONLINE	6,22s / 4,34s	14,32s / 36,40s	44,01s / 1,53s

DML Operation war auf 100.000 Zeilen beschränkt

DML beschleunigen

- **Wenn Sie fast die ganze (sehr große) Tabelle mit einem Update (auf die Index-Spalte) beglücken (bzw. mit einem Multi-Row Insert die Tabelle füllen), ist meist folgendes Verfahren günstiger:**
  - `DROP INDEX scott.big_ix;`
  - `UPDATE scott.big SET sal=sal+1;`
  - `--INSERT /*+ APPEND */ INTO scott.big`
    `--SELECT * FROM tab;`
  - `--DELETE FROM scott.big WHERE deptno=10;`
  - `CREATE INDEX scott.big_ix ON`
    `scott.big_emp(sal) NOLOGGING`
    `…;`

Fazit

- **NOLOGGING beschleunigt die Index-Erstellung, birgt aber Gefahren**
- **Parallelisierung bringt nicht zwingend etwas**
- **COMPRESS kostet bei der Erstellung sogar weniger Zeit und man erhält einen schlanken Index**
- **ONLINE (nur in der EE) kostet keine zusätzliche Zeit, die Tabelle ist aber bereits während der Indexerstellung verfügbar**
- **Update und Delete sind ca. Faktor 4 langsamer. Ein Insert ist Faktor 14 langsamer!!!**

MuniQSoft GmbH

- Gegründet 1998
- Tätigkeitsbereiche:
  - Oracle IT Consulting & Services
  - Oracle Schulungen (SQL, PL/SQL, DBA, APEX, B&R, …)
  - Software-Lösungen
  - Oracle Lizenzen

MuniQSoft GmbH
Grünwalder Weg 13 a
D-82008 Unterhaching
www.muniqsoft.de
+49 89 6228 6789-0