

Hausputz für Einsteiger

Datenbank-Reorganisation und mehr

Markus Flechtner



markusdba



markusdba.de|.net

BASEL | BERN | BRUGG | BUCHAREST | DÜSSELDORF | FRANKFURT A.M. | FREIBURG I.B.R.
GENEVA | HAMBURG | LAUSANNE | MANNHEIM | MUNICH | STUTTGART | VIENNA | ZÜRICH

trivadis

Markus Flechtner

- Principal Consultant, Trivadis, Düsseldorf
- Oracle seit 1990: SW-Entwicklung, Support, DBA
- Schwerpunkte: RAC, HA, Upgrade & Migration
- Kursreferent: RAC, New Features, Multitenant, PostgreSQL
- Co-Autor des Buches "Der Oracle DBA"
(Hanser, 2016)



@markusdba



www.markusdba.de|.net



ORACLE®
ACE



BASEL | BERN | BRUGG | BUKAREST | DÜSSELDORF | FRANKFURT A.M. | FREIBURG LBR. | GENÈVE
HAMBURG | KOPENHAGEN | LAUSANNE | MANNHEIM | MÜNCHEN | STUTTGART | WIEN | ZÜRICH

trivadis

1994
GEGRÜNDET

300 SLA's
(SERVICE LEVEL AGREEMENTS)

 **700**
MITARBEITENDE

 **15** **TRIVADIS**
WORKSPACES
SCHWEIZ, DEUTSCHLAND,
ÖSTERREICH, RUMÄNIEN

4000 
TRAININGS-TEILNEHMENDE PRO JAHR

5 **MIO.**
CHF 
**FORSCHUNGS- &
ENTWICKLUNGSBUDGET**

118 **MIO.**
CHF **UMSATZ** 

800 
KUNDEN

ERFAHRUNGEN AUS
1900 **PROJEKTEN**
PRO JAHR 

trivadis

Agenda

- Warum aufräumen?
- Reorganisation von Tabellen
- Reorganisation von Indizes
- Ungültige Objekte in der Datenbank
- Nicht mehr genutzte User/Schemata
- Alte Log- und Trace-Dateien
- Zusammenfassung



Warum aufräumen?

5 28.05.20 DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

Warum aufräumen?

- Es gilt das gleiche wie zu Hause:

"Ordnung braucht nur der Dumme, das Genie beherrscht das Chaos."

Albert Einstein (1879-1955)

- Effizientere Speicherplatznutzung
- Bessere Performance
- Einfachere Erkennung von Problemen
- Vermeidung von Systemausfällen

"Housekeeping" sollte automatisiert werden

- Regelmäßig alte Log- und Trace-Dateien komprimieren und/oder löschen
- Regelmäßiges Monitoring ungültiger Objekte in der Datenbank
- Reorganisations-Hinweise des Space Advisors beachten
 - Automatische tägliche "Space-Auswertung" in der DB (Default-Wartungs-Job)
 - Ergebnisse im OEM
 - Package DBMS_ADVISOR
 - Views (u.a.)
 - DBA_ADVISOR_FINDINGS
 - DBA_ADVISOR_RECOMMENDATIONS

Oracle Enterprise Manager – Segment Advisor

- DB-Startseite → Performance → Advisors Home → Segment Advisor

Zentrale Advisory > Segment Advisor Task: SEGMENTADVISOR_TASK1 > Empfehlung-Detail für Tablespace: EXAMPLE

Empfehlung-Detail für Tablespace: EXAMPLE

Die folgende Tabelle enthält die Informationen zu dem Speichersplatz, der für die unorganisierten Segmente in dem gewählten Tablespace freigegeben werden kann. Oracle empfiehlt, dass diese Segmente verkleinert, reorganisiert oder kompaktiert werden, damit der verschwendete Platz freigegeben wird. Wählen Sie das Segment, um die Empfehlung zu implementieren.

Aufgabename: SEGMENTADVISOR_TASK1
Status: COMPLETED
Ausführungszeit (Sekunden): 9
Zeitgrenze (Minuten): UNLIMITED
Erstellt: 23.05.2020 10:48:07 MESZ
Beendet: 23.05.2020 10:48:16 MESZ

Schema:
Segment:
Partition:

Minimale freigegebene Speicherplatz (MB):
Empfehlung: Alle Typen
Suchen:

Verkleinern Reorganisieren Kompaktieren

Alle auswählen Nicht auswählen

Auswahl	Schema	Segment	Empfehlung	Freigegebener Speicherplatz (MB)	Zugewiesener Speicherplatz (MB)	Belagter Speicherplatz (MB)	Segmenttyp
<input checked="" type="checkbox"/>	DOAGDB2020	SALES	Verkleinern	6.204,31	15.888,00	23.583,60	TABLE

Segment verkleinern: Optionen

Bei dem Verkleinern wird fragmentierter Speicherplatz kompaktiert und optional Speicherplatz freigegeben. Der Verkürzungsabgang dauert etwas länger und wird als Job geplant.

[SQL anzeigen](#) [Abbrechen](#) [Implementieren](#)

☒ **Segmente kompaktieren und Platz freigeben**
Mit dieser Option werden zuerst die Segmente kompaktiert und dann der wiederhergestellte Platz für den Tablespace freigegeben. Während der kurzen Phase der Platzfreigabe werden Cursor, die diese Segmente referenzieren, möglicherweise inaktiviert und Abfragen für das Segment können betroffen sein.

☐ **Segmente kompaktieren**
Beim Kompaktieren werden Segmente kompaktiert, ohne den wiederhergestellten Platz freizugeben. Nach dem Kompaktieren der Daten kann der wiederhergestellte Platz schnell freigegeben werden, indem die Option Segmente kompaktieren und Platz freigeben ausgewählt wird.

[SQL anzeigen](#) [Abbrechen](#) [Implementieren](#)

Reorganisation von Tabellen

9 28.05.20 DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

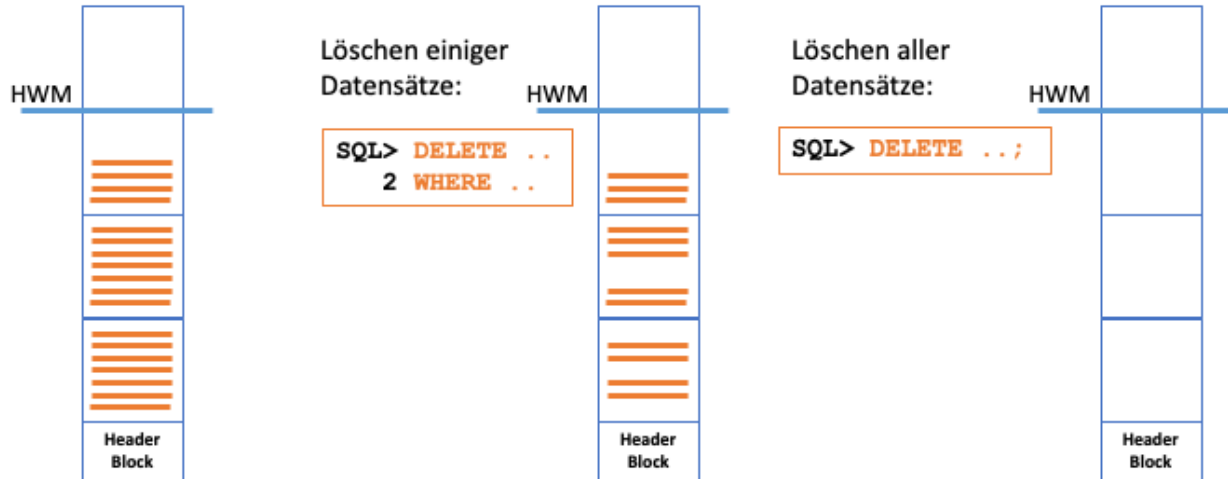
trivadis

Warum sollte man eine Tabelle reorganisieren?

- Heruntersetzen der High-Water-Mark nach größeren Löschaktionen
- Eliminieren von Row-Chaining und Row-Migration
- Ändern der Blockparameter PCTFREE und INITRANS
 - PCTFREE = Wieviel Platz (%) wird in einem Block für Updates reserviert (Default: 10%)
 - INITRANS = initiale Anzahl von "Transaktionslots" in einem Block, grob: "maximale Anzahl von gleichzeitigen Transaktionen mit diesem Block" (Default: 1 für Tabellen, 2 für Indizes), Änderung nur selten erforderlich
- Verschieben von Segmenten in einen anderen Tablespace
- Daten nach bestimmten Kriterien sortiert ablegen

Was ist die High-Water-Mark?

- High-Water-Mark (HWM) = Obergrenze der Blöcke eines Segments, die jemals genutzt wurden



11

28.05.20

DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

Wo liegt das Problem mit der High-Water-Mark?

- Platzverbrauch => Platz- und Zeitbedarf für Backups
- Full-Table-Scans => beim Full-Table-Scan werden alle Blöcke bis zur HWM gelesen
- Beispiel:

Aktion	#rows	Dauer FTS	#physical reads
Tabelle anlegen	12 Mio	0,26 sek	100384
Delete von 3 Mio Datensätzen	9 Mio.	0,20 sek	100385
Delete der kompletten Tabelle	0	0,18 sek.	100384
Truncate auf die Tabelle	0	0,01 sek.	10

- "Truncate table" ist ein DDL-Befehl (kein Rollback möglich), der eine Tabelle komplett löscht und den Speicherplatz wieder freigibt

Komplettes Beispiel:
hwm_delete_test.sql

12 28.05.20 DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

Beim Test wurde vor jedem SELECT ("select /* +FULL */ count(*) from testuser.t;") mittels "alter system flush buffer_cache;" der Buffer-Cache geleert.

Wie kann man das analysieren? Füllgrad der Blöcke

```
SQL> set serveroutput on
declare
v_unformatted_blocks number;
[...]
v_full_bytes number;
begin
  dbms_space.space_usage ('TESTUSER', 'T', 'TABLE', v_unformatted_blocks, v_unformatted_bytes,
v_fs1_blocks, v_fs1_bytes, v_fs2_blocks, v_fs2_bytes, v_fs3_blocks, v_fs3_bytes, v_fs4_blocks,
v_fs4_bytes, v_full_blocks, v_full_bytes);
  dbms_output.put_line('FS1 Blocks ( 0-25% free) = '||v_fs1_blocks);
  dbms_output.put_line('FS2 Blocks (25-50% free) = '||v_fs2_blocks);
  dbms_output.put_line('FS3 Blocks (50-75% free) = '||v_fs3_blocks);
  dbms_output.put_line('FS4 Blocks (75-99% free) = '||v_fs4_blocks);
  dbms_output.put_line('Full Blocks          = '||v_full_blocks);
end;
/
```

```
FS1 Blocks ( 0-25% free) = 0
FS2 Blocks (25-50% free) = 66806
FS3 Blocks (50-75% free) = 1331
FS4 Blocks (75-99% free) = 99
Full Blocks          = 1
```

Skript:
auswertung_beispiel.sql

13

28.05.20

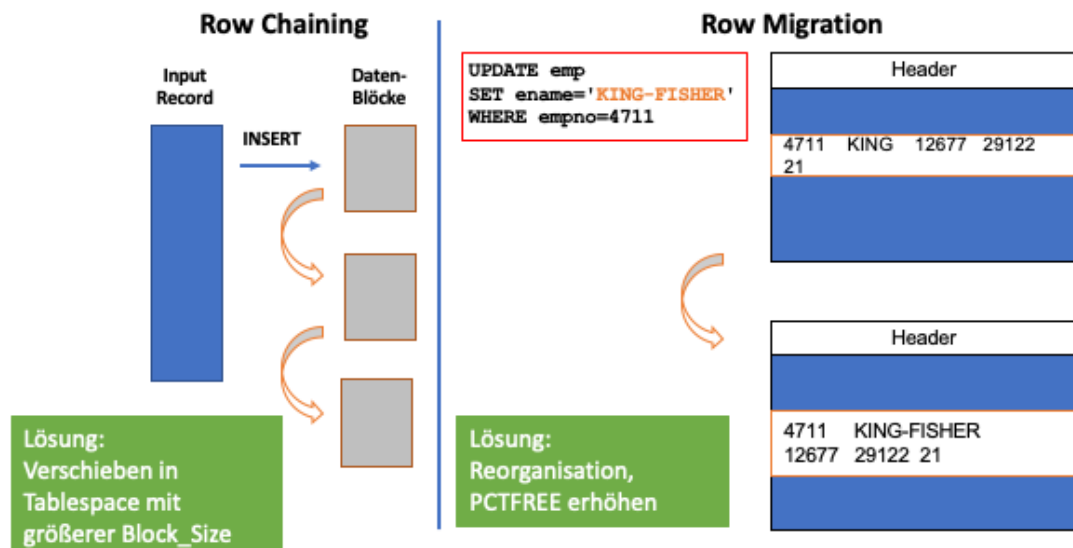
DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

```
SQL> set serveroutput on
declare
v_unformatted_blocks number;
v_unformatted_bytes number;
v_fs1_blocks number;
v_fs1_bytes number;
v_fs2_blocks number;
v_fs2_bytes number;
v_fs3_blocks number;
v_fs3_bytes number;
v_fs4_blocks number;
v_fs4_bytes number;
v_full_blocks number;
v_full_bytes number;
begin
  dbms_space.space_usage ('TESTUSER', 'T', 'TABLE', v_unformatted_blocks,
v_unformatted_bytes, v_fs1_blocks, v_fs1_bytes, v_fs2_blocks,
v_fs2_bytes, v_fs3_blocks, v_fs3_bytes, v_fs4_blocks, v_fs4_bytes, v_full_blocks,
v_full_bytes);
  dbms_output.put_line('FS1 Blocks ( 0-25% free) = '||v_fs1_blocks);
  dbms_output.put_line('FS2 Blocks (25-50% free) = '||v_fs2_blocks);
  dbms_output.put_line('FS3 Blocks (50-75% free) = '||v_fs3_blocks);
  dbms_output.put_line('FS4 Blocks (75-99% free) = '||v_fs4_blocks);
  dbms_output.put_line('Full Blocks          = '||v_full_blocks);
end;
```

/

Was sind Row-Chaining und Row-Migration (1)



14 28.05.20 DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

Row-Chaining:

- Datensatz ist größer als der in einem Datenbank-Block zur Verfügung stehende Platz
- Der Block wird über mehrere Blöcke verteilt
- Beim Lesen müssen ggf. mehrere Blöcke gelesen werden (=> abhängig davon, welche Spalten gelesen werden => "richtige" Reihenfolge der Spalten kann I/O reduzieren)
- Lösungen:
 - Größere DB-Blöcke (=> ALTER TABLE MOVE .. OTHER_TABLESPACE)
 - Spaltenreihenfolge anpassen (=> DBMS_REDEFINITION)
 - PCTFREE verkleinern (Vorsicht!)
 -

Row-Migration

- Nach einem Update passt ein Datensatz nicht mehr in einen Block
 - => Datensatz wird verschoben, aber ROWID bleibt
 - => beim Lesen müssen (mindestens) 2 Blöcke gelesen werden
- Lösungen
 - - PCTFREE vergrößern (Vorsicht!)
 - - Reorg der Tabelle

Analyse von Row-Chaining & Row-Migration

```
SQL> ANALYZE TABLE <owner>.<table_name> LIST CHAINED ROWS;
```

- Vorher die Tabelle CHAINED_ROWS anlegen (\$ORACLE_HOME/rdbms/admin/utlchain.sql)
 - Sonst gibt es den Fehler "ORA-01495: specified chain row table not found"
- "analyze .. list chained rows" schreibt die Row-IDs der betroffenen Datensätze in die Tabelle CHAINED_ROWS
- (Statistische) Informationen gibt es auch in V\$SYSSTAT:
"table fetch continued row"

```
SQL> desc CHAINED_ROWS
Name                               Null?  Type
-----
OWNER_NAME                         VARCHAR2(128)
TABLE_NAME                         VARCHAR2(128)
CLUSTER_NAME                       VARCHAR2(128)
PARTITION_NAME                     VARCHAR2(128)
SUBPARTITION_NAME                  VARCHAR2(128)
HEAD_ROWID                         ROWID
ANALYZE_TIMESTAMP                   DATE
```

15

28.05.20

DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

Wenn die Tabelle CHAINED_ROWS in einem anderen Schema liegt:

```
ANALYZE TABLE ... LIST CHAINED ROWS into <schema>.<tabelle>;
```

Anzeige der Gesamtzahl "chained rows" (genauer: Index-Zugriffe auf chained-rows):

```
SQL> SELECT name,value FROM v$sysstat
       2 WHERE name = 'table fetch continued row';
```

NAME	VALUE

table fetch continued row	49,176

Row-Chaining & Row-Migration - Beispiel

```
SQL> analyze table t list chained rows;
SQL> select count(*) from chained_rows where table_name='T';
      0

SQL> update t set ...;

SQL> analyze table t list chained rows;
SQL> select count(*) from chained_rows where table_name='T';
      931

SQL> alter table t move;

SQL> delete chained_rows;
SQL> analyze table t list chained rows;
SQL> select count(*) from chained_rows where table_name='T';
      0
```

Skript:
row_chaining.sql

Methoden für die Reorganisation von Tabellen

- **Online**

- ALTER TABLE .. MOVE ("Online" nur für die Enterprise Edition)
- SHRINK SPACE
- DBMS_REDEFINITION (Enterprise Edition)

- **Offline**

- CREATE TABLE AS SELECT
- Data Pump Export/Import

ALTER TABLE .. MOVE (1)

- Reorganisation einer Tabelle oder eines LOB-Segments
 - In anderen Tablespace verschieben
 - Storage Parameter, etc. ändern
- MOVE TABLE unterstützt R/I-Constraints
- Temporär wird der doppelte Platz benötigt
- Row-IDs ändern sich
- Beispiele:

```
SQL> ALTER TABLE dept MOVE  
2     TABLESPACE hr_data STORAGE (INITIAL 256K NEXT 256K) PCTFREE 0;
```

- Online-Move (seit Version 12.2, Enterprise Edition erforderlich)

```
SQL> ALTER TABLE emp MOVE ONLINE TABLESPACE hr_data;
```

ALTER TABLE .. MOVE (2) – Was passiert mit den Indizes?

- **Problem:** wenn ein Datensatz verschoben wird, dann ändert sich die Row-ID und Indizes werden "UNUSABLE"

- Beim "einfachen MOVE ONLINE" werden die Indizes automatisch neu aufgebaut

```
SQL> ALTER TABLE emp MOVE ONLINE TABLESPACE hr_data;
```

- Wenn ein "Online Move" mit Komprimierung verbunden wird (ACO erforderlich), dann werden die Indizes nicht automatisch neu aufgebaut → mit "UPDATE INDEXES" arbeiten

```
SQL> ALTER TABLE sh.orders MOVE ONLINE COMPRESS  
2 UPDATE INDEXES TABLESPACE ts_new;
```

- Beim "Offline Move" werden die Indizes "UNUSABLE" → mit "UPDATE INDEXES" arbeiten

- **Tipp:** Immer mit "UPDATE INDEXES" arbeiten

ALTER TABLE .. MOVE (3) - Beispiel

```
SQL> select num_rows,blocks from user_tab_statistics where table_name='T';
```

NUM_ROWS	BLOCKS
6000000	51068

```
SQL> delete t where mod(id,4)=0;
```

```
1500000 rows deleted.
```

```
SQL> commit;
```

```
Commit complete.
```

25% der Datensätze
werden gelöscht

```
SQL> exec dbms_stats.gather_table_stats(ownname=>user,tabname=>'T',estimate_percent=>100);  
PL/SQL procedure successfully completed.
```

```
SQL> select num_rows,blocks from user_tab_statistics where table_name='T';
```

NUM_ROWS	BLOCKS
4500000	51068

ALTER TABLE .. MOVE (4) - Beispiel

```
SQL> alter table t move;  
Table altered.
```

```
SQL> exec dbms_stats.gather_table_stats(ownname=>user,tabname=>'T',estimate_percent=>100);  
PL/SQL procedure successfully completed.
```

```
SQL> select num_rows,blocks from user_tab_statistics where table_name='T';  
NUM_ROWS    BLOCKS  
-----  
4500000      38070
```

Skript:
table_move_test.sql

Reorganisation (MOVE) von Tabellen-Partitionen

- Es gelten (im Wesentlichen) die gleichen Regeln wie für Tabellen
- Online Move:
 - Indizes werden NICHT automatisch aktualisiert

```
SQL> ALTER TABLE emp MOVE PARTITION hire_2011 TABLESPACE newdatatbs ONLINE;
```

- Deshalb besser:

```
SQL > ALTER TABLE emp  
2 MOVE PARTITION hire_2011 TABLESPACE newdatatbs UPDATE INDEXES ONLINE;
```

- Einschränkungen:
 - Kein paralleles DML gleichzeitig
 - Kein Supplemental Logging
 - Wenn "MOVE" mit Komprimierung verbunden wird, dann ist "ACO" erforderlich

SHRINK SPACE (1)

- Online Segment Shrink erlaubt es, Tabellen, Indizes, LOBs, IOTs und Materialized Views online zu verkleinern, inkl. Heruntersetzen der Highwater-Mark
- Voraussetzung: Row Movement auf der Tabelle muss aktiviert sein

```
SQL> ALTER TABLE .. ENABLE ROW MOVEMENT;
```

- Benötigt keinen zusätzlichen Platz während der Reorganisation
- Locks:
 - Rows-Locks beim Verschieben von Rows
 - Exklusiver Lock auf die Tabelle beim Anpassen der HWM

Die Shrink-Operation läuft in *zwei Phasen* ab:

Phase 1:

Komprimierung der Objekte: In dieser Phase werden die Rows soweit wie möglich an den Anfang des Segmentes verschoben. Weil dabei die ROWID ändert, muss ROW MOVEMENT auf Objektebene aktiviert sein und Trigger, die auf der ROWID basieren, müssen disabled sein. Während der Komprimierung gibt es nur DML-Locks auf die betroffenen Rows, resp. Blocks.

Phase 2:

Verschiebung der HWM (High Water Mark): Die HWM wird soweit wie möglich verschoben und der vom Segment nicht mehr benötigte Platz wird dealloziert und im Tablespace für andere Objekte wieder zur Verfügung gestellt. Während dieser Phase gibt es einen exklusiven Lock auf das Objekt.

Die beiden Phasen können auch separat ausgeführt werden, wenn z.B. das Locking von Phase 2 auf später verschoben werden soll:

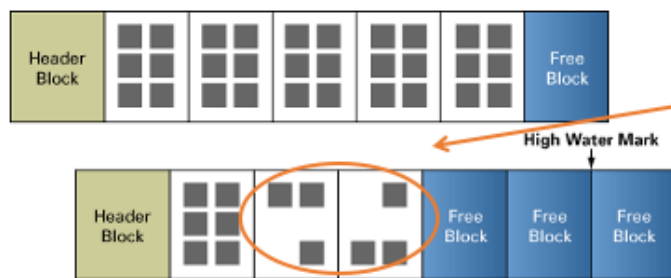
Phase 1:

Nur Komprimierung -> ALTER TABLE scott.emp SHRINK SPACE COMPACT;

Phase 2:

HWM verschieben -> ALTER TABLE scott.emp SHRINK SPACE;

SHRINK SPACE (2)



Deleted Records:

- Platz bleibt alloziert
Inserts mit Direct Path Inserts
werden hinter der HWM platziert
Platzverschwendung!
- HWM bleibt unverändert
Fulltable-Scantime!

```
ALTER TABLE scott.emp
SHRINK SPACE COMPACT;
```



```
ALTER TABLE scott.emp
SHRINK SPACE;
```



24 28.05.20

DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

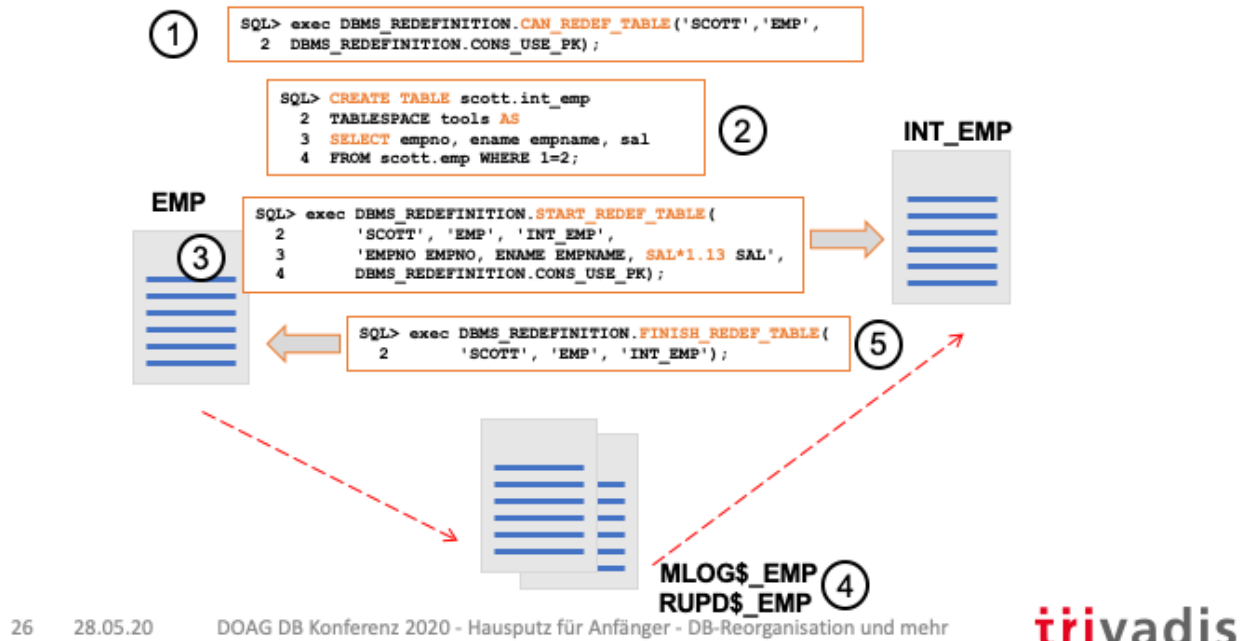
Einschränkungen von "SHRINK SPACE"

- Tabellen in Clusters
- Tabellen mit LONG-Attributen
- Tabellen mit on-commit Materialized Views
- Tabellen mit Rowid Based Materialized Views
- Lob Indizes

DBMS_REDEFINITION (1)

- DBMS_REDEFINITION erlaubt die Online-Reorganisation und -Redefinition von Tabellen und Partitionen (Enterprise Edition erforderlich)
 - Ändern von Storage-Parametern, Tablespace
 - Zufügen, Löschen, Umbenennen oder Ändern von Attributen
 - Partitionieren (geht mit 19c einfacher)
 - Partitionierung rückgängig machen
- Defragmentierung (einfacher mit Shrink)
- Funktionsprinzip ("materialized view on a pre-built table")
 1. Aufbau einer Interims-Tabelle mit neuer Struktur
 2. Start der Redefinition "by_key" oder "by_rowid"
 3. Abschluss der Redefinition (kurzer Lock der Originaltabelle)
= Tausch der Namen "Interims-Tabelle ↔ Originaltabelle"

DBMS_REDEFINITION (2)



26 28.05.20 DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

Mittels DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS können auch die zur Tabelle gehörigen Trigger, Constraints, Indizes, Grants etc. auf die Interim-Tabelle kopiert werden.

Folgende "Hilfstabellen" werden beim Reorg mittels DBMS_REDEFINITION angelegt:

- mlog\$_emp: Snapshot Log (permanente Tabelle) der Tabelle EMP, welche alle DML auf EMP während der Reorg hält
- rupd\$_emp: temporäre Tabelle für Session-Daten

Legende:

1. Prüfen ob die Tabelle neu organisiert werden kann
2. Interimstabelle anlegen
3. Starten, die Daten werden in die Interimstabelle repliziert
4. Im Hintergrund werden zusätzliche Tabellen angelegt
5. Stoppen, die Tabellenbezeichnungen werden getauscht, die Objekt-ID bleibt die gleiche

Anzeige von laufenden DBMS_REDEFINITION Aktionen:

```
SQL> SELECT object_name, interim_object_name, object_type  
2 FROM dba_redefinition_objects;
```

OBJECT_NAME	INTERIM_OBJECT_NAME	OBJECT_TYPE
-----	-----	-----

EMP

INT_EMP

TABLE

DBMS_REDEFINITION (3)

- Mittels **DBMS_REDEFINITION.COPY_TABLE_DEPENDENTS** können auch die zur Tabelle gehörigen Trigger, Constraints, Indizes, Grants etc. auf die Interim-Tabelle kopiert werden.
 - !! Spaltennamen müssen passen
- Seit Oracle 12.2 hat DBMS_REDEFINITION eine "Rollback"-Funktionalität
 - Nach dem "FINISH_REDEF_TABLE" erfolgt die Replikation in die umgekehrte Richtung
 - !! Spaltennamen müssen passen
- Aktivierung:

```
DBMS_REDEFINITION.START_REDEF_TABLE (  
  UNAME => 'SCOTT',  
  ORIG_TABLE => 'EMP', INT_TABLE => 'INT_EMP',  
  ENABLE_ROLLBACK => TRUE);
```

- "Rollback" (= Umbenennung der Tabellen rückgängigmachen):
DBMS_REDEFINITION.ROLLBACK

Offline-Methoden: CTAS & DataPump

- **Create Table As Select (CTAS)**

```
SQL> CREATE TABLE emp_new AS SELECT * FROM emp ORDER BY deptno;
```

- Mit "ORDER BY" können die Daten (momentan) sortiert gespeichert werden
→ I/O-Optimierung bei Zugriffen auf bestimmte Attribute
- Anschließend Tabelle umbenennen
- Vorteil: man kann die Reihenfolge der Spalten ändern, Spalten weglassen etc.
- Problem: Abhängigkeiten (Foreign Keys, Indizes)
- → wird eher selten genutzt

- **DataPump**

1. Exportieren der Tabelle
 2. Droppen (oder Truncate) der Tabelle
 3. Importieren der Tabelle
- Problem: Abhängigkeiten (Foreign Keys)

28 28.05.20 DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

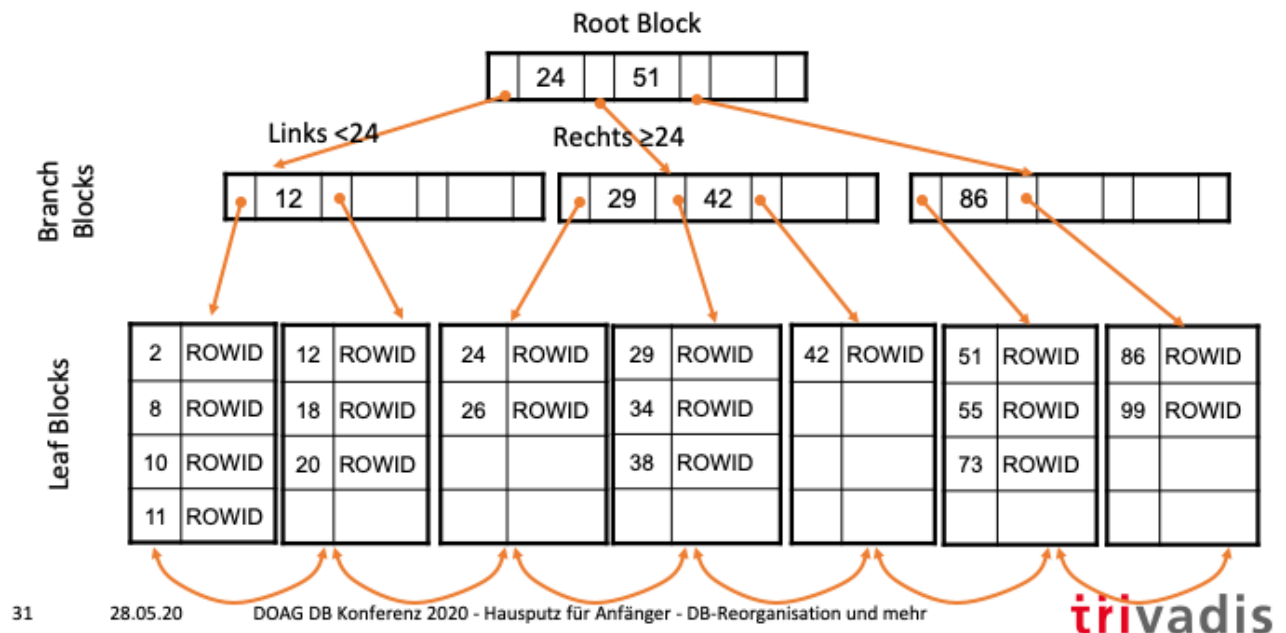
trivadis

Reorganisation von Indizes

Wann sollte man einen Index reorganisieren?

- 5 Oracle-Spezialisten, 6 Meinungen?
- **Generell:** man muss einen Index (im Allgemeinen) seltener reorganisieren, als man denkt
- Faustregel:
 - Wenn 20% der Datensätze der Tabelle gelöscht wurden
 - Wenn der Indexbaum eine Höhe von 4 (oder mehr) hat

Ein Blick auf einen B*Tree-Index



Wie analysiert man einen Index?

```
SQL> ANALYZE INDEX <owner>.<index_name> VALIDATE STRUCTURE;
```

- Benötigt einen exklusiven Lock auf den Index
- Ergebnis:

```
SQL> select height,lf_rows,lf_blks,br_rows,br_blks,del LF_ROWS,DEL_LF_ROWS_LEN  
2      from index_stats where name='I';
```

HEIGHT	LF_ROWS	LF_BLKs	BR_ROWS	BR_BLKs	DEL_LF_ROWS	DEL_LF_ROWS_LEN
3	4487523	20510	20509	48	487523	8283940

HEIGHT = Height of the B-Tree

LF_ROWS = Number of leaf rows (values in the index)

LF_BLKs = Number of leaf blocks in the B-Tree

BR_ROWS = Number of branch rows in the B-Tree

BR_BLKs = Number of branch blocks in the B-Tree

DEL_LF_ROWS = Number of deleted leaf rows in the index

DEL_LF_ROWS_LEN = Total length of all deleted rows in the index

Wie reorganisiert man einen Index?

- Zusammenführen der Leaf-Blöcke
 - Höhe des Index ändert sich nicht

```
SQL> ALTER INDEX <owner>.<index_name> COALESCE;
```

- Neu-Aufbau des Indexes
 - Nicht nur für einen Reorg, sondern auch für den Neuaufbau eines "UNUSABLE"-Indexes
 - Änderung der Storage-Parameter möglich
 - "REBUILD ONLINE" nur mit Enterprise Edition

```
SQL> ALTER INDEX idx_dept_deptno REBUILD ONLINE;
```

```
SQL> ALTER INDEX idx_dept_deptno REBUILD PCTFREE 5 TABLESPACE app_idx;
```

Beispiel: Ergebnis von INDEX COALESCE & REBUILD

```
SQL> alter index i coalesce;
Index altered.
SQL> analyze index i validate structure;
Index analyzed.
SQL> select height,lf_rows,LF_BLKs,BR_ROWS,BR_BLKs,DEL_LF_ROWS,DEL_LF_ROWS_LEN
2   from index_stats where name='I';
HEIGHT      LF_ROWS  LF_BLKs    BR_ROWS  BR_BLKs  DEL_LF_ROWS  DEL_LF_ROWS_LEN
-----
      3      4000000      9300    9299         48         0         0

SQL> alter index i rebuild;
Index altered.
SQL> analyze index i validate structure;
Index analyzed.
SQL> select height,lf_rows,LF_BLKs,BR_ROWS,BR_BLKs,DEL_LF_ROWS,DEL_LF_ROWS_LEN
2   from index_stats where name='I';
HEIGHT      LF_ROWS  LF_BLKs    BR_ROWS  BR_BLKs  DEL_LF_ROWS  DEL_LF_ROWS_LEN
-----
      3      4000000     9271    9270         19         0         0
```

Skript: Index_rebuild.sql

34 28.05.20 DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

Es gibt auch noch

"ALTER INDEX DEALLOCATE STORAGE .."

"ALTER INDEX SHRINK SPACE .."

Ungültige Objekte in der Datenbank

35 28.05.20 DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

Ungültige Objekte in der Datenbank – (1)

- Ein Objekt (View, Synonym, PL/SQL-Code) wird ungültig, wenn
 - Notwendige Rechte fehlen
 - Ein erforderliches Objekt fehlt (View ohne zugrundeliegende Tabelle, Synonym ohne referenziertes Objekt etc.)
 - Tippfehler im PL/SQL-Code
 - ...
- Ungültige Objekte stören nicht, ..
 - .. So lange man nicht probiert, sie zu verwenden
- **Tipps:**
 - Ziel sollte "0 ungültige Objekte" sein → regelmäßig prüfen, wenn Re-Compile erfolglos: Objekte ggf. löschen
 - Vor einem DB-Upgrade: ungültige Objekte prüfen und "merken" und nach dem Upgrade abgleichen → sind es mehr, gibt es ggf. ein Problem 😊

Ungültige Objekte in der Datenbank – (2)

- Gibt es ungültige Objekte in der Datenbank?

```
SQL> SELECT owner,object_type,object_name FROM dba_invalid_objects;
```

- Re-Compile probieren (als SYS)

```
SQL> @?/rdbms/admin/utlrp.sql
```

- Ergebnis prüfen und Fehler suchen

```
SQL> SELECT owner,object_type,object_name FROM dba_invalid_objects;
```

```
SQL> SELECT * FROM dba_errors;
```

Nicht mehr genutzte User/Schemata

38 28.05.20 DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

Nicht mehr genutzte Datenbank-User (1)

- Offene, aber nicht mehr genutzte Datenbank-Benutzer, können ein Sicherheitsrisiko sein!
- Seit Oracle 12c (12.1) protokolliert Oracle die letzte erfolgreiche Anmeldung eines Benutzers
 - Spalte LAST_LOGIN in DBA_USERS

```
oracle@training19c:~/ [TVDNCDB] sqlplus hr/hr

SQL*Plus: Release 19.0.0.0.0 - Production on Fri May 22 15:06:34 2020
Version 19.7.0.0.0
Copyright (c) 1982, 2020, Oracle. All rights reserved.

Last Successful login time: Tue Mar 03 2020 14:12:14 +02:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.7.0.0.0
SQL>
```

Nicht mehr genutzte Datenbank-User (2)

- Seit Oracle 12.2 können inaktive Datenbank-Benutzer automatisch gesperrt werden
 - Zuweisung eines entsprechenden Profils
 - ACCOUNT_STATUS in DBA_USERS: "LOCKED(TIMED)"

- "INACTIVE_ACCOUNT_TIME"
 - Von 15 (Tagen) bis 24855 Tage (= ca. 68 Jahre)
 - Default: "UNLIMITED"

```
SQL> CREATE PROFILE prf_default_user LIMIT
2  FAILED_LOGIN_ATTEMPTS 5
3  PASSWORD_LIFE_TIME 30
4  PASSWORD_REUSE_TIME 60
5  PASSWORD_LOCK_TIME UNLIMITED
6  PASSWORD_GRACE_TIME 7
7  INACTIVE_ACCOUNT_TIME 95;
```

- Automatisches Sperren von Accounts ist nicht immer anwendbar
- "Inaktivitätszeitraum" ist applikationsspezifisch
- ==> ggf. nicht sperren sondern nur auswerten und prüfen

Nicht mehr genutzte Schemata

- Nicht mehr genutzte Schemata
 - Belegen unnötigen Platz in den Datendateien
 - Verlängern die Dauer von Datenbank-Backups
- Die zugehörigen (offenen) Datenbank-Benutzer sind ein Sicherheitsrisiko
- **Tipp:** reine Schema-Benutzer (d.h. Benutzer die nur Objekte haben, die aber nicht für die Anmeldung an die Datenbank genutzt werden) als "schema-only accounts anlegen" (seit 18c)

```
SQL> CREATE USER scott_data NO AUTHENTICATION;
```

- Zugriff über Proxy-User

```
SQL> ALTER USER scott_data GRANT CONNECT THROUGH scott;  
SQL> CONNECT scott[scott_data]/tiger
```

Wie kann man nicht mehr genutzte Schemata erkennen?

- Indizien (!) können sein
 - Alter der Statistiken UND keine Einträge in DBA_TAB_MODIFICATIONS
➔ d.h., dass es seit der Aktualisierung der Statistiken keine DML-Aktivitäten gegeben hat
 - Letzte Indexnutzung (in DBA_OBJECT_USAGE) liegt weit zurück
➔ d.h., dass es seitdem keine Abfragen gegeben hat, die einen Index genutzt haben
 - Letzter Login des Schema-Users und von Benutzern, die Rechte auf die Objekte des Schemas haben, liegen weit zurück
- Das sind alles nur Indizien!

Alte Log- und Trace-Dateien

Log- und Trace-Dateien – Aufräumen im ADR

- Im Laufe des Datenbank-Lebens fallen viele Log- und Trace-Dateien an
.. und werden nicht automatisch gelöscht
 - Log- und Trace-Dateien im ADR (Automatic Diagnostic Repository),
meist unter \$ORACLE_BASE/diag

```
SQL> show parameter diagnostic_dest
NAME                                TYPE                                VALUE
-----
diagnostic_dest string                /opt/oracle
```

```
oracle:/opt/oracle/diag/[XE] tree -d
.
[.]
|-- rdbms
|   |-- xe
|       |-- XE
|           |-- alert
[.].      [.]
|           |-- trace
|-- tnslnsr
|   |-- bourbaki
|       |-- listener
|           |-- alert
[.].      [.]
|           |-- trace
```

- Audit-Daten in \$ORACLE_BASE/audit
- Diese Dateien kann man
 - Löschen
 - Komprimieren oder
 - Archivieren

Aufräumen – ADRCI – (1)

- Die Datenbank-Instanz kann selbst aufräumen (löschen, "purge")
 - MMON-Prozess,
 - 1.Durchlauf 48 h nach dem Startup der Instanz, dann alle 7 Tage
- Gesteuert von 2 Policies
 - **LONGP_POLICY** (Default 365 Tage, für Dateien im alert, incident, stage, sweep und hm-Verzeichnis)
 - **SHORTP_POLICY** (Default 30 Tage, für Dateien im trace, cdump, utcsdump und ips-Verzeichnis)

Aufräumen – ADRCI – (2)

- Anzeigen und Ändern der Policies:

```
oracle@bourbaki:~/ [XE] adrci

ADRCI: Release 18.0.0.0.0 - Production on Sat May 23 15:57:42 2020
Copyright (c) 1982, 2018, Oracle and/or its affiliates. All rights reserved.
No ADR base is set
adrci> set base /opt/oracle
adrci> set home diag/rdbms/xe/XE
adrci> show control
ADR Home = /opt/oracle/diag/rdbms/xe/XE:
*****
ADRID                SHORTP_POLICY          LONGP_POLICY          LAST_MOD_TIME
[...]
```

ADRID	SHORTP_POLICY	LONGP_POLICY	LAST_MOD_TIME
3216998116	720	8760	2020-05-21 17:16:41

```
adrci> set control (SHORTP_POLICY=24)
adrci> set control (LONGP_POLICY=168)
```

Angabe in Stunden

Aufräumen – ADRCI (3)

- Manuelles Aufräumen gemäß den Policy-Einstellungen

```
adrci> purge
```

- Löschen aller Dateien im Trace-Verzeichnis älter als 60 Minuten

```
adrci> purge -age 60 -type trace
```

- Löschen aller Dateien > 10 MB

```
adrci> purge -size 10000000
```

Für den Listener kann seit Oracle 19c über zwei listener.ora-Parameter auch ein logrotate konfigurieren:

LOG_FILE_NUM_listener=3 ← wieviel Dateien sollen aufbewahrt werden (default nicht gesetzt => es wird nichts gelöscht)

LOG_FILE_SIZE_listener_name= bei welcher Größe erfolgt ein Rotate (Default 300M)

Aufräumen mit eigenen Shell-Skripten (Beispiele)

- Meist nutzt man Shell-Skripte zum "Housekeeping"
- Löschen von Audit-Dateien älter 30 Tage

```
oracle> find /opt/oracle/audit -name "*.aud" -mtime +30 -exec rm {} \;
```

- Komprimieren und Umbenennen der *.log-Dateien (z.B. alert.log und listener.log)

```
oracle> find /opt/oracle/diag/ -name "*.log" -exec mv {} {}_`date +%Y%m%d`  
\;  
Oracle> find /opt/oracle/diag/ -name "*.log_*" -exec gzip {} \;
```

- Löschen von alten Trace-Dateien

```
oracle> find /opt/oracle/diag -name "*.tr*" -mtime +30 -exec rm {} \;
```

.. Insgesamt: eine große Spielwiese für Shell-Programmierer ☺

Zusammenfassung & Weitere Informationen

49 28.05.20 DOAG DB Konferenz 2020 - Hausputz für Anfänger - DB-Reorganisation und mehr

trivadis

Zusammenfassung

- Reorganisation der Datenbank (Tabellen, Indizes) ist seltener nötig als man denkt
- Nicht "pauschal" reorganisieren, sondern vorher analysieren
- Es gibt eine Vielzahl von Reorganisationsmethoden, die oft mehr können als "nur" reorganisieren
- Regelmäßig prüfen, ob alle Benutzer und Schemata noch benötigt werden
- Aufräumen der Log- und Trace-Dateien nicht vergessen

Weitere Informationen (1) – MOS-Notes

- [How to View High Water Mark - Step-by-Step Instructions \(Doc ID 262353.1\)](#)
- [Row Chaining and Row Migration \(Doc ID 122020.1\)](#)
- [How to Identify, Avoid and Eliminate Chained and Migrated Rows ? \(Doc ID 746778.1\)](#)
- [How to Find and Eliminate Migrated and Chained Rows \(Doc ID 102989.1\)](#)
- [How to Reorganize a Table \(Doc ID 151588.1\)](#)
- [Optimizing Database disk space using Alter table shrink space/move compress \(Doc ID 1173241.1\)](#)
- [SEGMENT SHRINK and Details. \(Doc ID 242090.1\)](#)
- [Generate Script to Shrink Segment Advisor Recommendations \(Doc ID 1171054.1\)](#)
- [DBMS_REDEFINITION ONLINE REORGANIZATION OF TABLES \(Doc ID 149564.1\)](#)
- [Index Rebuild, the Need vs the Implications \(Doc ID 989093.1\)](#)
- [Script to investigate a b-tree index structure \(Doc ID 989186.1\)](#)
- [Retention Policy for ADR \(Doc ID 564269.1\)](#)
- [Which Files Are Part Of SHORTP_POLICY And LONGP_POLICY In ADR? \(Doc ID 975448.1\)](#)
- [Why Are My Listener Logs & Traces Not Purged By The ADR? \(Doc ID 1438242.1\)](#)

Weitere Informationen (2) - Webseiten

- <https://carlos-sierra.net>
 - <https://carlos-sierra.net/2017/07/12/script-to-identify-index-rebuild-candidates-on-12c/>
 - <https://carlos-sierra.net/2014/07/18/free-script-to-very-quickly-and-cheaply-estimate-the-size-of-an-index-if-it-were-to-be-rebuilt/>
- <https://richardfoote.files.wordpress.com>
 - <https://richardfoote.wordpress.com/2014/03/05/index-rebuild-the-need-vs-the-implications-support-note-989093-1-getting-better/>
 - <https://richardfoote.files.wordpress.com/2008/02/coalesce-vs-shrink-demo.pdf>
- <https://oracledbwr.com/improve-performance-oracle-12c-table-reorganization/>
- <https://dba12c.wordpress.com/2016/09/08/oracle-table-reorganization-script/>
- <https://oracle-base.com/articles/misc/reclaiming-unused-space>
- Oracle 11g files housekeeping methods - <http://www.dadbm.com/oracle-11g-files-housekeeping-methods/>
- Row Chaining & Row Migration: https://blog.toadworld.com/Row_Chaining_and_Migration

Fragen & Antworten

Markus Flechtner

markus.flechtner@trivadis.com

Telefon +49 211 5866 64725



@markusdba



Vortrag + Skripte auf
www.markusdba.de.net

BASEL | BERN | BRUGG | BUKAREST | DÜSSELDORF | FRANKFURT A.M. | FREIBURG I.B.R. | GENÈVE
HAMBURG | KOPENHAGEN | LAUSANNE | MANNHEIM | MÜNCHEN | STUTTGART | WIEN | ZÜRICH

trivadis



Eine **WELT** ermöglichen,
in der **intelligente IT**
LEBEN und **ARBEITEN**
völlig selbstverständlich
erleichtert.