

Tipps & Tricks: Mai 2003

| | | | |
|---------------|----------------------------|-----------------------|------------|
| Bereich: | DBA | Erstellung: | 05/2003 HA |
| Versionsinfo: | getestet mit 10.2 und 11.1 | Letzte Überarbeitung: | 05/2009 RM |

 [Als PDF Downloaden!](#)

Space Management von Objekten

Tabellen und Indices benötigen Speicherplatz, der in Form von Extents allokiert werden muss, wobei ein Extent aus einer bestimmten Anzahl an Blöcken besteht. Woher bekomme ich Informationen darüber, wie viel Platz meine Objekte benötigen, und wie viel Platz noch verfügbar ist, bevor ein neues Extent allokiert werden muss oder bevor der Tablespace voll ist?

Data Dictionary Views

Folgende Data Dictionary Views sind in diesen Zusammenhang von Interesse:

- DBA_SEGMENTS
- DBA_EXTENTS
- DBA_TABLES bzw. DBA_INDEXES
- DBA_FREE_SPACE

In DBA_SEGMENTS kann man sehen, wie viele Blöcke / Bytes ein Segment, also beispielsweise eine Tabelle oder ein Index, im Augenblick belegt, und auf welchem Tablespace. Ausserdem kann abgelesen werden, wie vielen Extents das entspricht, und wie groß das nächste zu allozierende Extent sein wird:

```
SELECT tablespace_name, bytes, blocks, extents, next_extent
FROM DBA_SEGMENTS
WHERE segment_name = 'BIGEMP' AND owner = 'SCOTT';
```

| TABLESPACE_NAME | BYTES | BLOCKS | EXTENTS | NEXT_EXTENT |
|-----------------|---------|--------|---------|-------------|
| USERS | 4718592 | 576 | 8 | 573440 |

Das heisst allerdings nicht, dass alle Blöcke auch gefüllt sein müssen; sie sind nur allokiert. In DBA_EXTENTS wird die Information aufgeschlüsselt nach einzelnen Extents. Für BIGEMP gibt es daher acht Einträge.

In DBA_TABLES (bzw. DBA_INDEXES) findet man ähnliche Informationen, nur wird hier auch noch unterschieden, wie viele der allokierten Blöcke in Benutzung sind bzw. waren, und wie viele noch nie gefüllt wurden (d.h., sich jenseits der High Water Mark befinden). Voraussetzung ist allerdings, dass die Tabelle (bzw. der Index) analysiert wurde.

```
SELECT tablespace_name, blocks, empty_blocks, next_extent
FROM DBA_TABLES
WHERE table_name = 'BIGEMP' AND owner = 'SCOTT';
```

| TABLESPACE_NAME | BLOCKS | EMPTY_BLOCKS | NEXT_EXTENT |
|-----------------|--------|--------------|-------------|
| ----- | ----- | ----- | ----- |

| | | | |
|-------|-----|----|--------|
| USERS | 520 | 55 | 573440 |
|-------|-----|----|--------|

Die Differenz von einem Block zu DBA_SEGMENTS ergibt sich aus dem nicht mitgezählten Segment Header.

In DBA_FREE_SPACE kann man nachsehen, ob es auf diesem Tablespace noch genügend zusammenhängenden Speicherplatz gibt, um ein nächstes Extent zu allokkieren:

```
SELECT file_id, bytes, blocks FROM dba_free_space
WHERE tablespace_name = 'USERS' AND bytes >= 573440;
```

| FILE_ID | BYTES | BLOCKS |
|---------|---------|--------|
| 3 | 4186112 | 511 |
| 8 | 5890048 | 719 |

Das DBMS_SPACE-Package

Mit diesem Package können ebenfalls Informationen über die Segmentgröße und Platzbedarf gesammelt werden, die z.T. noch über die Informationen in den Data Dictionary Views hinausgehen. Das Package enthält in Version 9.x folgende Prozeduren:

- UNUSED_SPACE
- FREE_BLOCKS
- SPACE_USAGE (ab Version 9i)

UNUSED_SPACE liefert ähnliche Informationen wie die DD-Views, also Gesamtzahl der allokkierten Blöcke / Bytes und Anzahl der Blöcke jenseits der High Water Mark. FREE_BLOCKS dagegen schlüsselt auf, wie viele Blöcke unterhalb der High Water Mark als frei betrachtet werden, also für INSERT und UPDATE zur Verfügung stehen. Für locally managed Tablespaces mit autoallocate muss ab Version 9i statt FREE_BLOCKS die Prozedur SPACE_USAGE genutzt werden, die aufschlüsselt, wie viele Blöcke / Bytes unterhalb der High Water Mark komplett leer, bis zu 25%, 50%, 75%, 100% frei oder komplett voll sind.

Um diese Prozeduren aufrufen zu können, benötigt man das ANALYZE-Recht auf das jeweilige Objekt bzw. ANALYZE ANY-Recht. Angegeben werden müssen jeweils:

- Segment_Owner
- Segment_Name
- Segment_Typ (Möglichkeiten: TABLE, TABLE PARTITION, TABLE SUBPARTITION, INDEX, INDEX PARTITION, INDEX SUBPARTITION, CLUSTER, LOB)

Bei FREE_BLOCKS muss zusätzlich als vierter Parameter noch angegeben werden, welche freelist_group untersucht werden soll.

Beispiele:

```
CREATE OR REPLACE PROCEDURE analyze_object (
  p_owner in VARCHAR2,
  p_name in VARCHAR2,
  p_type in VARCHAR2 DEFAULT 'TABLE',
  p_partition_name IN VARCHAR2 DEFAULT NULL)
AS
  v_total_blocks NUMBER;
  v_total_bytes NUMBER;
  v_unused_blocks NUMBER;
  v_unused_bytes NUMBER;
```

```
v_last_used_extent_file_id NUMBER;
v_last_used_extent_block_id NUMBER;
v_last_used_block NUMBER;
BEGIN
  DBMS_SPACE.UNUSED_SPACE(p_owner, p_name, p_type,
    v_total_blocks, v_total_bytes,
    v_unused_blocks, v_unused_bytes,
    v_last_used_extent_file_id, v_last_used_extent_block_id,
    v_last_used_block, p_partition_name);
  DBMS_OUTPUT.PUT_LINE('Blöcke insgesamt: ' || v_total_blocks);
  DBMS_OUTPUT.PUT_LINE('Bytes insgesamt: ' || v_total_bytes);
  DBMS_OUTPUT.PUT_LINE('Ungenutzte Blöcke: ' || v_unused_blocks);
  DBMS_OUTPUT.PUT_LINE('Ungenutzte Bytes: ' || v_unused_bytes);
END;
EXEC analyze_object('SCOTT', 'BIGEMP')
```

```
Blöcke insgesamt: 576
Bytes insgesamt: 4718592
Ungenutzte Blöcke: 55
Ungenutzte Bytes: 450560
```

```
CREATE OR REPLACE PROCEDURE analyze_object2(
  p_owner in VARCHAR2,
  p_name in VARCHAR2,
  p_type in VARCHAR2 DEFAULT 'TABLE',
  p_freelist_group IN NUMBER DEFAULT 0,
  p_scan_limit IN NUMBER DEFAULT NULL,
  p_partition_name IN VARCHAR2 DEFAULT NULL)
  IS
  v_free NUMBER;
BEGIN
  DBMS_SPACE.FREE_BLOCKS(p_owner, p_name, p_type,
    p_freelist_group, v_free,
    p_scan_limit, p_partition_name);
  DBMS_OUTPUT.PUT_LINE('freie Blöcke: ' || v_free);
END;
EXEC analyze_object2('SCOTT', 'BIGEMP')
```

```
freie Blöcke: 503
```

```
CREATE PROCEDURE analyze_object3(
  p_owner in VARCHAR2,
  p_name in VARCHAR2,
  p_type in VARCHAR2 DEFAULT 'TABLE',
  p_freelist_group IN NUMBER DEFAULT 0,
  p_partition_name IN VARCHAR2 DEFAULT NULL)
  IS
  v_unf_by NUMBER;
  v_unf_bl NUMBER;
  v_25_by NUMBER;
  v_25_bl NUMBER;
  v_50_by NUMBER;
  v_50_bl NUMBER;
  v_75_by NUMBER;
```

```
v_75_bl NUMBER;  
v_100_by NUMBER;  
v_100_bl NUMBER;  
v_full_by NUMBER;  
v_full_bl NUMBER;  
BEGIN  
  DBMS_SPACE.SPACE_USAGE(p_owner, p_name, p_type,  
    v_unf_bl, v_unf_by,  
    v_25_bl, v_25_by,  
    v_50_bl, v_50_by,  
    v_75_bl, v_75_by,  
    v_100_bl, v_100_by,  
    v_full_bl, v_full_by,  
    p_partition_name);  
  DBMS_OUTPUT.PUT_LINE('freie Blöcke: ' || v_unf_bl);  
  DBMS_OUTPUT.PUT_LINE('bis 25% freie Blöcke: ' || v_25_bl);  
  DBMS_OUTPUT.PUT_LINE('25-50% freie Blöcke: ' || v_50_bl);  
  DBMS_OUTPUT.PUT_LINE('50-75% freie Blöcke: ' || v_75_bl);  
  DBMS_OUTPUT.PUT_LINE('75-100% freie Blöcke: ' || v_100_bl);  
  DBMS_OUTPUT.PUT_LINE('volle Blöcke: ' || v_full_bl);  
END;  
  
exec analyze_object3('SCOTT', 'EMP')
```

```
freie Blöcke: 0  
bis 25% freie Blöcke: 0  
25-50% freie Blöcke: 0  
50-75% freie Blöcke: 0  
75-100% freie Blöcke: 5  
volle Blöcke: 0
```

Alle Variablen (v...), die in den obigen Beispielen an die DBMS_SPACE-Prozeduren übergeben werden, sind OUT-Parameter, während der Rest (p...) IN-Parameter sind.

Wenn man nur daran interessiert ist, ob noch n Blöcke (innerhalb der freelist group) frei sind, dann kann man dieses n als scan_limit bei FREE_BLOCKS mit angeben; dann bricht die Prozedur die Analyse ab, sobald so viele freie Blöcke gefunden wurden.