Oracle 11g – Ten less popular Features

Martin Frauendorfer, SAP Active Global Support

martin.frauendorfer@sap.com

November 2011

Agenda

Introduction

1. SQL Monitoring
2. Extended Statistics
3. SQL Repair Advisor
4. I/O Calibration
5. Automatic Shared Pool Extensions
6. DDL Lock Timeout
7. Unrecoverable SCN Tracking
8. Nested Loop Join Implementation
9. Full Table Scan Implementation
10. Maintenance Windows and Resource Manager

Agenda

Introduction

1. SQL Monitoring
2. Extended Statistics
3. SQL Repair Advisor
4. I/O Calibration
5. Automatic Shared Pool Extensions
6. DDL Lock Timeout
7. Unrecoverable SCN Tracking
8. Nested Loop Join Implementation
9. Full Table Scan Implementation
10. Maintenance Windows and Resource Manager

Introduction

Motivation

- **There are several popular new Oracle 11g features that can be helpful for database administration like:**
  - Advanced Compression
  - Dictionary Only ADD COLUMN
  - Invisible Indexes
  - Pending Statistics
  - Tablespace Encryption
- **But there are also some less popular new features and some minor changes that are good to know in SAP environments.**
- **This presentation wants to focus on these areas.**
- **Occurrences of** *SQL: „<script_name>.txt“* **reference scripts available in SAP Note 1438410**

Agenda

Introduction

1. **SQL Monitoring**
2. Extended Statistics
3. SQL Repair Advisor
4. I/O Calibration
5. Automatic Shared Pool Extensions
6. DDL Lock Timeout
7. Unrecoverable SCN Tracking
8. Nested Loop Join Implementation
9. Full Table Scan Implementation
10. Maintenance Windows and Resource Manager

SQL Monitoring (1)

Overview

- **Difficult questions while analyzing complex SQL statements:**
  - What are the bind variable contents of a long running execution (V$SQL_BIND_CAPTURE doesn't necessarily capture the long running executions)?
  - How many rows are processed, how much data is read from disk and how much PGA / PSAPTEMP is used **in each step of the execution plan**?
  - Which timed events happen for each step of the execution plan?
  - Was the statement successful or did it fail (e.g. due to user requested cancel)?
  - How well did the PX slaves collaborate in case of parallel execution?
- **SQL Monitoring is now able to provide answers**
- **Focus on current and recent SQL statement executions**
- **Part of the Oracle Tuning Pack**

SQL Monitoring (2)

Views

- **GV$SQL_MONITOR**
  - Contains all current or recent SQL statement executions with a runtime of at least 5 seconds (_SQLMON_THRESHOLD)
  - Is refreshed every second
  - Minimum retention time after execution end is 60 seconds (_SQLMON_RECYCLE_TIME)
  - Each SQL statement execution can be uniquely identified via INST_ID, SID (session ID), SQL_ID and SQL_EXEC_ID
  - SQL_EXEC_ID is a counter that allows to distinguish between different executions of same SQL_ID by same SID on the same INST_ID
  - Contains information like execution status (EXECUTING, DONE, …), start and end time, parallelism degree, elapsed time, CPU time, buffer gets, disk reads and error message
  - Complete set of bind variable values (column BINDS_XML)

SQL Monitoring (3)

Views

- **GV$SQL_PLAN_MONITOR**
  - Classic execution plan information similar to GV$SQL_PLAN
  - Number of executions of each plan line (STARTS), can be significantly larger than 1 in case of nested loops with a high amount of accesses to the inner table
  - Number of rows returned by each plan line
  - Read and write activity of each plan line
  - Workareas in PGA and PSAPTEMP used by each plan line

- **GV$ACTIVE_SESSION_HISTORY**
  - Standard ASH table that can be joined to the SQL Monitoring views for a better understanding of the runtime behavior
  - Contains (as of Oracle 11g) the execution plan line ID (SQL_PLAN_LINE_ID) so that the samples can be mapped to steps of the execution plan
  - *SQL: "ASH_Filter_11g+.txt"* and *SQL: "ASH_Aggregation_11g+.txt"* can be used to filter and aggregate related information

SQL Monitoring (4)

Examples

- **SQL Monitoring overview:**
  - *SQL: "SQL_SQLMonitoring_Overview_11g+.txt"*
  - **STATUS**: Status of the SQL statement (EXECUTING, DONE (ERROR), DONE (FIRST N ROWS), DONE (ALL ROWS) or DONE).
  - **PX_REQ**: Requested parallelism degree
  - **PX_ALLOC**: Used parallelism degree (lower than PX_REQ in case of downgrades)
  - **ERR_MSG**: Oracle error message that terminated SQL statement (e.g. ORA-01013, ORA-01652).

INST	SID	SQL_ID	STATUS	SQL_EXEC_START	LAST_REFRESH_TIME	ELAPSED_S	CPU_S	BUFFER_GETS	DISK_READS	PX_REQ	PX_ALLOC	ERR_MSG
1	95	bkcn950409qzz	DONE (ALL ROWS)	06.09.2010 12:43:14	06.09.2010 12:43:28	15	1	67624	1118	0	0	
1	537	7zvvzgz304zs9	DONE (ALL ROWS)	06.09.2010 12:43:12	06.09.2010 12:43:28	15	1	67300	1193	0	0	
1	95	fv4kuztn1jn01	DONE (ERROR)	06.09.2010 12:29:45	06.09.2010 12:30:17	31	30	1163	10	0	0	ORA-01013
1	275	8s7tq9z1uqyfz	DONE (ALL ROWS)	06.09.2010 10:41:51	06.09.2010 10:41:56	4	0	25	1338	0	0	
1	6	2s23asxwfrpt9	DONE (ERROR)	06.09.2010 09:00:46	06.09.2010 09:01:19	33	1	4677	4676	0	0	ORA-01013
1	364	8s7tq9z1uqyfz	DONE (ALL ROWS)	06.09.2010 06:31:47	06.09.2010 06:31:51	3	0	25	1338	0	0	
1	277	8s7tq9z1uqyfz	DONE (ALL ROWS)	06.09.2010 03:41:48	06.09.2010 03:41:53	5	0	25	1338	0	0	

SQL Monitoring (5)

Examples

- **Execution Plan Details:**
  - *SQL: "SQL_SQLMonitoring_ExecutionPlan_11g+.txt"*
  - **STARTS:** Number of time the execution plan line was executed
  - **RECORDS:** Number of output records produced
  - **IO_READ_MB:** Amount of disk reads (in MB)
  - **WA_TEMP_MB:** Maximum work area size in the temporary tablespace (in MB)

INST	SID	SQL_ID	SQL_EXEC_ID	PLAN_ID	ACTION_INFO	STARTS	RECORDS	IO_READ_MB	WA_TEMP_MB
1	2101	1h3znud0q2ut6	16777344	0	SELECT STATEMENT	1	16506	0	
				1	FILTER	1	16506	0	
				2	NESTED LOOPS	1	16506	0	
				3	NESTED LOOPS	1	60617	0	
				4	NESTED LOOPS	1	60617	0	
				5	NESTED LOOPS	1	80230	0	
				6	INDEX UNIQUE SCAN (TVKWZ~0)	1	1	0	
				7	INDEX RANGE SCAN (MARC~0)	1	80230	246	
				8	TABLE ACCESS BY INDEX ROWID (MVKE)	80230	60617	258	
				9	INDEX UNIQUE SCAN (MVKE~0)	80230	80230	231	
				10	INDEX UNIQUE SCAN (MARA~0)	71278	60617	160	
				11	TABLE ACCESS BY INDEX ROWID (MARA)	98187	16506	81	

SQL Monitoring (6)

Examples

- **Bind variable contents:**
  - *SQL: "SQL_SQLMonitoring_BindVariableContent_11g+.txt"*

```
-------------------------------------------------------------------------------------------------------
|INST|SID   |SQL_ID       |SQL_EXEC_START     |BIND_NAME  |BIND_VALUE                    |BIND_TYPE    |
-------------------------------------------------------------------------------------------------------
|   1|   271|davq584d5uv0m|28.12.2010 01:57:54|:A0        |#%_C%                         |CHAR(32)     |
|    |      |             |                   |:A1        |T                             |CHAR(32)     |
|   1|   627|fp4n5d5p7zcjv|28.12.2010 01:03:06|:B2        |116796                        |NUMBER       |
|    |      |             |                   |:B1        |EXEC_19423                    |VARCHAR2(32) |
|    |      |             |                   |:B1        |EXEC_19423                    |VARCHAR2(32) |
|   1|   359|14w126fhfbzf8|28.12.2010 01:01:24|:A0        |R3TR                          |CHAR(32)     |
|    |      |             |                   |:A1        |PROG                          |CHAR(32)     |
|    |      |             |                   |:A2        |SVER_PKRT_P3                  |CHAR(128)    |
|    |      |             |                   |:A3        |SVER_PKRT_P2                  |CHAR(128)    |
|    |      |             |                   |:A4        |SVER_PKRT_P1                  |CHAR(128)    |
|    |      |             |                   |:A5        |SVER_PKRT_ENH_ROOT            |CHAR(128)    |
|    |      |             |                   |:A6        |SVER_PKRT_ENH                 |CHAR(128)    |
|    |      |             |                   |:A7        |SVER_PKRT_ABAP_OBJECTS_P2     |CHAR(128)    |
|    |      |             |                   |:A8        |SVER_PKRT_ABAP_OBJECTS_P1     |CHAR(128)    |
|    |      |             |                   |:A9        |SVER_PKRT_ABAP_OBJECTS        |CHAR(128)    |
|    |      |             |                   |:A10       |SVER_PKRT                     |CHAR(128)    |
|    |      |             |                   |:A11       |SVER_PAKG                     |CHAR(128)    |
|    |      |             |                   |:A12       |SVER_PAKE                     |CHAR(128)    |
|    |      |             |                   |:A13       |SVER_PAKD                     |CHAR(128)    |
|    |      |             |                   |:A14       |SVER_PAKC_X                   |CHAR(128)    |
-------------------------------------------------------------------------------------------------------
```

- Real-life case study in file `CaseStudy_SQL_Monitoring.txt`

Agenda

Introduction

1. SQL Monitoring
2. **Extended Statistics**
3. SQL Repair Advisor
4. I/O Calibration
5. Automatic Shared Pool Extensions
6. DDL Lock Timeout
7. Unrecoverable SCN Tracking
8. Nested Loop Join Implementation
9. Full Table Scan Implementation
10. Maintenance Windows and Resource Manager

Extended Statistics (1)

Motivation

- The Cost Based Optimizer (CBO) assumes that values of different columns are not correlated
- If column A has 100 distinct values and column B has 100 distinct values the CBO assumes that there are 100 * 100 = 10.000 existing combinations.
- In reality the amount of combinations can vary between 100 and 10.000.
- As a consequence the CBO may assume an amount of „Estimated rows“ that is significantly lower than in reality
- This wrong assumption can significantly impact join orders and performance.
- Extended Statistics (or Multi Column Statistics) can make the CBO aware about correlation.
- The creation of Extended Statistics is often a result of SQL statement analysis and no proactive task.

Extended Statistics (2)

Details

- Extended Statistics on a set of columns can be defined (and created) in the following way:
  - Definition:

```
SELECT DBMS_STATS.CREATE_EXTENDED_STATS ('<owner>',
'"<table_name>"', ' ("<col1>", ..., "<colN>")') FROM DUAL;
```

  - Definition and creation:

```
EXEC DBMS_STATS.GATHER_TABLE_STATS('<owner>', '"<table_name>"',
METHOD_OPT => 'FOR COLUMNS ("<col1>", ..., "<colN>") SIZE 1');
```

- Afterwards they are automatically taken into account during BRCONNECT statistic runs
- Existing Extended Statistics can be found in DBA_STAT_EXTENSION (*SQL: „CBOStatistics_ExtendedStatistics_11g+.txt“*)
- Currently Extended Statistics are ignored if range conditions are used, so that columns with range predicates have an unjustified advantage
- Hopefully this „feature“ is fixed soon

Extended Statistics (3)

Example

- Table AUSP often causes trouble in Oracle environments because it has several rather similar indexes and the CBO sometimes decides to pick the wrong one.
- Main problem: Optimal costs are calculated for several indexes
- In reality some indexes are not optimal because columns are correlated.
- Often columns KLART and ATINN are correlated significantly.
- In order to make the CBO aware about that, a column group is defined:

```
SELECT DBMS_STATS.CREATE_EXTENDED_STATS(:OWNER, 'AUSP', '(KLART, ATINN)') FROM DUAL;
```

- The next CBO statistic creation will automatically create Extended Statistics for this column group and a proper CBO decision will be more likely.
- This command is part of the SAP delivered CBO statistics (SAP Note 1020260).
- Details in file `CaseStudy_ExtendedStatistics.txt`

Agenda

Introduction

1. SQL Monitoring
2. Extended Statistics
3. **SQL Repair Advisor**
4. I/O Calibration
5. Automatic Shared Pool Extensions
6. DDL Lock Timeout
7. Unrecoverable SCN Tracking
8. Nested Loop Join Implementation
9. Full Table Scan Implementation
10. Maintenance Windows and Resource Manager

SQL Repair Advisor (1)

Overview

- SQL Repair Advisor can be used to repair SQL statements with a problem like:
  - Wrong result set
  - Bad performance
  - Termination with an error
- Implemented based on procedures of the **DBMS_SQLDIAG** package:
  - CREATE_DIAGNOSIS_TASK
  - EXECUTE_DIAGNOSIS_TASK
  - REPORT_DIAGNOSIS_TASK
  - ACCEPT_SQL_PATCH
  - DROP_SQL_PATCH
- **DBA_SQL_PATCHES** contains proposed and implemented patches

SQL Repair Advisor (2)

Details

- A database that is able to patch itself – how does it work?
- Is the Oracle software changed and recompiled to come around the problem?
- No, that‘s not possible of course
- Instead the patch is nothing else than a special SQL plan baseline
- A SQL plan baseline is a fixed execution plan similar to a stored outline
- All problems like performance, wrong results or terminations are typically caused by specific execution plans.
- If the SQL Repair Advisor is able to determine another execution plan without that problem, it can be used as a SQL plan baseline for the SQL statement.

SQL Repair Advisor (3)

Example – Wrong result set

- SQL statement was identified that returns only 1 record although it should return 135 records
- Creation of diagnosis task:

```
 T_ID := DBMS_SQLDIAG.CREATE_DIAGNOSIS_TASK
( SQL_ID     => '<sql_id>',
  TASK_NAME    => 'STAR_MJC',
  PROBLEM_TYPE => DBMS_SQLDIAG.PROBLEM_TYPE_WRONG_RESULTS,
  TIME_LIMIT => 100000
);
```

- Execution of diagnosis task:

```
 DBMS_SQLDIAG.EXECUTE_DIAGNOSIS_TASK (T_ID);
```

- Report for diagnosis task:

```
REP_OUT := DBMS_SQLDIAG.REPORT_DIAGNOSIS_TASK (T_ID, DBMS_SQLDIAG.TYPE_TEXT);
DBMS_OUTPUT.PUT_LINE (REP_OUT);
```

SQL Repair Advisor (4)

Example – Wrong result set (2)

- Report content:

```
      GENERAL INFORMATION SECTION
-------------------------------------------------------------------------------
Tuning Task Name   : STAR_MJC
Tuning Task Owner  : SAPBAH
Workload Type      : Single SQL Statement
Scope              : COMPREHENSIVE
Time Limit(seconds): 100000
Completion Status  : COMPLETED
Started at         : 12/16/2010 09:00:08
Completed at       : 12/16/2010 10:51:02

-------------------------------------------------------------------------------
Schema Name: SAPBAH
SQL ID     : 8cdjyrjk0857g
SQL_TEXT   : ...

-------------------------------------------------------------------------------
FINDINGS SECTION (1 finding)
-------------------------------------------------------------------------------

1- SQL Patch Finding (see explain plans section below)
--------------------------------------------------------
  A potentially better execution plan was found for this statement.

  Recommendation

--------------
  - Consider accepting the recommended SQL patch.

    execute dbms_sqldiag.accept_sql_patch(task_name => 'STAR_MJC', task_owner =>
    'SAPBAH', replace => TRUE);

  Rationale
  ---------
    Recommended plan with hash value 352655436 has number of rows 135, check
    sum 289811451458135, execution time 249 and 34718 buffer gets
```

SQL Repair Advisor (5)

Example – Wrong result set (3)

- Accepting the patch:

```
EXEC DBMS_SQLDIAG.ACCEPT_SQL_PATCH -
( TASK_NAME  => 'STAR_MJC', -
  TASK_OWNER => 'SAPBAH', -
  REPLACE    => TRUE );
```

- With this patch 135 rows are correctly returned.
- How was the correct result set achieved?

```
SELECT
  A.NAME,
  B.COMP_DATA
FROM
  SQLOBJ$ A,
  SQLOBJ$DATA B
WHERE
  A.SIGNATURE = B.SIGNATURE AND
  A.PLAN_ID = B.PLAN_ID AND
  A.OBJ_TYPE = B.OBJ_TYPE AND
  A.CATEGORY = B.CATEGORY AND
  A.NAME LIKE '%SQLPTCH%';

NAME                           COMP_DATA
------------------------------ --------------------------------------------------------------------------------
SYS_SQLPTCH_012ceeb58e1e0000   <outline_data><hint><![CDATA[NO_STAR_TRANSFORMATION(@"SEL$0F5ACDD3")]]></hint>
                               <hint><![CDATA[NO_STAR_TRANSFORMATION(@"SEL$0F5ACDD3")]]></hint></outline_data>
```

SQL Repair Advisor (6)

Example – Wrong result set (4)

- Improvement only valid for this particular SQL_ID (and very similar SQL statements that e.g. only differ in blanks / line breaks)
- Details in document `CaseStudy_SQLRepairAdvisor.txt`

Agenda

Introduction

1. SQL Monitoring
2. Extended Statistics
3. SQL Repair Advisor
4. **I/O Calibration**
5. Automatic Shared Pool Extensions
6. DDL Lock Timeout
7. Unrecoverable SCN Tracking
8. Nested Loop Join Implementation
9. Full Table Scan Implementation
10. Maintenance Windows and Resource Manager

I/O Calibration (1)

Overview

- I/O Calibration can be used to understand the maximum I/O read throughput of a single process and of the whole I/O sub system.
- This information is useful for the following reasons:
  - An efficient parallel execution degree (e.g. during reorganizations) typically depends more on the I/O capabilities than on CPUs.
  - You can check how much additional I/O can be handled by the system on top of the normal SAP workload.
- Calibration can be done with the procedure **DBMS_RESOURCE_MANAGER.CALIBRATE_IO**.
- Oracle features like the Resource Manager or the new parallel execution approach (PARALLEL_DEGREE_POLICY = AUTO) take advantage of I/O calibration.

I/O Calibration (2)

Details

- The following information is generated:
  - max_iops: Maximum number of I/O requests per second that can be sustained. The I/O requests are randomly-distributed I/O reads of 8K (database block size)
  - max_mbps: Maximum throughput of I/O that can be sustained (in MB / s). The I/O requests are randomly distributed.
  - max_pmbps: Maximum throughput of I/O that can be sustained **by a single process** (in MB / s)
- The ratio of max_pmbps and max_mbps can be a first guidance for good parallel execution degrees of I/O intensive operations like reorganizations.
- Example: `See Example_Calibration.txt`

Agenda

Introduction

1. SQL Monitoring
2. Extended Statistics
3. SQL Repair Advisor
4. I/O Calibration
5. **Automatic Shared Pool Extensions**
6. DDL Lock Timeout
7. Unrecoverable SCN Tracking
8. Nested Loop Join Implementation
9. Full Table Scan Implementation
10. Maintenance Windows and Resource Manager

Automatic Shared Pool Extensions

Overview

- Even without ASMM (SGA_TARGET) and AMM (MEMORY_TARGET) Oracle may resize SGA components, e.g. in order to avoid ORA-04031 due to shared pool overflows.
- In the case study described in `CaseStudy_SharedPoolExtensions.txt` the size of the buffer pool was reduced from 1 GB to 100 MB while the shared pool increased from 500 MB to 1.4 GB.
- This is a feature, not a bug!
- It can be turned off by setting the following parameter:

  _MEMORY_IMM_MODE_WITHOUT_AUTOSGA = FALSE

- As long as the resizes don't cause trouble there is no need to deactivate this feature.

Agenda

Introduction

1. SQL Monitoring
2. Extended Statistics
3. SQL Repair Advisor
4. I/O Calibration
5. Automatic Shared Pool Extensions
6. **DDL Lock Timeout**
7. Unrecoverable SCN Tracking
8. Nested Loop Join Implementation
9. Full Table Scan Implementation
10. Maintenance Windows and Resource Manager

DDL Lock Timeout

Overview

- DDL statements like CREATE or ALTER typically require an exclusive table lock.
- As long as uncommitted changes on the underlying table exist, this lock can‘t be allocated and **ORA-00054** is issued.
- Some DDL commands like ALTER INDEX ... REBUILD ONLINE or DBMS_REDEFINITION.FINISH_REDEF_TABLE already use individual solutions to avoid ORA-00054, but for the majority of DDL commands it is required to repeat the command until the lock can be allocated successfully.
- With Oracle 11g a new parameter **DDL_LOCK_TIMEOUT** is introduced that can be set to a number of seconds the DDL command should aggressively wait for the lock.
- Any INSERT / UPDATE / DELETE to the table will be blocked during that wait time, so DDL_LOCK_TIMEOUT should not be set too high.
- If the lock can’t be retrieved and the timeout is reached, an ORA-00054 is thrown.

Agenda

Introduction

1. SQL Monitoring
2. Extended Statistics
3. SQL Repair Advisor
4. I/O Calibration
5. Automatic Shared Pool Extensions
6. DDL Lock Timeout
7. **Unrecoverable SCN Tracking**
8. Nested Loop Join Implementation
9. Full Table Scan Implementation
10. Maintenance Windows and Resource Manager

Unrecoverable SCN Tracking

Overview

- If actions are executed with NOLOGGING, the „unrecoverable SCN“ is updated in the controlfile, so that RMAN knows about activities that can‘t be recovered properly. This information is usually not required in SAP environments.
- In case of parallelized and concurrent NOLOGGING operations (e.g. creation of partitioned indexes during BW data loads) the controlfiles become a bottleneck and controlfile enqueue waits (**„enq: CF – contention“**) show up.
- As of Oracle 11.2.0.2 the following parameter can be used to switch the controlfile updates off:

  DB_UNRECOVERABLE_SCN_TRACKING = FALSE
- Due to a bug it may not work with early 11.2.0.2 versions, then event 10359 can be set to level 1 as a workaround. This event is also available with earlier Oracle releases.

Agenda

Introduction

1. SQL Monitoring
2. Extended Statistics
3. SQL Repair Advisor
4. I/O Calibration
5. Automatic Shared Pool Extensions
6. DDL Lock Timeout
7. Unrecoverable SCN Tracking
8. **Nested Loop Join Implementation**
9. Full Table Scan Implementation
10. Maintenance Windows and Resource Manager

Nested Loop Join Implementation

Overview

- Nested Loop Joins are optimized in two ways with Oracle 11g:
  - The inner table is not immediately accessed for each record found in the inner index. Instead a batch of records is accessed at once (Nested Loop Batching)
  - The position in the inner index is kept and can be reused if the next record is found in the same area. No complete INDEX RANGE SCAN or INDEX UNIQUE SCAN is necessary any more. This can significantly reduce the amount of buffer gets.
- Nested Loop Batching results in a surprising change in the execution plan: Two NESTED LOOPS steps show up when two tables are joined – the first for the loop on the inner index, the second for the decoupled batch accesses to the inner table.
- See file `Example_NestedLoopJoins.txt` for more details.

Agenda

Introduction

1. SQL Monitoring
2. Extended Statistics
3. SQL Repair Advisor
4. I/O Calibration
5. Automatic Shared Pool Extensions
6. DDL Lock Timeout
7. Unrecoverable SCN Tracking
8. Nested Loop Join Implementation
9. **Full Table Scan Implementation**
10. Maintenance Windows and Resource Manager

Full Table Scan Implementation

Overview

- As of Oracle 11g **full table scans** on large tables (> 2 % of buffer pool size) read blocks directly from disk („**direct path read**“) rather than loading them into the buffer pool („db file scattered read“).
- This is typically useful, because in the past the blocks were put at the cold end of the buffer pool and so they were replaced quickly. Now this overhead is avoided.
- As a consequence the wait time for „db file scattered read“ can go down significantly with Oracle 11g while the wait time for „direct path read“ increases.

Agenda

Introduction

1. SQL Monitoring
2. Extended Statistics
3. SQL Repair Advisor
4. I/O Calibration
5. Automatic Shared Pool Extensions
6. DDL Lock Timeout
7. Unrecoverable SCN Tracking
8. Nested Loop Join Implementation
9. Full Table Scan Implementation
10. Maintenance Windows and Resource Manager

Maintenance Windows and Resource Manager

Overview

- Both Maintenance Windows and Resource Manager already exist since Oracle 10g.
- They never caused trouble because the Resource Manager was not active.
- With 11g a Resource Manager Plan is assigned to the Maintenance Windows so that a resource limitation can take place during Maintenance Windows and waits for „**resmgr:cpu quantum**“ or „**resmgr:resource group CPU method**“ are possible.
- See `CaseStudy_ResourceManager.txt` for a real life example.
- *SQL: „ Configuration_MaintenanceWindows.txt“* displays Maintenance Windows information
- It‘s recommended to remove the Resource Manager assignment from the Maintenance Windows according to SAP Note 1579946.
- SAP Note 1589924 describes how the Resource Manager can be activated and used on purpose.

Thank you!