Optimizer Fails

Dierk Lenz

DOAG 2018 Datenbank

14. Mai 2018

Herrmann & Lenz Services GmbH
Herrmann & Lenz Solutions GmbH

Herrmann & Lenz Services
Herrmann & Lenz Solutions

- Erfolgreich seit 1996 am Markt
- Firmensitz: Burscheid (bei Leverkusen)
- Beratung, Schulung und Betrieb/Fernwartung rund um das Thema Oracle Datenbanken
- Schwerpunktthemen: Hochverfügbarkeit, Tuning, Migrationen und Troubleshooting / Monitoring
- Herrmann & Lenz Solutions GmbH
  - Produkte: Monitoring Module, Taskzone

Prolog: Optimizer, Ausführungspläne und Kardinalitäten

Komponenten des Optimizers

- Tabellen, Indizes, Constraints usw.:
  - Objekte bestimmen Möglichkeiten des Optimizers
  - Kein Indexzugriff ohne Index
- Statistiken für alle diese Objekte
  - Regelmäßige Berechnung durch Autotasks
  - Intelligente Entscheidungen, u.a.:
    - Histogramme?
    - Histogrammtyp?
- Berechnung von Ausführungsplänen anhand der vorliegenden Daten

Zentraler Fakt in der Berechnung: Kardinalität

- Kardinalität: geschätzte Größe der (Zwischen-) Ergebnismenge (Einheit: Anzahl Datensätze)
- Je komplexer die Abfrage, desto schwieriger die Schätzung
- Einfach: Abfrage auf Gleichheit mit Primärschlüssel (Kardinalität maximal 1)
- Alles andere: schwieriger
- Inhalt des Vortrags: Beispiele, die über falsche Kardinalitäten zu falschen Einschätzungen führen

…und damit oft zu problemtaischen Ausführungsplänen!

Histogramme

- Im Rahmen der Statistiksammlung automatisch erstellt, wenn Werteverteilung nicht gleichmäßig
- Unterschiedliche Ausführungspläne für verschiedene Werte
- Gut funktionierender Ansatz, aber Probleme…
  - bei Verwendung von Bindevariablen
  - wenn Histogrammverfahren „überfordert“
- Verbesserungen bei 12c: Neue Histogrammtypen!

Verwendete Werkzeuge

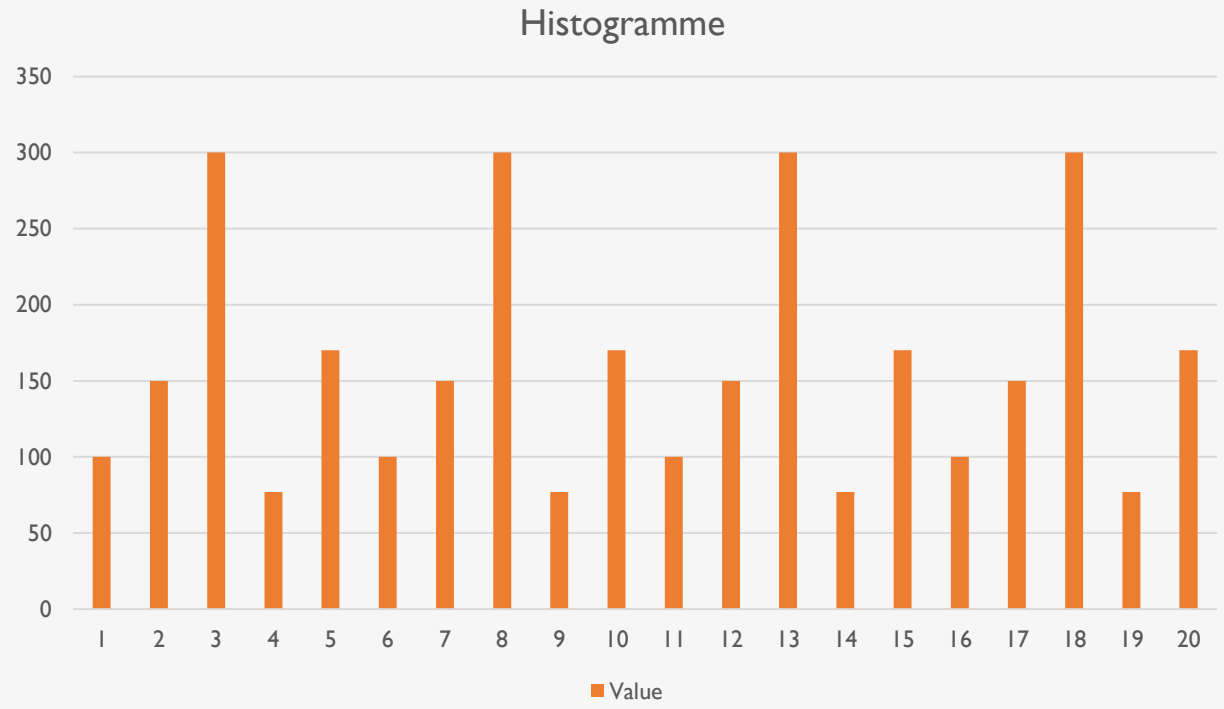
- Beispiele auf 12c R2 nachgestellt
- SQL Developer als Frontend
- Zum Befüllen von Beispieltabellen unerlässlich:
  - PL/SQL
  - DBMS_RANDOM
- Mit DBMS_RANDOM.SEED Zufallswerte wiederholbar erzeugen

Constructing a Fail

Histogramme

Histogramme: Basic Facts

- Anzahl möglicher Histogram Buckets: 2048 (früher 255)
- Höhenbasierte Histogramme:



Histogramme austricksen

- Bei 2048 Buckets: 4096 Werte erzeugen mit abwechselnd geringer und großer Kardinalität

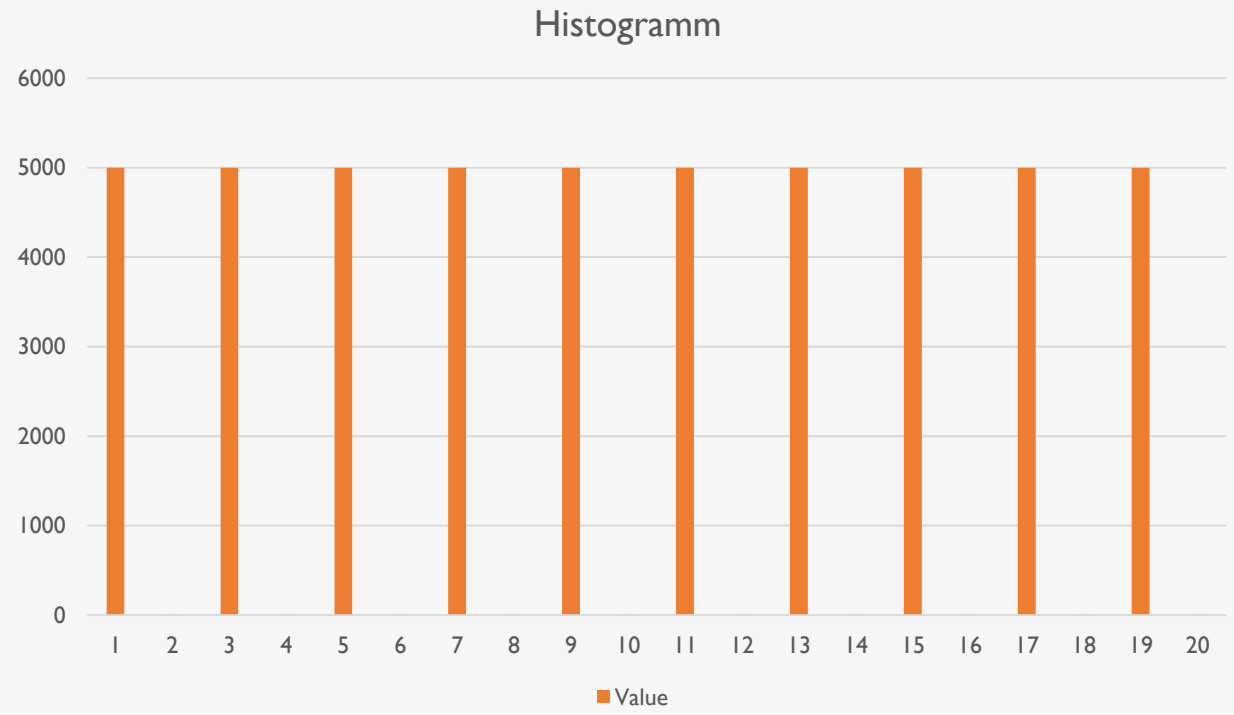


Tabelle füllen

```
CREATE TABLE size2048 (
    i NUMBER,
    c VARCHAR2(1000)
)
PCTFREE 0
;

BEGIN
    FOR vi IN 1..4096 LOOP
        IF MOD(vi,2) = 0 THEN
            INSERT INTO size2048 VALUES (vi, dbms_random.string('X', 10));
        ELSE
            FOR vj IN 1..5000 LOOP
                INSERT INTO size2048 VALUES (vi, dbms_random.string('X', 10));
            END LOOP;
        END IF;
        IF MOD(vi, 100) = 0 THEN
            COMMIT;
        END IF;
    END LOOP;
    COMMIT;
END;
/

CREATE INDEX i_size2048 ON size2048 (i)
;

EXEC dbms_stats.gather_table_stats('DOAG2018', 'SIZE2048', method_opt => 'FOR COLUMNS i SIZE 2048');
```

Demo

Lessons Learned

- Alternierende Größen der Datenmenge pro Wert problematisch!
- Beispiel konstruiert, aber man weiß nie…

Lange VARCHARs

Problemstellung: Zusammengesetzte Schlüssel

- Statt einem Feld für jeden Wert: Konkatenierte Schlüssel (Zeichen 1-8: Wert A, Zeichen 9-20: Wert B usw.)
- Beispiel hier: Führende Felder für zukünftige Werte, daher alle '0'

Tabelle füllen

```
CREATE TABLE looongvc
(c varchar2(200))
;

BEGIN
    dbms_random.seed('DOAG2018');
    FOR i in 1..1000 LOOP
        FOR j in 1..(TRUNC(dbms_random.value * 1000) + 1) LOOP
            INSERT INTO looongvc
               VALUES (lpad(to_char(i), 120, '0'));
        END LOOP;
    END LOOP;
END;
/

CREATE INDEX i_looongvc ON looongvc (c)
;

EXEC dbms_stats.gather_table_stats('DOAG2018', 'LOOONGVC',
            method_opt => 'FOR COLUMNS c SIZE 2048')
```

Demo

Lessons Learned

- Für Histogramme: VARCHARs auf 64 Zeichen gekürzt!
- Wenn kein Unterschied in den ersten 64 Zeichen: kein Unterschied für Berechnung des Ausführungsplans
- Signifikante Daten nach vorne!

Klassiker für Extended Statistics: Postleitzahlen und Orte

Korrelationen von Spalteninhalten

- PLZ und Ort korreliert, da PLZ den Ort bereits bestimmt
- Wenn sauber in Normalform designed: entweder nur PLZ in Tabelle oder Fremdschlüssel auf PLZ/Orte-Tabelle
- Macht niemand!
- Sondern…

Tabelle füllen

```
CREATE TABLE orte (
    osm_id NUMBER,
    ort VARCHAR2(100 CHAR),
    plz VARCHAR2(5 CHAR),
    bundesland VARCHAR2(100 CHAR)
)
;

-- SQL Loader zum Laden einer CSV Datei benutzt

CREATE TABLE personen (
    name VARCHAR2(20 CHAR),
    plz VARCHAR2(5 CHAR),
    ort VARCHAR2(100 CHAR)
)
;

DECLARE
    TYPE ttyp IS TABLE OF orte%ROWTYPE INDEX BY binary_integer;
    t ttyp;
    i NUMBER;
    l NUMBER;
BEGIN
    dbms_random.seed('PLZundOrte');
    i := 0;
    FOR r IN (SELECT plz, ort FROM orte) LOOP
        t(i).plz := r.plz;
        t(i).ort := r.ort;
        i := i + 1;
    END LOOP;
    FOR k in 1..100000 LOOP
        l := TRUNC(dbms_random.value * i);
        INSERT INTO personen VALUES (dbms_random.string('U', 20), t(l).plz, t(l).ort);
    END LOOP;
    COMMIT;
END;
/
```

Demo

Lessons Learned

- Vorsicht bei korrelierten Spalteninhalten!
- Ggfs. Extended Statistics einsetzen
- Bei Oracle 12.1 überprüfen: Automatische Erzeugung!
- Manchmal Redesign sinnvoll

Fragen & Kontakt

- Mail: dierk.lenz@hl-services.de
- Web: www.hl-services.de
- Blog: blog.hl-services.de
- Twitter: @ora1578

Vielen Dank für Ihre Aufmerksamkeit!