SIG Database 24.2.2011

Stored Outlines - SQL Profiles - SQL Plan Management

Dr. Günter Unbescheid
Database Consult GmbH
Jachenau

Database Consult GmbH

- Gegründet 1996
- Kompetenzen im Umfeld von ORACLE-basierten Systemen
- Tätigkeitsbereiche
  - Tuning, Installation, Konfiguration
  - Security, Identity Management
  - Expertisen/Gutachten
  - Support, Troubleshooting, DBA-Aufgaben
  - Datenmodellierung und –design
  - Datenbankdesign, Systemanalysen
  - Programmierung: SQL,PL/SQL,Java, JSP, ADF, BC4J
  - Workshops
  - www.database-consult.de

Prolog

Die System- und Zugriffsoptimierung ist nach wie vor eines wichtigsten Betätigungsfelder für Datenbank-Verantwortliche, das – immer noch – einen beträchtlichen Teil der Arbeitszeit beansprucht.

Dabei gilt ein besonderes Augenmerk der **dauerhaften Stabilität** des Zugriffsverhaltens von Anwendungen.

Besonders schwierig wird dieses Unterfangen, wenn die Eingriffsmöglichkeiten begrenzt sind, weil die Kontrolle über den Source-Code und die Schema-Strukturen nicht oder nur teilweise gegeben sind.

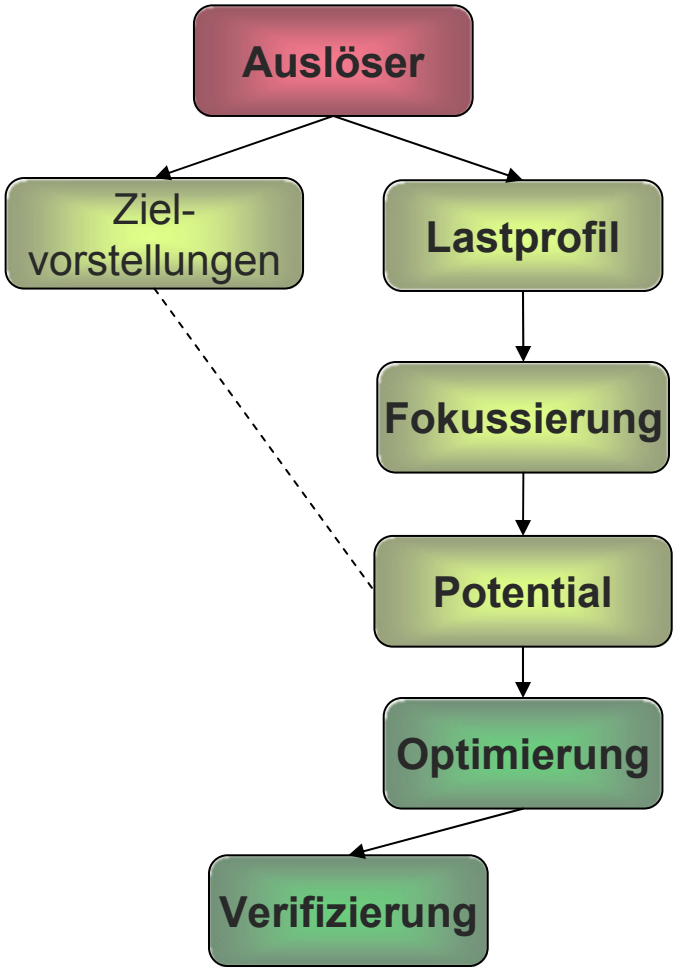
Agenda

- Einführung - Strategie und IST-Analyse
- Stored Outlines
- SQL Profiles
- SQL Baselines, SQL Plan Management

Strategie und Ist-Analyse

Der Wald und die Bäume.....

Lastprofil als Strategie



A **profile** is a tabular decomposition of response time, typically listed in descending order of component response time contribution.

C.Millsap

- Ermittlung der "richtigen" SQL's
- Präzise IST-Analyse
- Ausräumen periferer Störquellen

AWR/ASH

- Automatic Workload Repository und Active Session History
  - eine von mehreren Möglichkeiten
- Reporting über
  - SQL-Skripte (?/rdbms/admin/awr* oder ash* [11g])
  - Package API DBMS_WORKLOAD_REPOSITORY über (table) functions
  - Format: HTML oder Text
- Kontexte
  - AWR, ASH, AWR_SQL (11g)
- In Anlehung an STATSPACK

Nagelprobe über AWR

WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
TESTDB	1698441787	TESTDB	1	10.2.0.4.0	NO	TestHost

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	5794	29-Jul-09 15:13:10	18	1.9
End Snap:	5813	30-Jul-09 10:00:05	50	6.5
Elapsed:		1,126.91 (mins)		
DB Time:		28.62 (mins)		

Report Summary

Cache Sizes

	Begin	End		
Buffer Cache:	4,096M	4,096M	Std Block Size:	8K
Shared Pool Size:	1,024M	1,024M	Log Buffer:	16,896K

Nagelprobe - AWR

Load Profile

	Per Second	Per Transaction
Redo size:	1,096,216.35	1,599,091.43
Logical reads:	5,111.90	7,456.91
Block changes:	1,293.45	1,886.81
Physical reads:	4.78	6.97
Physical writes:	141.33	206.17
User calls:	3.60	5.24
Parses:	2.33	3.39
Hard parses:	0.03	0.04
Sorts:	1.49	2.17
Logons:		
Executes:		
Transactions:		

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.91	In-memory Sort %:	100.00
Library Hit %:	99.51	Soft Parse %:	98.88
Execute to Parse %:	69.18	Latch Hit %:	99.99
Parse CPU to Parse Elapsd %:	41.84	% Non-Parse CPU:	99.86

Nagelprobe - AWR

Time Model Statistics

- Total time in database user-calls (DB Time): 10109,3s
- Statistics including the word "background" measure background process time, and so do not contribute to the DB time statistic
- Ordered by % or DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	8,395.21	83.04
DB CPU	3,432.53	33.95
PL/SQL execution elapsed time	932.80	9.23
parse time elapsed	16.37	0.16
connection management call elapsed time	13.33	0.13
hard parse elapsed time	10.07	0.10

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
db file parallel write	320,253	21,267	66	210.4	System I/O
log file parallel write	16,956	7,456	440	73.8	System I/O
log buffer space	5,988	4,550	760	45.0	Configuration
CPU time		3,433		34.0	
log file sync	2,109	1,739	825	17.2	Commit

Nagelprobe - AWR

Tablespace IO Stats

- ordered by IOs (Reads + Writes) desc

Tablespace	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
TBS_DATA_01	15,269	1	15.57	1.01	621,294	43	0	0.00
UNDOTBS1	179	0	90.73	1.00	206,014	14	3	496.67
TBS_INDEX_01	5,443	0	13.54	1.00	32,596	2	0	0.00
TBS_DATA_02	23,434	2	6.10	1.00	198	0	0	0.00
TBS_INDEX_02	18,734	1	0.58	1.00				
SYSTEM	1,645	0	18.54	1.83				
SYSAUX	799	0	10.21	1.01				

SQL Statistics

- SQL ordered by Elapsed Time
- SQL ordered by CPU Time
- SQL ordered by Gets
- SQL ordered by Reads
- SQL ordered by Executions
- SQL ordered by Parse Calls
- SQL ordered by Sharable Memory
- SQL ordered by Version Count
- Complete List of SQL Text

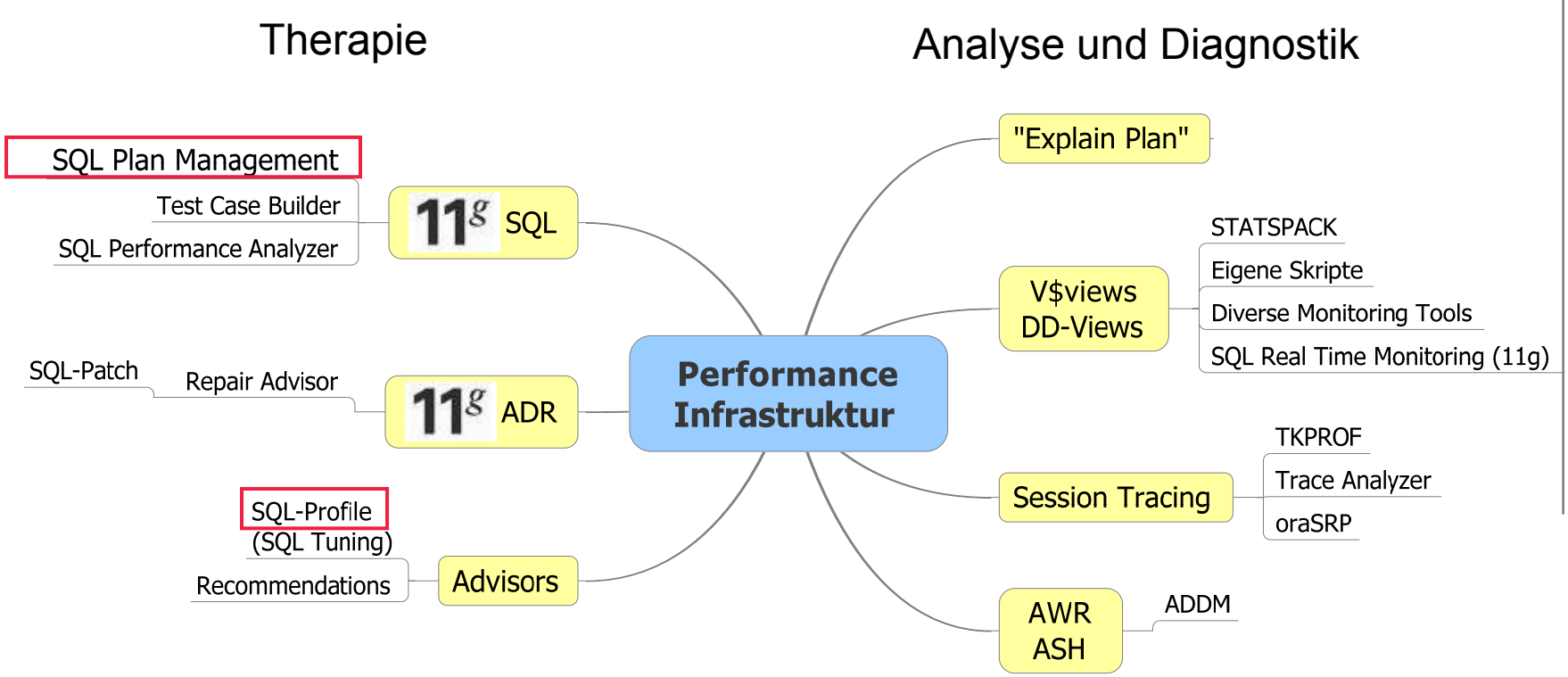
Nagelprobe – AWR

SQL ordered by Gets

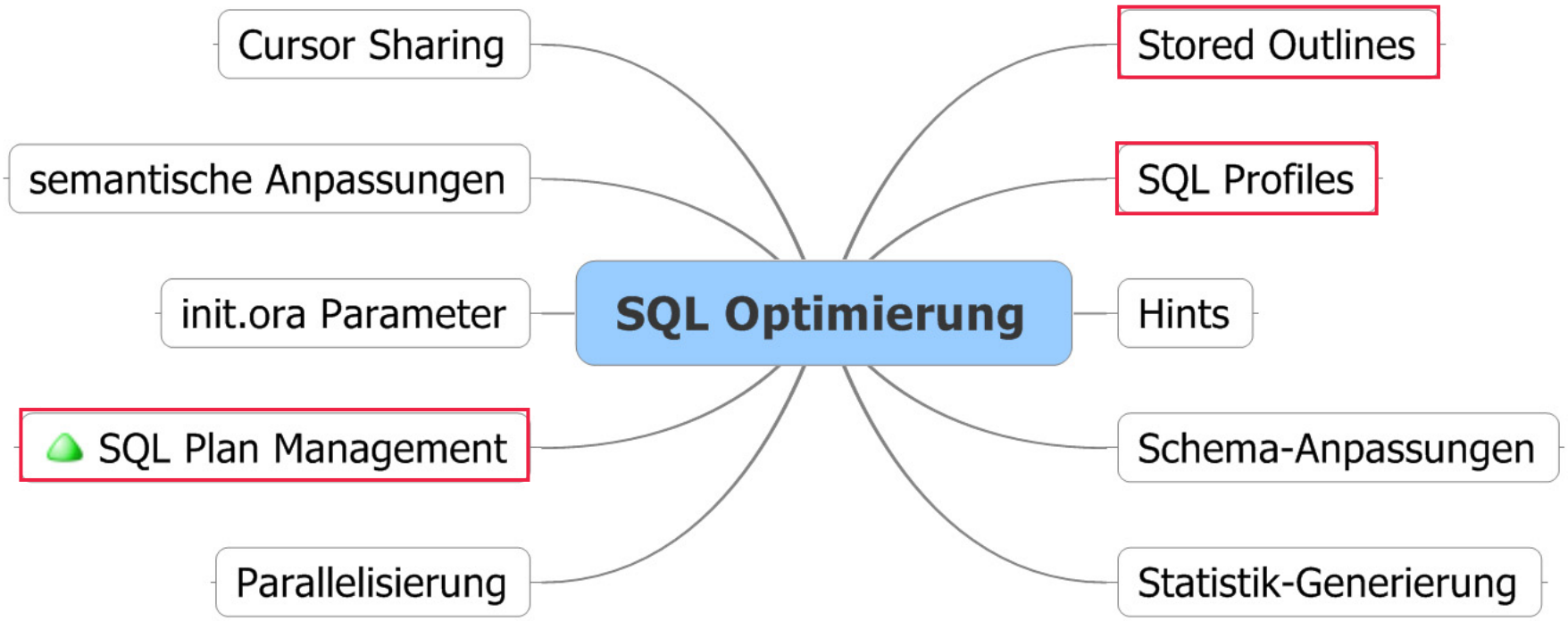
- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- Total Buffer Gets: 73,569,892
- Captured SQL account for 0.9% of Total

Buffer Gets	Executions	Gets per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	
72,765,346	1	72,765,346.00	98.91	3251.44	8148.50	8509rsz5k113u	BATCH
45,380,490	0		61.68	2754.62	7065.78	646xva575tbrt	XMODUL
306,948	4	76,737.00	0.42	10.04	17.36	c8khjbn635s3c	plsqldev.e
106,449	9,676	11.00	0.14	11.94	19.41	fszf36ykt382v	X_DELETE
104,967	1	104,967.00	0.14	5.08	5.13	572fbaj0fdw2b	sqlplus@s
94,263	18	5,236.83	0.13	31.61	68.51	a2g62xnwzmnq3	JDBC Thin

Infrastruktur



Tuning-Möglichkeiten SQL



Ausgewähltes Monitoring

- Genauere Fokusierung auf Kontexte – Drill Down
  - gewonnen aus dem generellen Lastprofil
  - Präzisierung der Problemanalyse durch AWR
- Manuelle SQL-Auswahl
  - explizit, AWR Reservoir/Stats-Tabellen, shared pool
- Tracing
  - Aufzeichung von DB-Aktivitäten über Zeiträume
  - rohe Trace-Dateien
  - Session, Modul, DB-Service, Client-ID
- Formatierung und Analyse
  - Aufbereitung der „rohen" Trace-Dateien

SQL-Analysen

- set autotrace – SQL*Plus
  - einfachste Methode für einzelne SQLs
  - Ausgabe steuerbar: Ergebnisse, Pläne und Statistiken
- explain plan – SQL
  - parsed Statement in aktueller Session!
  - nutzt plan_table, statement_id zur Unterscheidung
  - Formatierung per select oder dbms_xplan
  - Achtung (!) ggf. kein originaler Zugriffsplan
- dbms_xplan –Package zur formatierten Ausgabe
  - verarbeitet plan_table, AWR, cached cursor, SQL Plan Baselines, SQL Tuning Sets, Table-Functions
  - Ausgabe steuerbar über unterschiedliche Formatvorgaben

DBMS_XPLAN

<format>

Value	Description
basic	Displays only the minimum amount of information, basically only the operations and the objects on which they are executed
typical	Displays the most relevant information, basically everything except for alias, outline, and column projection information
serial	Like typical, except that information about parallel processing is not displayed
all	Displays all available information except the outline
advanced	Displays all available information

```
explain plan for <sql statement>
set linesize 110
-- Ausgabe von plan_table
select * from table(dbms_xplan.display());
-- Format der display Funktion
<table_name>,<statement_id>,<format>,<filter predicates>
-- Ausgabe des vorangehenden Statements aus library chache
select * from
table(dbms_xplan.display_cursor(null,null, 'all'));
-- Weitere Varianten
select * from
  table(dbms_xplan.display_cursor(<sqlid>, <child>,'advanced');
select * from table(dbms_xplan.display_awr(<sqlid>))
```


DBMS_XPLAN

<format> kann durch Zusätze erweitert (+) oder minimiert (-) werden

Value	Description
alias	Controls the display of the section containing query block names and object aliases.
bytes	Controls the display of the column Bytes in the execution plan table.
cost	Controls the display of the column Cost in the execution plan table.
note	Controls the display of the section containing the notes.
outline	Controls the display of the section containing the outline.
parallel	Controls the display of parallel processing information, specifically, the columns TQ, IN-OUT, and PQ Distrib in the execution plan table.
partition	Controls the display of partitioning information, specifically the columns Pstart and Pstop in the execution plan table.
peeked_binds	Controls the display of the section containing peeked bind variables. Since the current implementation of the SQL statement EXPLAIN PLAN does not perform bind variable peeking, this section was not shown in the previous examples.
predicate	Controls the display of the section containing filter and access predicates.
projection	Controls the display of the section containing column projection information.
remote	Controls the display of SQL statements executed remotely.
rows	Controls the display of the column Rows in the execution plan table.

```
SELECT * FROM
table(dbms_xplan.display(NULL,NULL,'typical -bytes -note',NULL));
```


Weitere SQL-Analysen

- SQL-Report auf Basis von statspack - sprepsql.sql
- SQL-Report aus AWR - awrsqrpt.sql
  - Zugriffsplan und Statistiken gewonnen aus Snapshots
- enhanced explain plan
  - für stand-alone SQL oder cached cursor
  - hervorragende Darstellung aller relevanten Daten:
    - Metadaten , Zugriffspläne, Traces, Statisitken ...
  - über ML „Diagnostic Tools Catalog [ID 559339.1]”

Enhanced explain plan

General

- Environment
- SQL Identification
- SQL Statement
- Observations
- Constraints
- Tablespaces
- DBMS STATS Setup
- System Statistics
- Optimizer Environment
- Bug Fix Control Environment
- Initialization Parameters
- Tool Configuration Parameters

Bind Variables

- Peeked Binds
- For Latest Execution

SQL Execution

- Segment Statistics
- Session Statistics
- Session Events

Execution Plans

- Execution Plan
- SQL Monitor
- SQL Plan Monitor
- SQL Statistics
- Child Plan
- Plan Statistics
- Workareas
- AWR Hist SQL Statistics
- AWR Hist SQL Plan

Plan Control

- Hints found in 10053
- Stored Outlines
- SQL Profiles
- SQL Tuning Advisor

Schema Objects and CBO Stats

- Object Dependency
- Objects
- Fixed Objects
- Fixed Object Columns
- Tables
- Table Columns
- Indexes
- Indexed Columns
- Index Columns
- Metadata
- Column Histograms
- Column Histograms History
- Table Partitions
- Index Partitions
- Table Partition Columns
- Table Partition Histograms
- Table Subpartitions
- Index Subpartitions
- Table Subpartition Columns
- Table Subpartition Histograms

Weitere Sourcen

- V$SQL_PLAN
  - Zugriffspläne von Cursorn
- V$SQL_WORKAREA
  - memory Bereiche von Cursorn
- V$SQL_PLAN_STATISTICS
  - Statistiken für Zugriffsoperationen
- V$SQL_PLAN_STATISTICS_ALL
  - volle Daten mit statistics_level=all oder hint gather_plan_statistics
  - führt alle vrangehendenn Views zusammen

Weitere Sourcen

- V$SQL_MONITOR / V$SQL_PLAN_MONITOR
  - 11g Online Monitoring
- event 10132 level 1
- Dump des execution plan (bei hard parse)

```
set long 10000000
set longchunksize 10000000
set linesize 200

select dbms_sqltune.report_sql_monitor from dual;
```


SQL-Optimierung

- SQL-Profiles – (EE + Diagnostic pack) + tuning pack
  - (ab 10g) generiert von SQL Tuning Advisor und akzeptiert
  - Vergleichbar mit Stored Outlines
- SQL Plan Baselines / Plan Management
  - ab 11g EE
  - Nur Ausführung „akzeptierter" Ausführungspläne der „plan history"
  - geschaffen zur Stabilisierung von Zugriffsplänen
  - Expliziter Austausch von Zugriffsplänen über DBMS_SPM möglich

Stored Outlines

Outlines

- Verfügbar seit Version 8.1.6
- Unter 11g offiziell "deprecated"
- Übersteuern SQL Profiles und Baselines (!)
- "Fixierung" des Zugriffsplans – im Gegensatz zu SQL-Profiles
- Nutzt Session- nicht init-Systemparameter (!)
  - automatisierte Einschaltung per Trigger
- Hint-basiert – Pro's und Con's
  - Ungültige Hints erzeugen keine Fehlermeldung
  - "starre" Anweisung
- Entfernt Leerzeichen aus dem Code = Signature
- Abhängigkeiten zum Code – Install/migrate

Outlines

- Stored Outlines – "harte" Hints
  - dynamische Einschleusung von Hints in den SQL-Code
- Automatisierte oder manuelle Erstellung, Bearbeitung und „Umzug“
  - create private outline
  - dbms_outln_edit.create_edit_tables
  - update ol$hints – Erzeugen Public von Private Outline
- Kategorisierung möglich
  - Kategorien lassen sich aktivieren und deaktivieren
- Sinnvoll wenn
  - physische Strukturen existieren, jedoch nicht genutzt werden
  - Manuelle Hint-Plazierung unmöglich im Original
  - „Share“-Potential bei SQL

Outlines - Basis

- Benutzer OUTLN
  - per Default - EXPIRED & LOCKED
- Zusätzliche Packages unter SYS + public Synonyme
- System Privileg `create any outline`

```
OBJECT_TYPE      OBJECT_NAME
---------------- --------------------------
INDEX            OL$HNT_NUM
                 OL$NAME
                 OL$NODE_OL_NAME
                 OL$SIGNATURE
LOB              SYS_LOB0000000450C00021$$
PROCEDURE        ORA$GRANT_SYS_SELECT
TABLE            OL$
                 OL$HINTS
                 OL$NODES
```

```
SYS Pakete
------------
OUTLN_PKG
OUTLN_EDIT_PKG
```


Stored Outlines

```
-- init-Systemparameter einstellen dann pauschal, Name autom.
create_stored_outlines=<TRUE|<category> -init.ora (system/session)

-- alternativ gezielt manuell anlegen
CREATE OR REPLACE OUTLINE
outline_name FOR CATEGORY test
ON <SELECT-Statement>

-- Views
user_outlines, user_outline_hints

-- auch per PL/SQL - Name automatisch
SQL> BEGIN
2 dbms_outln.create_outline(
3 hash_value => '308120306',
4 child_number => 0,
5 category => 'test'
6 );
7 END;
-- nutzen muss immer explizit gesetzt werden, ggf. DB-Trigger
ALTER SESSION SET use_stored_outlines = test (keine init.ora)
```


Stored Outlines

```
ALTER OUTLINE SYS_OUTLINE_11022222434734504
RENAME TO outline_sprechend;

ALTER OUTLINE outline_sprechend CHANGE CATEGORY TO newcat;

begin
dbms_outln.update_by_cat(oldcat => 'ALT', newcat => 'NEU');
end;

-- Aufbau neuer Hints
ALTER OUTLINE outline_sprechend REBUILD;

ALTER OUTLINE outline_from_text DISABLE;

-- Umzug
exp tables=(outln.ol$,outln.ol$hints,outln.ol$nodes)
file=outln.dmp
```


Stored Outlines

```
NAME                 CATEGORY     USED   TIMESTAMP SIGNATURE
ENABLED  FORMAT
-------------------- ------------ ------ --------- -------------------------------- --
-----    ------
OL_DEMO              GUTEST       USED   22-FEB-11 AF5DBEE071A0172400FD5CD43F8E0D08
ENABLED  NORMAL


NAME                 SQL_TEXT
-------------------- ------------------------------------------------
OL_DEMO              select object_type ,
                     count(*)
                       from x
                     group  by object_type


NAME                   NODE      STAGE   JOIN_POS HINT
---------------- ---------- ---------- ---------- ------------------------------------
OL_DEMO                 1          1          0 USE_HASH_AGGREGATION(@"SEL$1")
OL_DEMO                 1          1          1 FULL(@"SEL$1" "X"@"SEL$1")
OL_DEMO                 1          1          0 OUTLINE_LEAF(@"SEL$1")
OL_DEMO                 1          1          0 ALL_ROWS
OL_DEMO                 1          1          0 DB_VERSION('11.2.0.1')
OL_DEMO                 1          1          0 OPTIMIZER_FEATURES_ENABLE('11.2.0.1')
OL_DEMO                 1          1          0 IGNORE_OPTIM_EMBEDDED_HINTS
```


Outlines anpassen

- sinnvoll wenn CBO nicht zu optimalem Plan bewegt werden kann
- Prozedere
  - zunächst manuelle Tests über explizite Hints zur Verifizierung der Strategie
  - ggf. Anlegen der Edit-Tables (`dbms_outln_edit.create_edit_tables`)
    - ab 10g bereits als Temp Tab in SYSTEM mit PubSyn
  - Anlegen und anpassen von `private outlines` (Daten in temporary tables, keine Data Dictionary)
  - create public from private outline

Outlines anpassen


```
-- Session einstellen: Opti/Init Parameter, Stats etc.

CREATE PRIVATE OUTLINE <name> FROM <SQL/public outln>;

-- in gleicher Session anzeigen
SELECT ... ol$/ol$hints - per Syn auf System Temp Table

UPDATE ol$hints ....

-- Laden der Updates
execute dbms_outln_edit.refresh_private_outline ...

--Test ...
ALTER SESSION SET use_stored_outlines = FALSE;
ALTER SESSION SET use_private_outlines = TRUE;

-- "publizieren"
CREATE OR REPLACE OUTLINE <pubname> FROM PRIVATE <priv_name>;
```


Beachten

- Auswahl über "signature" – identische Signaturen ziehen spezifische Outlines
- Zusätzliche Abhängigkeiten für Installationen und Migrationen und Code-Anpassungen
- Kontrolle der Nutzung
  - `v$sql` – outline_category, outline_sid
  - used Attribut in `user_outlines` View
    - ggf. vorher `dbms_outln.clear_used`
- Testen zur Vermeidung ungültiger/wirkungsloser Hints
- Bei `cusor_sharing` ggf. umgewandelten SQL-String nutzen

SQL Profiles

SQL-Profiles

- SQL-Profiles – (EE + Diagnostic pack) + tuning pack
  - (ab 10g) generiert von SQL Tuning Advisor
  - muss "akzeptiert" werden
  - Beeinflussen Optimizer durch "statistische Korrekturen" (OPT_ESTIMATE Hint "faktorisiert")
    - kein "locking" des Ausführungsplanes
  - ignoriert falsche Hint-Angaben
  - wie Outlines auch kategorisierbar
  - kann auch modifiziert/kreiert werden.
  - Gefährlich, wenn Algorithmen und/oder Stats sich verändern.

Prozedere

- Statement identifizieren und isolieren
  - SQL-Text, shared pool (sql_id)
  - AWR (sql_id), SQL Tuning Set
- Start eines SQL Tuning Task im Tuning Advisor
- Automatic Tuning Optimizer analysiert Statement, Statistics, init.ora, What-if Analysen
- SQL Profile wird gerechnet und an Tuning Advisor übergeben.
- Der Benutzer akzeptiert das Profile.
- Das profile wird im DD gespeichert und kann dann genutzt werden.

Prozedere

```
dbms_sqltune.create_tuning_task (function)

dbms_sqltune.execute_tuning_task

SELECT dbms_sqltune.report_tuning_task('taskname')
FROM dual;

dbms_sqltune.accept_sql_profile

dbms_sqltune.drop_tuning_task

ALTER SESSION SET sqltune_category = GU; (true/false)
```

- Kann ein- und ausgeschaltet werden
- kategorisierbar
- "Normalisierung" für Signatur (spaces, Groß-/Kleinschreibung)
  - zusätzlich force_match
- Systemprivileg `advisor` – Profiles `administer sql management object` (11g)

SQL Tuning Advisor

- Bericht enthält
  - GENERAL INFORMATION SECTION
  - FINDINGS SECTION
  - EXPLAIN PLANS SECTION
    - 1- Original With Adjusted Cost
    - 2- Using SQL Profile
- Profile muss akzeptiert werden

SQL-Profiles

```
dbms_sqltune.accept_sql_profile
dbms_sqltune.alter_sql_profile (enable/disable)
ALTER SESSION SET sqltune_category = test; -- Kategorie schalten

-- force_matching = false - keine Ersetzung von Literalen
SELECT '&sql' sql_text, dbms_sqltune.sqltext_to_signature('&sql',0)
signature FROM dual;

SQL_TEXT                                          SIGNATURE
----------------------------------------- ----------------------
select * from test                          7199376810451276368
SELECT * from tEst                          7199376810451276368
select * from test WHERE c1=3               6536985043163659612
select   *   FROM test WHERE c1=4           4917319147810962328

-- force_matching = TRUE

SQL_TEXT                                          SIGNATURE
----------------------------------------- ----------------------
select * from test                          7199376810451276368
SELECT * from tEst                          7199376810451276368
select * from test WHERE c1=3              11989761555880899295
select   *   FROM test WHERE c1=4          11989761555880899295
```


SQL-Profiles

- View DBA_SQL_PROFILES
  - zeigt allgemeine Attribute
  - keine Hints
- Hints über `sqlprof$attr` (10g) , xmltype über clob in 11g

```
SQL> SELECT attr_val
2 FROM sys.sqlprof$ p, sys.sqlprof$attr a
3 WHERE p.sp_name = 'first_rows'
4 AND p.signature = a.signature
5 AND p.category = a.category;

ATTR_VAL
----------------------------------------
FIRST_ROWS(6)

SELECT extractValue(value(h),'.') AS hint
FROM sys.sqlobj$data od, sys.sqlobj$ so,
table(xmlsequence(extract(xmltype(od.comp_data),'/outline_data/hint'))) h
WHERE so.name = 'all_rows.sql'
AND so.signature = od.signature AND so.category = od.category
AND so.obj_type = od.obj_type AND so.plan_id = od.plan_id;
```


Umzug

```
-- Aufbau Staging Bereich
dbms_sqltune.create_stgtab_sqlprof

-- Übernahme eines Profils
dbms_sqltune.pack_stgtab_sqlprof

-- Übertragen per exp/imp oder expdp/impdp

-- ggf. umbenennen /neue Kategoorie
dbms_sqltune.remap_stgtab_sqlprof

-- Einspielen ins DD
dbms_sqltune.unpack_stgtab_sqlprof
```

- Austausch auch zwischen unterschiedlichen Datenbanken

SQL-Profiles

- Profiles können auch explizit erstellt werden
  - offiziell nicht supported

```
dbms_sqltune.import_sql_profile(
        name => 'import_sql_profile',
        description => 'Test-Profile',
        category => 'TEST',
        sql_text => 'SELECT * FROM t ORDER BY id',
        profile => sqlprof_attr('first_rows(42)',
        'optimizer_features_enable(default)'),
        replace => FALSE,
        force_match => FALSE
);
-- weitere Beispiele
COLUMN_STATS("KSO"."SKEW", "PK_COL", scale, length=5)
COLUMN_STATS("KSO"."SKEW", "COL1", scale, length=4
distinct=828841 nulls=12.8723033 min=1 max=1000000)
TABLE_STATS("KSO"."SKEW", scale, blocks=162294 rows=35183107.66)
OPTIMIZER_FEATURES_ENABLE(default)
```


SQL Baselines
SQL Plan Management

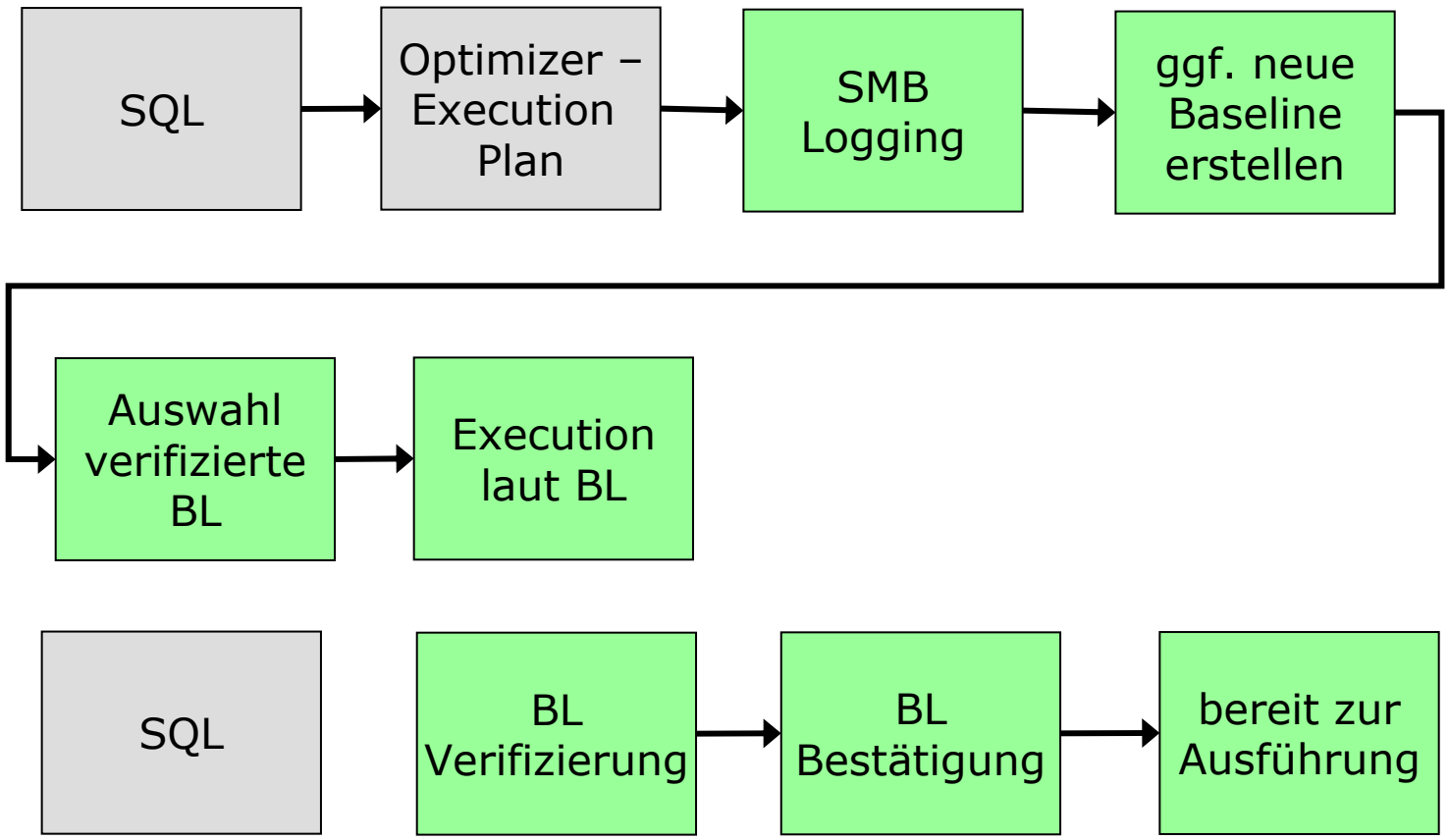
Baselines – Plan Management

- Ziel – stabiles und kontrollierbares Antwortverhalten des Optimizer
  - Einfluss über Hints
- Eingeführt in Version 11g
  - genutzt aber nicht erstellt per default (`optimizer_use_sql_plan_baselines`)
  - Erstellt per `optimizer_capture_sql_plan_baselines` oder explizites Laden
- Mehrere BLs per Statement sind möglich
- Kontrollierte Bereitstellung der BLs im Rahmen des SQL Plan Management
- Arbeitet mit SQL Profiles zusammen (merge)
- Wird überschrieben durch Outlines!

SQL Plan Management

- SQL Plan Baselines
  - für „wiederholt“ ausgeführte Statements – keine ad-hoc St.
  - gespeichert in SMB (SQL management Base - SYSAUX) in Form von „plan baselines“
    - SQL text, outline, bind variables, und compilation environment
  - automatisch über `optimizer_capture_sql_plan_baselines` (default FALSE) – View DBA_SQL_PLAN_BASELINES
  - oder explizit geladen (SQL Sets/AWR, Cursor Cache, Staging Table)
  - nutzbar über `optimizer_use_sql_plan_baselines`
  - Schnittstellen EM und DBMS_SPM API

Plan Management



SQL Plan Management

- Verifizieren und Akzeptieren von neuen Ausführungsplänen – Plan Evolution
- Varianten:
  - Aufruf von DBMS_SPM
  - Scheduler Job erstellen
  - Starten von SQL Tuning Advisor
  - Aktivieren von automatischer SQL-Tuning Task
- Baseline Attribute
  - enabled/disabled
  - Accepted/not accepted
  - fixed/not fixed – kein automatisches Hinzufügen

SQL Plan Management

- Neue Pläne werden als „non-accepted“ der Baseline hinzugefügt und als accepted eingestuft, wenn sie erfolgreich verifiziert werden können
- Konfiguration: %-Anteil von SYSAUX, Purge-Periode nicht genutzter Pläne (plan retention)

```
SET SERVEROUTPUT ON
SET LONG 10000
DECLARE
 report clob;
BEGIN
 report := DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(
           sql_handle => 'SYS_SQL_593bc74fca8e6738');
 DBMS_OUTPUT.PUT_LINE(report);
END;
```


Baselines anlegen , testen

```
-- SQL Plus variable definieren
-- capture einschalten:
alter session
   set optimizer_capture_sql_plan_baselines = true;
-- SQL Statement MEHRFACH ausführen
-- ausschalten optimizer_capture_sql_plan_baselines
-- Baseline anzeigen
SELECT sql_text, plan_name, enabled, accepted
FROM dba_sql_plan_baselines;
-- Neues Hard Parse provozieren
alter system flush shared_pool;
-- Bindevariable verändern SPM = neuer Plan:
select * from table(dbms_xplan.display_cursor
  ('2pk8fr49y4q7h', 2, 'basic note'));
-- Use einschalten:
alter session set optimizer_use_sql_plan_baselines = true;
SQL plan baseline SYS_SQL_PLAN_fcc170b0f1563ebc used for
this statement
```


Baselines übernehmen - manuell

```
SQL> var report clob;
SQL> exec :report := dbms_spm.evolve_sql_plan_baseline();
SQL> print :report
--------------------------------------------------------------------------------
                     Evolve SQL Plan Baseline Report
--------------------------------------------------------------------------------
Inputs:
-------
  SQL_HANDLE =
  PLAN_NAME  =
  TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
  VERIFY     = YES
  COMMIT     = YES
Plan: SYS_SQL_PLAN_fcc170b08cbcb825
-----------------------------------
  Plan was verified: Time used .1 seconds.
  Passed performance criterion: Compound improvement ratio >= 10.13
  Plan was changed to an accepted plan.
Baseline Plan      Test Plan     Improv. Ratio
-------------      ---------     -------------
  Execution Status:        COMPLETE       COMPLETE
  Rows Processed:               960            960
  Elapsed Time(ms):              19             15              1.27
  CPU Time(ms):                  18             15               1.2
  Buffer Gets:                 1188            116             10.24
  Disk Reads:                     0              0
  Direct Writes:                  0              0
  Fetches:                        0              0
  Executions:                     1              1
```


Baselines übernehmen - STA

```
-- SQL Tuning Advisor

SQL> var tname varchar2(30);
SQL> exec :tname := dbms_sqltune.create_tuning_task
        (sql_id => 'bfbr3zrg9d5cc');

SQL> exec dbms_sqltune.execute_tuning_task
        (task_name => :tname);

SQL> select dbms_sqltune.report_tuning_task
        (:tname, 'TEXT', 'BASIC') FROM dual;
SQL> exec dbms_sqltune.accept_sql_profile
        (task_name => :tname);
```

- SQL Tuning Advisor
  - akzeptiert Baseline
  - baut zusätzlich SQL-Profile
  - Kontrolle: dba_sql_profiles, dba_sql_plan_baselines

Plan History - Diverses

```
SQL> var pls number
SQL> exec :pls := dbms_spm.load_plans_from_cursor_cache
      ( sql_id => 'b17xnz4y8bqv1,
      , plan_hash_value => 2290236051
      , sql_handle => 'SYS_SQL_4bf02d85fcc171b0');

-- Alle Pläne einer Baseline

SQL> select * from
      table(dbms_xplan.display_sql_plan_baseline
           ( sql_handle => 'SYS_SQL_4bf02d85fcc171b0'
            , format => 'basic'));
```


- Baseline + SQL Profile
  - Profile hilft bei der Kostenbeurteilung akzeptierter Pläne
- Baseline + Stored Outline
  - wenn StO eingeschaltet ist wird Baseline ignoriert.
  - Desupport in Future – Migration per DBMS_SPM.LOAD_PLANS_FROM_SQLSET

SQL Plan Management

```
SELECT parameter_name, parameter_value
FROM DBA_SQL_MANAGEMENT_CONFIG
/
-- Konfigurieren init.ora inkl. „underscores"
BEGIN
DBMS_SPM.CONFIGURE(
'space_budget_percent',20);
-- default 10% von SYSAUX, gültige Werte 1 - 50, ALERT.LOG
-- plan_retention_weeks -> default 53 -> Werte 5 - 523
-- gilt für nicht genutzte Pläne!
END;
/
SELECT sql_handle, plan_name, enabled, accepted, fixed
from DBA_SQL_PLAN_BASELINES;
select * from table(
dbms_xplan.display_sql_plan_baseline(
sql_handle=>'SYS_SQL_209d10fabbedc741',
format=>'basic'));
```


SQL Plan Management

```
--------------------------------------------------------------------------------
SQL handle: SYS_SQL_38813422915fb3f1
SQL text: select /*+ gather_plan_statistics */ CUST_FIRST_NAME, CUST_LAST_NAME,
          CUST_GENDER from customers where CUST_STATE_PROVINCE = :sta
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Plan name: SYS_SQL_PLAN_915fb3f19f17d9e6
Enabled: YES     Fixed: NO      Accepted: YES     Origin: AUTO-CAPTURE
--------------------------------------------------------------------------------
Plan hash value: 1459632612
-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name           | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                |   383 | 10724 |   257   (0)| 00:00:04 |
|   1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS      |   383 | 10724 |   257   (0)| 00:00:04 |
|*  2 |   INDEX RANGE SCAN           | CUSTOMER_STATE |   383 |       |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("CUST_STATE_PROVINCE"=:STA)
```


SQL Plan Management

```
Plan: SYS_SQL_PLAN_b5429522ee05ab0e
-----------------------------------
  Plan was verified: Time used 3.9 seconds.
  Failed performance criterion: Compound improvement ratio <= 1.4.

                     Baseline Plan      Test Plan     Improv. Ratio
                     -------------      ---------     -------------
  Execution Status:        COMPLETE       COMPLETE
  Rows Processed:                 1              1
  Elapsed Time(ms):            3396            440            7.72
  CPU Time(ms):                1990            408            4.88
  Buffer Gets:                 7048           5140            1.37
  Disk Reads:                  4732             53           89.28
  Direct Writes:                  0              0
  Fetches:                     4732             25          189.28
  Executions:                     1              1
```


SQL Plan Management

- Migrations-Szenario:
- `optimizer_features_enabled` auf 10g
- „capture SQL Plans“
- `optimizer_features_enabled` auf 11g
- Statements „entwickeln“

Mögliches Migrations-Szenario

- Kritische SQL-Statements sammeln
- hiervon SQL Tuning Set erstellen
  - SQL text, bind variables, execution plans, execution statistics
  - DBMS_SQLTUNE.CAPTURE_CURSOR_CACHE_SQLSET Filter für SQL-String und/oder Schema möglich
  - Set in staging table „verpacken“ DBMS_SQLTUNE.PACK_STGTAB_SQLSET
  - DPexport des Staging Bereichs
- Tuning Set auf Ziel einführen
  - Import des Staging Bereichs
  - „Auspacken“ des Tuning Sets DBMS_SQLTUNE.UNPACK_STGTAB_SQLSET
  - Laden des Tuning Set in Plan Management DBMS_SPM.LOAD_PLANS_FROM_SQLSET

Mögliches Migrations-Szenario

- Tuning Set auf Ziel einführen
  - SQL-Statements per explain verifizieren
- Alternativ (11g zu 11g)
  - entsprechende Baselines auf Ziel löschen
  - Staging Table für Baselines erzeugen
    DBMS_SPM.CREATE_STGTAB_BASELINE
  - Baselines „verpacken“ übertragen und entpacken
    DBMS_SPM.PACK_STGTAB_BASELINE
- Alternativ (10g in 11g)
  - SPA Task erzeugen für entsprechenden STS
    DBMS_SQLPA.CREATE_ANALYSIS_TASK
  - starten mit unterschiedlichen
    OPTIMIZER_FEATURES_ENABLE
  - danach Analyse
    DBMS_SQLPA.EXECUTE_ANALYSIS_TASK

Danke für's Zuhören

www.database-consult.de