# Tablespace-Monitoring-Fallstricke

Autor: Dr. Alexander Kick, Credit Suisse, Zürich

DOAG*News* Q4_2005

*Neben der Alert-Log-Datei und der Datenbank-Instanz an sich gehören die Tablespaces zu den wichtigsten Dingen, die bei einer Oracle-Instanz überwacht werden sollten. Am aussagekräftigsten ist dabei der Prozentsatz des noch freien oder schon belegten Platzes. Die Schwelle, wann eine Benachrichtigung erfolgen soll, kann dabei unterschiedlich gewählt werden.*

In verschiedenen Büchern und im Internet werden unterschiedliche SQL-Scripts vorgestellt, mit denen der freie Platz in den einzelnen Tablespaces angezeigt wird. Leider sind erstaunlicherweise fast alle davon mit Fehlern behaftet oder suboptimal. Im Folgenden gehen wir auf die unterschiedlichen Fallstricke ein. Dabei beschränken wir uns auf die Version von Oracle, die zur Zeit am häufigsten eingesetzt wird, nämich Orcacle 9*i*. Ebenso beschränken wir uns – der Verständlichkeit halber – auf die Berechnung des Prozentsatzes. Die gezeigten Scripts können selbstversändlich so ergänzt werden, dass auch noch andere Angaben angezeigt werden.

## Test-Szenario

Als Test-Szenario und zur Veranschaulichung wird ein Tablespace mit zwei Datafiles kreiert, wovon eines autoextensible ist. In diesem Tablespace wird dann auch eine Test-Tabelle angelegt.

```
SQL> create tablespace test
     datafile '/oradata01/IMDM/test_dbf01' size 10M autoextend on maxsize 11M,
     '/oradata01/IMDM/test_dbf02' size 1M;
SQL> create table sys.testtable (a varchar2(4000)) storage (initial 9M) tablespace test;
```

Welchen freien Platz erwarten wir? Da durch die Tabelle 9 Megabyte (MB) belegt sind, das erste Datafile auf 11 MB automatisch erweitert werden kann und das zweite Datafile 1 MB gross ist, ist der Tablespace zu 9/(11 + 1) % = 75 % voll, hat also 25% freien Platz.

## Ohne Berücksichtigung von autoextensible

Obwohl Mike Ault's Buch [1] nur SQL-Scripts enthält, wird die automatische Erweiterbarkeit von Tablespaces bei der Berechnung des freien Prozentsatzes im Tablespace-Monitoring-Script nicht berücksichtigt. Dies ist leider ebenso bei Tom Kyte (Asktom) [2] der Fall. Im Folgenden wird nun ein einfaches Überwachungsscript für permanente Tablespaces vorgestellt, wie es im Prinzip sehr häufig vorkommt. Es werden die freien Bytes aus dba_free_space und die allozierten Bytes aus dba_data_files aufsummiert und daraus der Prozentsatz berechnet.

```
col tablespace_name format a20 heading 'Tablespace'
col pct_free format 990.00 heading '% Free'

select a.tablespace_name, round(nvl(free/allocated,0),4)*100 pct_free from
(select tablespace_name, sum(bytes) free
 from dba_free_space
 group by tablespace_name) f,
```

```
(select tablespace_name, sum(bytes) allocated
 from dba_data_files
 group by tablespace_name) a
where a.tablespace_name = f.tablespace_name;
```

Wie lautet das Ergebnis für das Testszenario?

```
Tablespace            % Free
--------------------- -------
TEST                   17.05
```

Das Script liefert also nicht das gewünschte Ergebnis. Ausserdem hat es einen anderen Fehler, der selbst bekannten Autoren wie Tim Gorman [3] unterläuft.

```
SQL> create tablespace testfull datafile '/oradata01/IMDM/testfull_dbf01' size 10M uniform size
1M;
SQL> create table sys.testtablef (a varchar2(4000)) storage (initial 9M) tablespace testfull;
```

Leider erscheint der gefüllte Tablespace nicht im Output:

```
Tablespace            % Free
--------------------- -------
…
```

Da es auch gar keine freien Extents in dba_free_space geben kann, ist ein Outer-Join notwendig:

```
select a.tablespace_name, round(nvl(free/allocated,0),4)*100 pct_free from
(select tablespace_name, sum(bytes) free
 from dba_free_space
 group by tablespace_name) f,
(select tablespace_name, sum(bytes) allocated
 from dba_data_files
 group by tablespace_name) a
where a.tablespace_name = f.tablespace_name(+);
```

```
Tablespace            % Free
--------------------- -------
TESTFULL                0.00
```

Wie weiter unten gezeigt wird, enthält dieses Script auch für nicht autoextensible Tablespaces noch Unschönheiten.

## Berücksichtigung von autoextensible

Wir sind der Meinung, dass das Autoextensible-Feature einige Vorteile mit sich bringt. Denn was macht man, wenn man 500 Datenbank-Instanzen betreuen muss? Läuft ein Tablespace voll und ist dieser nicht auf autoextensible, so wird er *manuell* erweitert – ohne gross nachzufragen, warum dieser vollgelaufen ist. In unserer Laufbahn als Oracle-DBA haben wir nur sehr selten erlebt, dass ein Tablespace wegen eines fehlerhaften Inserts bis ins "Unendliche" gewachsen ist. Dennoch ist es sinnvoll, die maxsize (bis zu dieser Grösse kann ein Datafile, das autoextensible ist, vergrössert werden) zu beschränken (insbesondere auch in Hinblick auf die Grösse des darunter liegenden Filesystems). Setzt man dieses Feature also vernünftig ein, kann man sich viel langweilige Arbeit sparen. Aus diesem Grunde sollte eine Tablespace-Überwachung die automatische Erweiterbarkeit berücksichtigen. Dabei ist selbstverständlich zu beachten, dass das Filesystem, in dem die Oracle-Datafiles liegen, auch überwacht wird und eine Meldung generiert wird, bevor dieses voll wird (oder die Summe der maxbytes-Werte aller im Filesystem liegenden Datafiles kleiner gewählt wird als die Filesystem-Größe).

In [4] wird ein Script vorgestellt, das autoextensible Tablespaces "berücksichtigt". Die Information, wie groß ein Datafile, das autoextensible ist, maximal werden kann, ist in der Spalte maxbytes in dba_data_files vorhanden.

```
select a.tablespace_name, round((free + potential)/(allocated + potential),4)*100 pct_free from
(select tablespace_name, sum(bytes) free
 from dba_free_space
 group by tablespace_name) f,
(select tablespace_name, sum(maxbytes) potential, sum(bytes) allocated
 from dba_data_files
 group by tablespace_name) a
where a.tablespace_name = f.tablespace_name(+);
```

Leider liefert es ein falsches Ergebnis:

```
Tablespace            % Free
-------------------- -------
TEST                   58.52
```

Das Script berücksichtigt nämlich nicht den Fall (wie im Testszenario), dass ein Tablespace aus mehreren Data-Files bestehen kann, wovon ein Teil autoextensible und ein anderer Teil nicht autoextensible ist. Darüber hinaus liefert es für ganz volle Tablespaces null anstatt 0 % frei zurück.

Im folgenden Script sind beide Fehler eliminiert, wobei der erste Fehler dadurch behoben wurde, dass nach der automatischen Erweiterbarkeit eines Data-Files unterschieden wird.

```
select
   a.tablespace_name,
   round((nvl(free,0) + potential)/(allocated + potential),4)*100 pct_free
```

```
from
(select tablespace_name, sum(bytes) free
 from dba_free_space
 group by tablespace_name) f,
(select tablespace_name, sum(decode(maxbytes,0,0,maxbytes - bytes)) potential,
 sum(bytes) allocated
 from dba_data_files
 group by tablespace_name) a
where a.tablespace_name = f.tablespace_name(+);
```

```
Tablespace            % Free
--------------------  -------
TEST                   23.96
```

## Maxbytes < Bytes

Auf einen Ausnahmefall wurde jedoch in obigen Scripts nicht Acht gegeben. Maxsize kann nämlich kleiner gesetzt werden als die tatsächliche Grösse des Datafiles.

```
alter database datafile '/oradata01/IMDM/test_dbf01' autoextend on maxsize 5M;
```

```
Tablespace            % Free
--------------------  -------
TEST                  -52.08
```

Oops! Der Tablespace hat -52.08 % freien Platz! Das müssen wir aber schnell beheben:

```
select
   a.tablespace_name,
   round((nvl(free,0) + potential)/(allocated + potential),4)*100 pct_free
from
(select tablespace_name, sum(bytes) free
 from dba_free_space
 group by tablespace_name) f,
(select tablespace_name,
 sum(case
   when maxbytes <= bytes then 0
   else maxbytes - bytes
 end) potential,
 sum(bytes) allocated
 from dba_data_files
 group by tablespace_name) a
where a.tablespace_name = f.tablespace_name(+);
```

Um eine andere Unschönheit zu veranschaulichen, wird ein weiterer Tablespace erzeugt:

```
SQL> create tablespace test2 datafile '/oradata01/IMDM/test2_dbf01' size 128k;
```

Das obige Script liefert nun:

```
Tablespace            % Free
-------------------- -------
TEST                   17.05
TEST2                  50.00
```

Der %-Free-Wert für den Tablespace TEST ist im Grossen und Ganzen in Ordnung: 1 - 9/(10 + 1) % = 18.2 % und dies ist ungefähr 17 %. Woran liegt es aber, dass der neu erzeugte Tablespace TEST2 nur halb leer ist?

## User_bytes

Gemäß Oracle9*i*-Database-Reference gibt die bytes-Spalte in dba_data_files die Größe des Data-Files in Bytes an, die user_bytes-Spalte hingegen die Byte-Anzahl, die von Datenbank-Objekten tatsächlich benutzt werden kann. Ein paar Blöcke werden von Oracle für die Verwaltung benötigt. Sicherlich ist das Test-Szenario mit dem Test2-Tablespace konstruiert, denn solch kleine Tablespaces legt man normalerweise nicht an. Bei großen Tablespaces fällt der Unterschied zwischen bytes und user_bytes nicht ins Gewicht, wenn man nur wissen will, ob %-Free einen bestimmten Schwellenwert unterschritten hat. Erzeugt man sich jedoch einen Report, so ist die Information, dass 100% statt 99.98 % frei sind, aussagekräftiger, denn bei 99.98 % weiss man nicht, ob rein gar nichts im Tablespace drin ist oder ob es doch ein kleines Segment gibt. Da es kaum aufwändiger ist, stellt sich jedoch die Frage, warum man es nicht genau macht. Wir mussten lange suchen, um Tablespace-Monitoring-Scripts zu entdecken, in denen user_bytes verwendet wird (Ein solches ist [3], das jedoch wieder einige andere Fehler hat.). Ändert man das vorherige Script, erhält man:

```
select
   a.tablespace_name,
   round((nvl(free,0) + potential)/(allocated + potential),4)*100 pct_free
from
(select tablespace_name, sum(bytes) free
 from dba_free_space
 group by tablespace_name) f,
(select tablespace_name,
 sum(case
   when maxbytes <= bytes then 0
   else maxbytes - bytes
 end) potential,
 sum(user_bytes) allocated
 from dba_data_files
 group by tablespace_name) a
where a.tablespace_name = f.tablespace_name(+);
```

Bevor wir dieses Script laufen lassen, setzen wir maxsize wieder auf den ursprünglichen Wert:

```
SQL> alter database datafile '/oradata01/IMDM/test_dbf01' autoextend on maxsize 11M;
```

```
Tablespace            % Free
-------------------- -------
TEST                   24.21
TEST2                 100.00
```

So sieht es schon viel besser aus!

## Weitere Fehler?

Kann es einen Division-By-Zero-Fehler geben? Nein, denn Tablespaces mit 0 user_bytes lassen sich zumindest in Oracle-Version 9.2 nicht anlegen:

```
SQL> create tablespace test3 datafile '/oradata01/IMDM/test3_dbf01' size 80k;
create tablespace test3 datafile '/oradata01/IMDM/test3_dbf01' size 80k
*
ERROR at line 1:
ORA-03214: File Size specified is smaller than minimum required
```

Was passiert, wenn ein tablespace offline genommen wird?

```
SQL> alter tablespace test2 offline;
```

```
Tablespace            % Free
-------------------- -------
TEST                   24.21
TEST2
```

Was liefert in dem Fall das zweite innere Select?

```
select tablespace_name,
 sum(case
   when maxbytes <= bytes then 0
   else maxbytes - bytes
 end) potential,
 sum(user_bytes) allocated
 from dba_data_files
 group by tablespace_name;
```

```
Tablespace            POTENTIAL  ALLOCATED
-------------------- ---------- ----------
TEST                    1048576   11403264
TEST2
```

Potential und allocated können also null sein.

Die Frage bleibt, was man angezeigt haben möchte: null oder 0 %. Für null spricht, dass Oracle im Moment nicht feststellen kann, was im Tablespace theoretisch noch frei ist. Für 0 %

spricht, dass in dem offline-Tablespace kein Extent mehr kreiert werden kann. Sammelt man die %-Frei-Werte pro Tablespace in regelmässigen Abständen und speichert diese für spätere Auswertungen – man kann daraus z.B. eine Grafik erzeugen – so ist 0 % wohl ebenfalls vorzuziehen. In diesem Fall würde das folgendes Script ergeben:

```
select
   a.tablespace_name,
   round(nvl((nvl(free,0) + potential)/(allocated + potential),0),4)*100 pct_free
from
(select tablespace_name, sum(bytes) free
 from dba_free_space
 group by tablespace_name) f,
(select tablespace_name,
 sum(case
   when maxbytes <= bytes then 0
   else maxbytes - bytes
 end) potential,
 sum(user_bytes) allocated
 from dba_data_files
 group by tablespace_name) a
where a.tablespace_name = f.tablespace_name(+);
```

```
Tablespace            % Free
-------------------- -------
TEST                   24.21
TEST2                   0.00
```

## Performance

In [5] wird ein anderes Script zur Verfügung gestellt. Die Gruppierungen in den beiden inneren Select-Abfragen sind jedoch unnötig. Lässt man sie weg, muss man aber das `union` durch ein `union all` ersetzen. Ändert man dieses Script darüber hinaus so, dass die besprochenen Fallstricke berücksichtigt werden, dann erhält man:

```
select
   tablespace_name,
   round(nvl(1 - (sum(allocated) - sum(free))/sum(potential),0),4)*100 pct_free
from
(select tablespace_name, bytes free, 0 potential, 0 allocated
 from dba_free_space
 union all
 select tablespace_name, 0 free,
 case
   when maxbytes <= bytes then user_bytes
   else maxbytes - bytes + user_bytes
 end potential,
```

```
 user_bytes allocated
 from dba_data_files)
group by tablespace_name;
```

```
Tablespace            % Free
-------------------- -------
TEST                   24.21
```

Das vorher gezeigte Script ist bei sehr großen Datenbanken mit vielen Tablespaces und Data-Files unwesentlich schneller als das obige. Beim 3.3-TB-Datawarehouse mit 134 Tablespaces und 310 Data-Files dauerte dieses Script im Schnitt 41.4 Sekunden verglichen mit 43.5 Sekunden, und beim 2.5-TB-Data-Mart mit 163 Tablespaces und 388 Datafiles 5 statt 4.8 Sekunden. Bei den meisten Datenbanken brauchen beide Scripts jedoch weniger als eine Zehntelsekunde.

## Temporary-Tablespaces

Bei den Temporary-Tablespaces stellt sich die Frage, ob diese überhaupt überwacht werden sollten. Läuft ein Temporary-Tablespace voll, so hat man in der Regel nicht mehr genug Zeit, um diesen manuell zu erweitern. Wird das Alert-Log überwacht, erhält man auch so die Meldung, dass temporäre Segmente nicht erweitert werden können. Hinzu kommt, dass es fehlerhafte Abfragen geben kann, die den Temporary-Tablespace füllen. Leicht herausfinden kann man dies mittels der Überwachungsprozeduren in [6]. In diesem Fall sollte der Temporary-Tablespace nicht erweitert werden. Bei der Überwachung von Temporary-Tablespaces sollte man daher – wenn man autoextend verwendet – keinen allzu großen Wert für maxsize wählen, da fehlerhafte Abfragen z.B. im Datawarehouse aufgrund vieler Adhoc-Abfragen durch so genannte Power-User regelmässig vorkommen.

In Oracle 9*i* werden Temporary-Tablespaces normalerweise als Locally-Managed-Tablespaces angelegt. Im Folgenden gehen wir vorerst nur auf erstere ein; auf Dictionary-Managed-Temporary-Tablespaces weisen wir dann weiter unten hin.

Zu beachten ist, dass in dba_free_space für temporäre Tablespaces oft weniger freier Platz angezeigt wird, als eigentlich frei ist, da einmal benutzte, aber nicht mehr benötigte temporäre Segmente erst verzögert freigegeben werden. Aus diesem Grund verwenden wir v$temp_extent_pool zur Berechnung. Dabei gibt die Spalte bytes_used in der view v$temp_extent_pool an, wie viele Bytes an temporärem Platz gerade benutzt werden (Oracle9*i*-Database-Reference: "This view displays the state of temporary space cached and used for a given instance."). Im Gegensatz zum Script für permanente Tablespaces kann man die Werte für allocated und potential im inneren Select auf einmal aufsummieren. Statt aus dba_data_files müssen die Informationen für (locally managed) temporäre Tablespaces jedoch aus dba_temp_files extrahiert werden. Berücksichtigt man dies und die besprochenen Fallstricke, so erhält man:

```
select
   t.tablespace_name,
   round((potential - nvl(used,0))/potential,4)*100 pct_free
from
```

```
(select tablespace_name, sum(bytes_used) used
 from v$temp_extent_pool
 group by tablespace_name) e,
(select tablespace_name,
 sum(case
    when maxbytes <= bytes then user_bytes
    else maxbytes - bytes + user_bytes
 end) potential
 from dba_temp_files
 group by tablespace_name) t
where t.tablespace_name = e.tablespace_name(+);
```

Da sich Locally-Managed-Temporary-Tablespaces nicht offline setzen lassen, braucht dies nicht wie im Fall der permanenten Tablespaces berücksichtigt zu werden.

## Gesamt-Script

Die Scripts für permanente und temporäre Tablespaces lassen sich mittels `union all` verknüpfen. Darüber hinaus möchte man sich oft nur diejenigen Tablespaces ansehen, deren %-Free-Wert kleiner als ein gewählter Schwellenwert ist und ausserdem nicht read only sind. Dies erreicht man auf einfache Weise:

```
select tablespace_name, pct_free from
   (select
     a.tablespace_name,
     round(nvl((nvl(free,0) + potential)/(allocated + potential),0),4)*100 pct_free
   from
   (select tablespace_name, sum(bytes) free
    from dba_free_space
    group by tablespace_name) f,
   (select tablespace_name,
    sum(case
      when maxbytes <= bytes then 0
      else maxbytes - bytes
    end) potential,
    sum(user_bytes) allocated
    from dba_data_files
    group by tablespace_name) a,
   dba_tablespaces t
   where a.tablespace_name = f.tablespace_name(+)
   and a.tablespace_name=t.tablespace_name
   and t.status != 'READ ONLY'
   union all
   select
    t.tablespace_name,
    round((potential - nvl(used,0))/potential,4)*100 pct_free
   from
   (select tablespace_name, sum(bytes_used) used
```

```
    from v$temp_extent_pool
    group by tablespace_name) e,
   (select tablespace_name,
    sum(case
       when maxbytes <= bytes then user_bytes
       else maxbytes - bytes + user_bytes
    end) potential
    from dba_temp_files
    group by tablespace_name) t
   where t.tablespace_name = e.tablespace_name(+))
where pct_free < 20;
```

Im obigen Gesamt-Script werden Dictionary-Managed-Temporary-Tablespaces im Teil für permanente Tablespaces erfasst. Möchte man gar keine Temporary-Tablespaces überwachen, müsste man das "union all" und das zweite innere Select (Locally-Managed-Temporary-Tablespaces) weglassen und dem ersten inneren Select noch die Einschränkung `t.contents='PERMANENT'` hinzufügen.

## Zusammenfassung

Ein in allen Fällen korrektes Script für die Tablespace-Überwachung zu schreiben ist also nicht einfach. Offenbar hat sich bis jetzt niemand so richtig die Mühe gemacht – zumindest konnten wir kein Script im Internet und in Büchern entdecken, das alle Fallstricke berücksichtigt. In diesem Artikel haben wir auf die vielen Fallstricke hingewiesen und ein hoffentlich fehlerfreies Script für Oracle 9*i* entwickelt. Feedback ist herzlich willkommen.

## Literatur


[1] Mike Ault, Oracle Internals Monitoring & Tuning Scripts, Rampant Tech Press, ISBN 0-9727513-8-6
[2] Tom Kyte, http://asktom.oracle.com/~tkyte/Misc/free.html
[3] Tim Gorman, http://www.evdbt.com/chk_spc.sh
[4] McDonald, Mastering Oracle PL/SQL, S. 404
[5] http://www.psoug.org/reference/tablespaces.html
[6] Alexander Kick und Daniel Steiger, Überwachung problematischer Abfragen im Datawarehouse-Umfeld, DOAG Newletter 4/2005



Kontakt:
Dr. Alexander Kick
alexander.kick@credit-suisse.com