Postbank

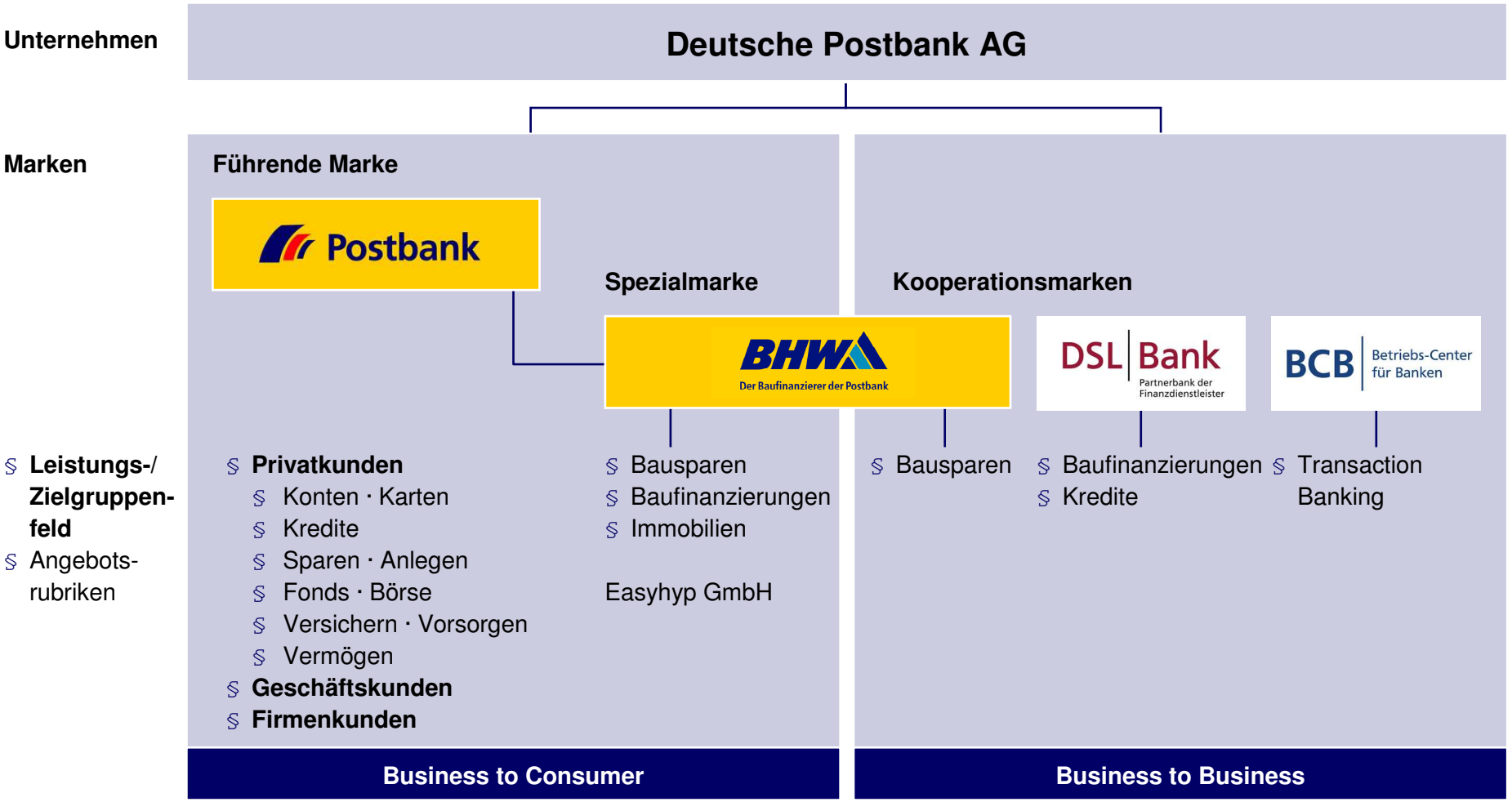
Datenbank-Tuning mit „TuTool“ im täglichen Betrieb

Jens-Christian Pokolm

Postbank

Agenda

Angebote der Postbank nach Marken



Postbank auf einen Blick: Eine der größten Privatkundenbanken Deutschlands

- 14 Millionen Kunden (Nummer 1 in Deutschland)
  davon: 260.000 Geschäftskunden, 30.000 Firmenkunden
- 21.000 Beschäftigte plus 4.000 mobile Berater
- Marktführer in Kernprodukten des Privatkundengeschäfts
- Multikanalstruktur über alle Vertriebswege
- Dichtestes Filialnetz einer Bank in Deutschland
- Führend im Online- und Telefonbanking,
  meistgenutztes Online-Angebot
- Führende voll integrierte Transaktionsbank in Deutschland

Multikanalbank: Eckdaten des Postbank-Vertriebs

Filialen

- 1.100 Postbank Finanzcenter, mehrere tausend Filialen der Deutschen Post
- 1.000 Beratungscenter der Postbank Finanzberatung

Mobiler Vertrieb

- 4.000 mobile Berater
- Beratung auch zu Hause beim Kunden

Direktvertrieb

- Call-Center 7 x 24 Stunden
- Online-Banking und Online-Broking
- Mobile Banking und Mobile Brokerage
- Dialogmarketing
- 3.100 Postbank Geldautomaten (inkl. Shell), plus Cash Group-Verbund
- 1.300 Kontoauszugsdrucker/Serviceterminals

Drittvertrieb

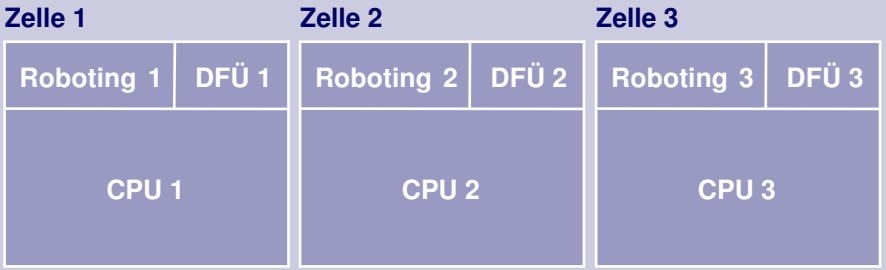
- DSL Bank mit ihren Vertriebspartnern
- Maklervertrieb von BHW

Standorte


Hamburg
Hannover
Hameln
Berlin
Bonn
Frankfurt
Saarbrücken
Nürnberg
München
Standort
Niederlassung
Außenstelle

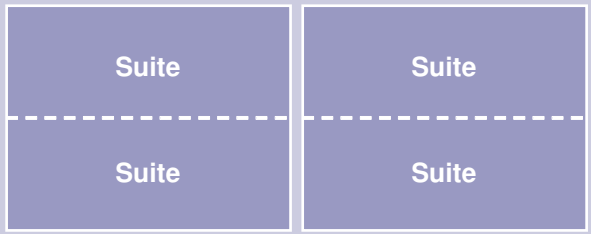
Rechenzentrum der Postbank Systems AG

Zentrales Rechenzentrum Bonn



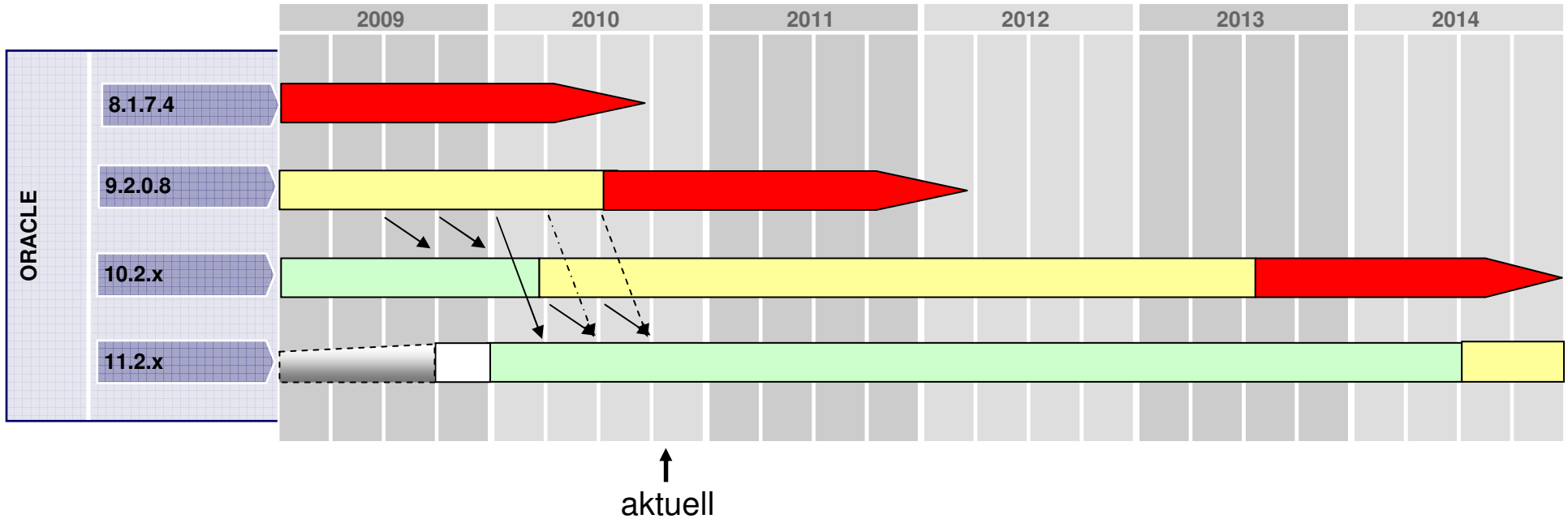
- Drei vollständig autarke Zellen mit jeweils ca. 470 m^2 Fläche
- Stromversorgung und Klimatisierung für 0,64 kW/m^2 (n+1)
- Erdbebensicherheit durch Installation der Komponenten auf Schwingboden
- Konzerneigenes Gebäude, eigener Betrieb der RZ-Infrastruktur (Gebäudesicherheit, Energie - und Klimaversorgung, Brandschutz)

Erweiterungsrechenzentrum Frankfurt



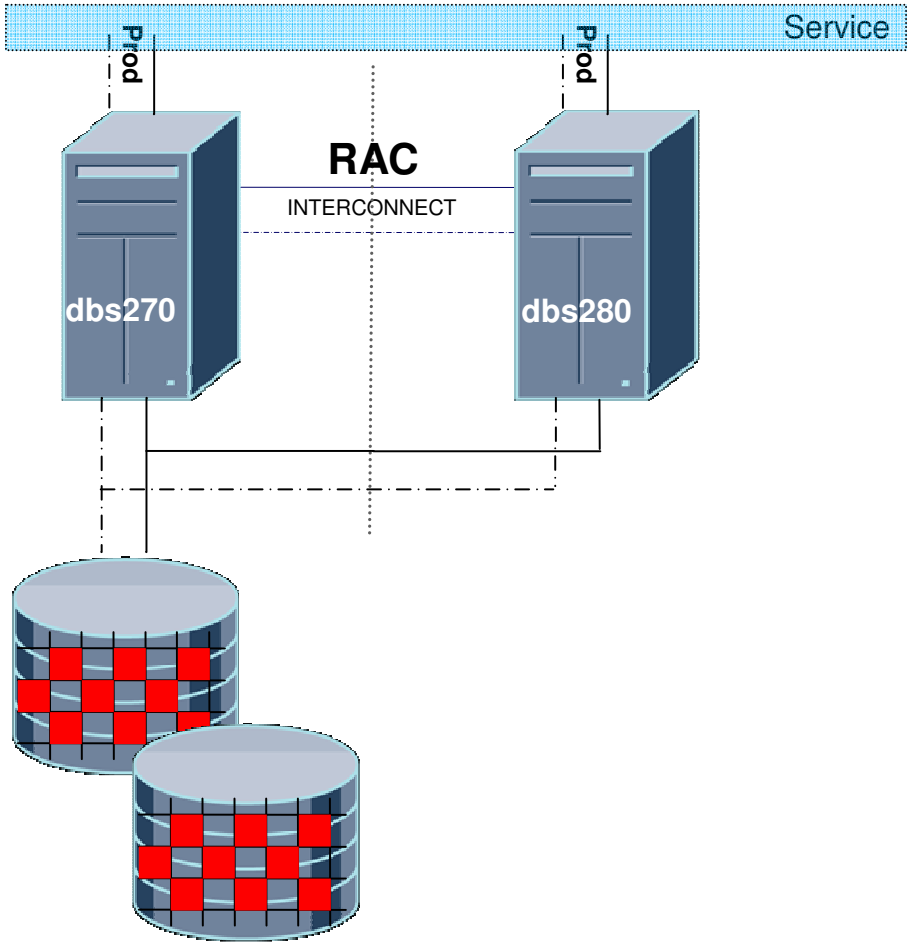
- Vier Suiten mit jeweils ca. 450 m^2 Fläche
- Stromversorgung und Klimatisierung für 1,0 KW/m^2 dimensioniert (doppelt redundant n+1)
- Anmietung von betriebenen Flächen

Lifecycle – Releaseplanung Oracle



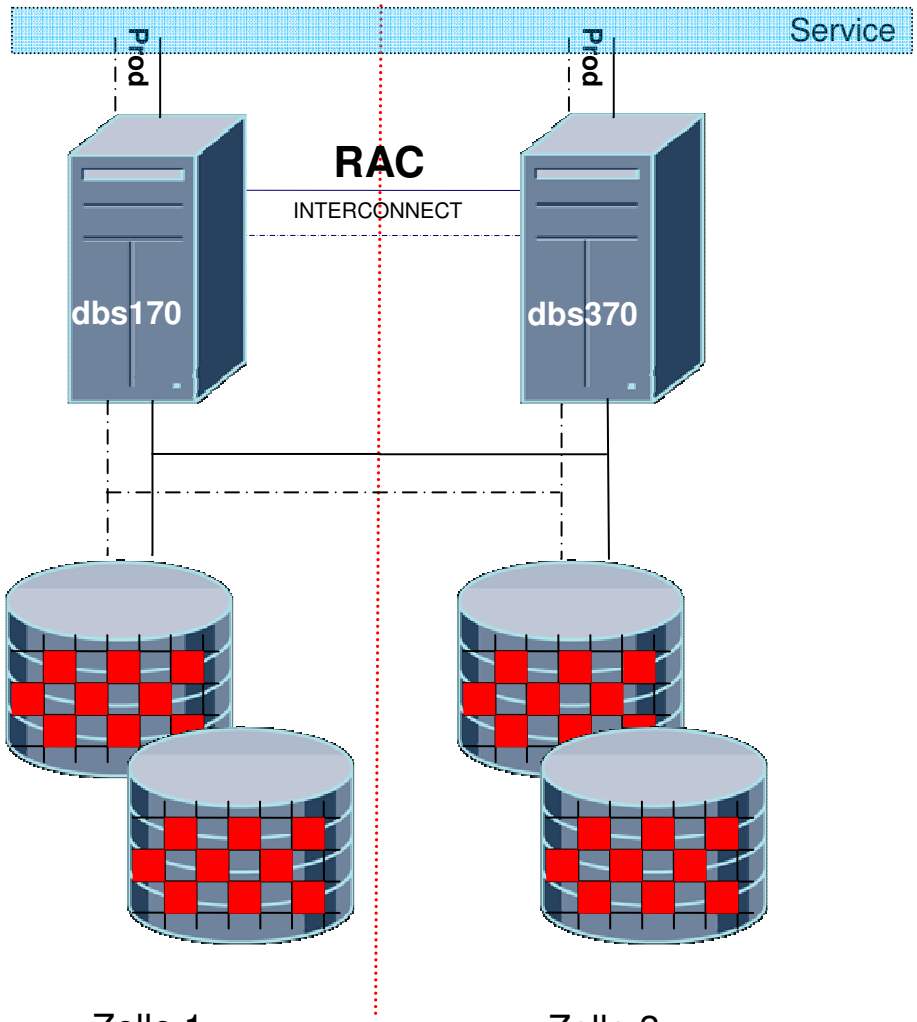
- Erstes Major-Release nur zu Tests und Evaluierung
- Zweites Release zur Produktion
- Seit 6/2008 Beta-Programm ORACLE 11.2

Einführungsphase | Reg. Produktion | Decommissioning | Risiko



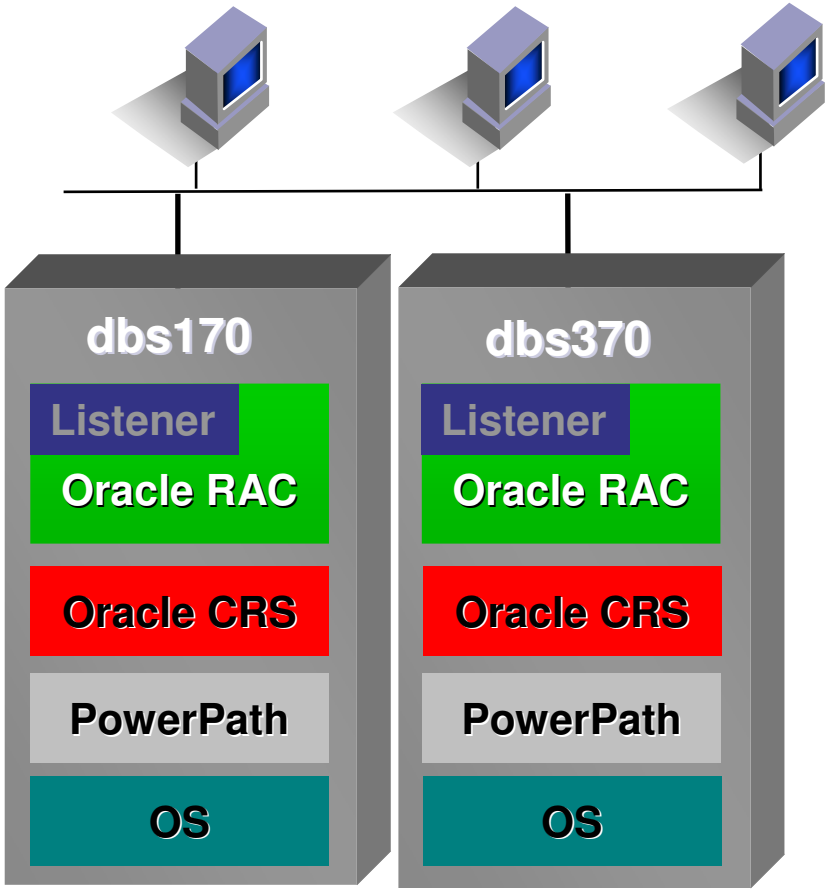
- Partitionierte od. separate Server
- Singuläres Speichersystem
- Speicher Midrange oder Enterprise
- Betriebssystem AIX / Solaris / Linux

ZRZ oder ERZ



- 2 separate räumlich getrennte Server
- Enterprise Storage - redundant
- Betriebssystem AIX / Solaris / Linux

ZRZ oder ERZ



Bisheriger Software-Stack in der Produktion

- AIX 5.3 , Sun Solaris 10, und SLES 9/10
- EMC Powerpath
- IBM Tivoli Data Protection for Oracle v5.3
- Oracle Clusterware 10.2.0.4 + Patch-Bundle 4 + PSU
- Oracle ASM 10.2.0.4 + PSU
- Oracle Database EE 10.2.0.4 + PSU

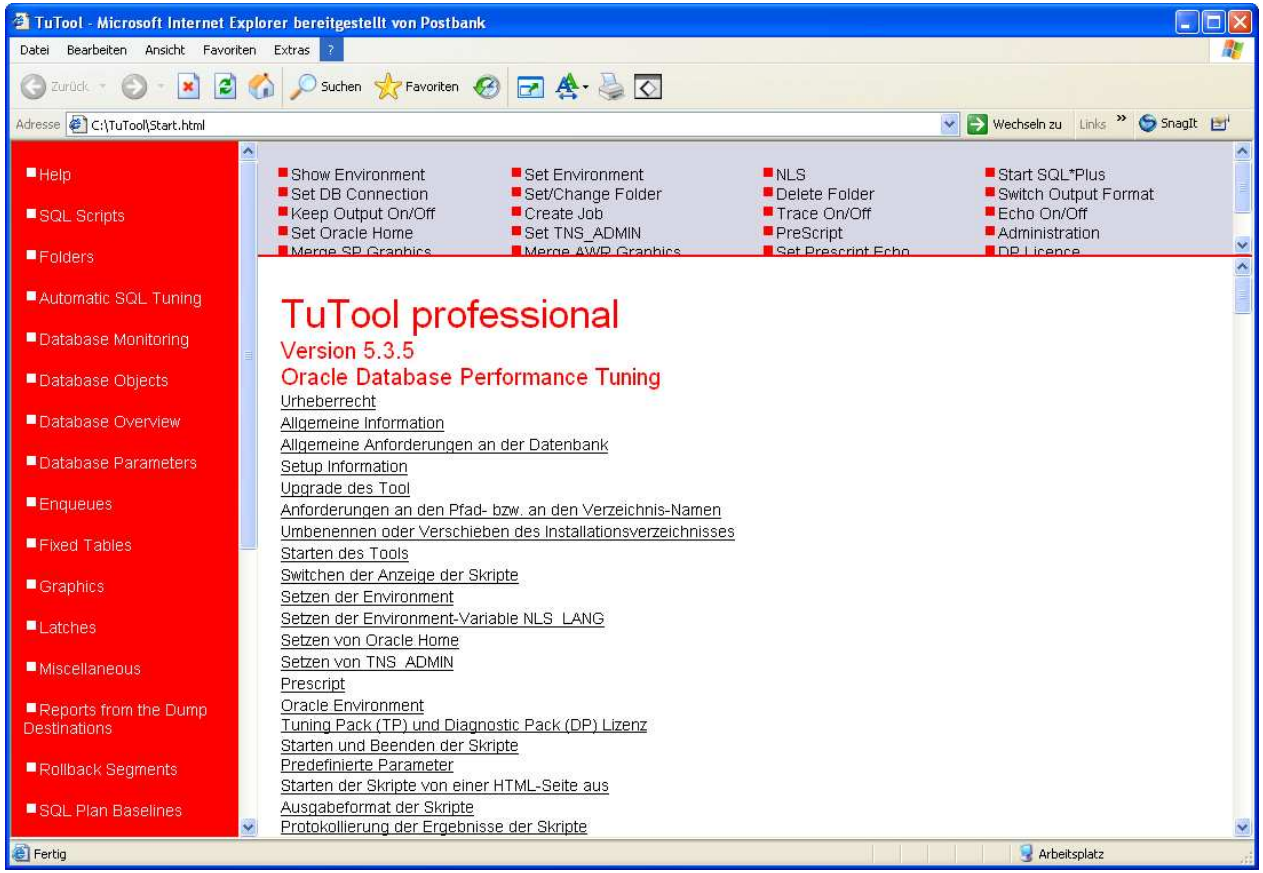
Zukünftiger Software-Stack in der Produktion (ab 2010)

- AIX 6.1, Sun Solaris 10, SLES 10
- EMC Powerpath
- IBM Tivoli Data Protection for Oracle v5.5
- Oracle Grid Infrastructure 11.2.0.1
- Oracle Database EE 11.2.0.1

TuTool – ein Überblick

Die Postbank Systems AG hat eine Unternehmenslizenz der Software TuTool gekauft die es allen DBA‘s ermöglicht das Produkt sowohl im Test als auch der Produktion zu nutzen.

- Browser-Oberfläche
- Läuft auf allen Windows
- Umgebungen
- Installation von TuTool kann als auch als „Read-Only“ erfolgen
- Alle Ausgaben werden in Dateien protokolliert und sind somit revisionssicher dokumentierbar
- Grafische Ausgaben werden als PNG (Portable Networks Graphic) gespeichert

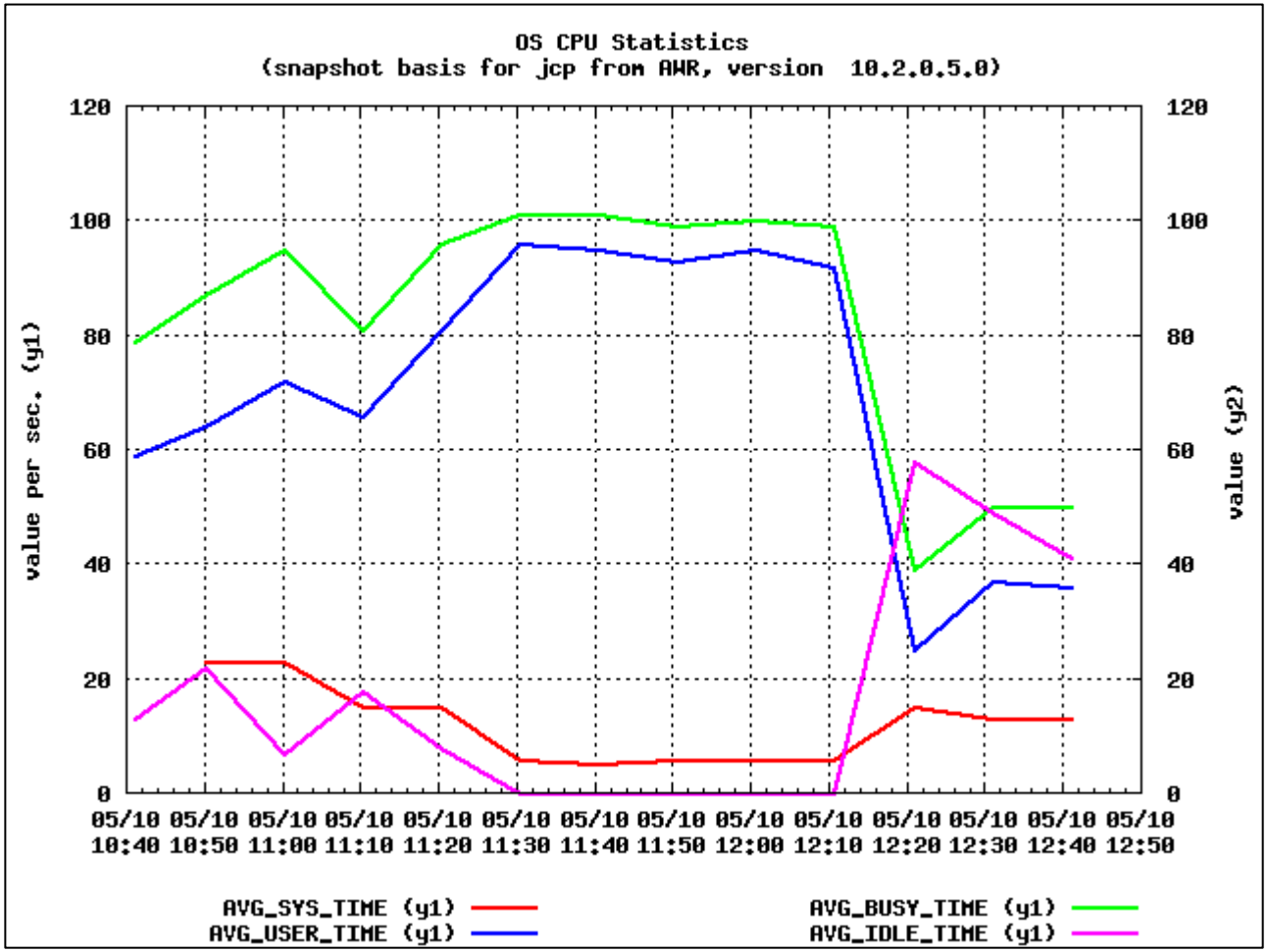


Ein ganz einfaches und alltägliches Szenario:

Anruf der Fachabteilung gegen 14 Uhr – die Auswertungen sind seit ein paar Stunden sehr langsam.

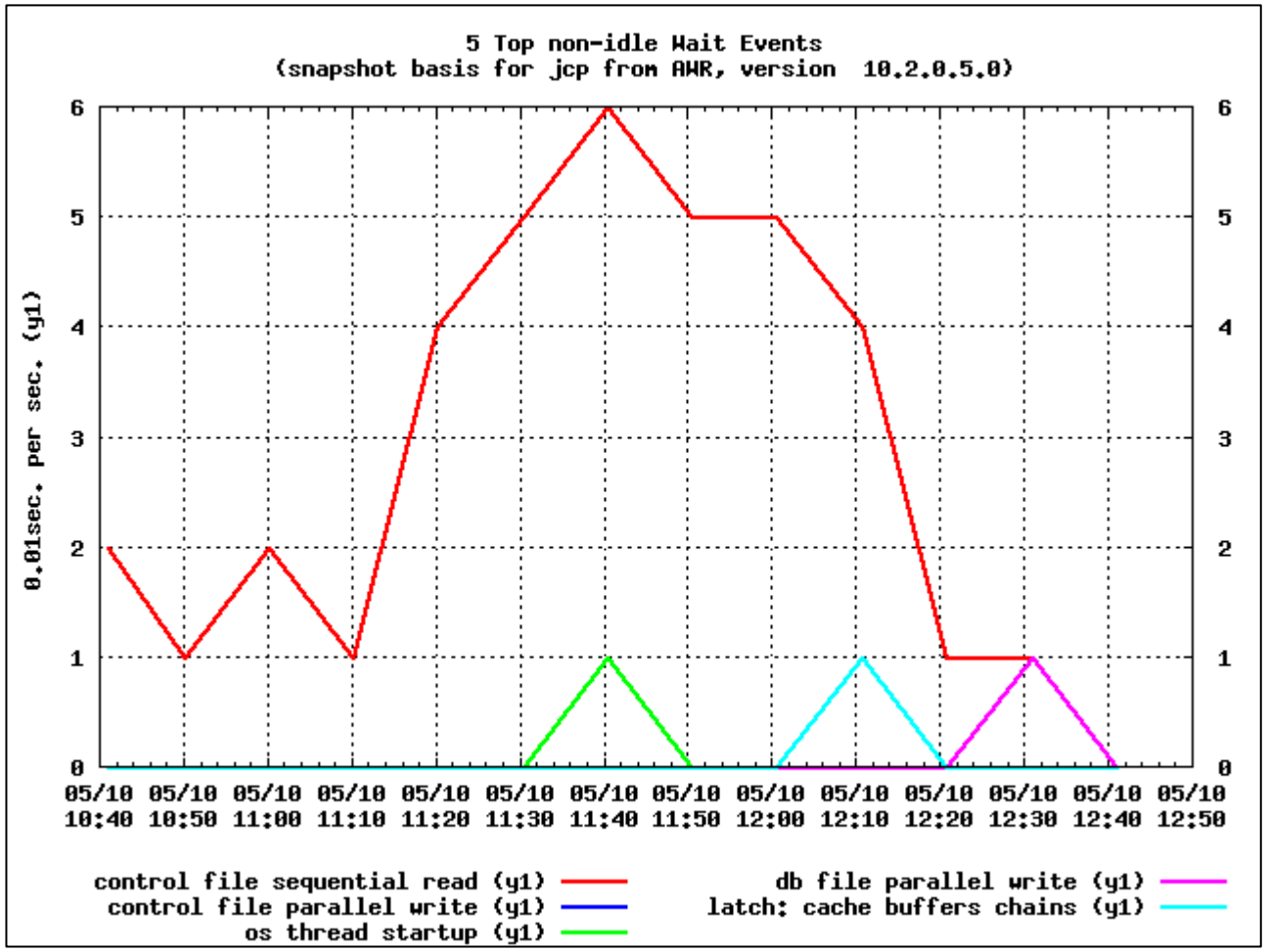
Die Anmeldung an der Datenbank in der Mittagszeit war nahezu unmöglich …

Start der Analyse mittels `awr_cpu_osstats_graph102`



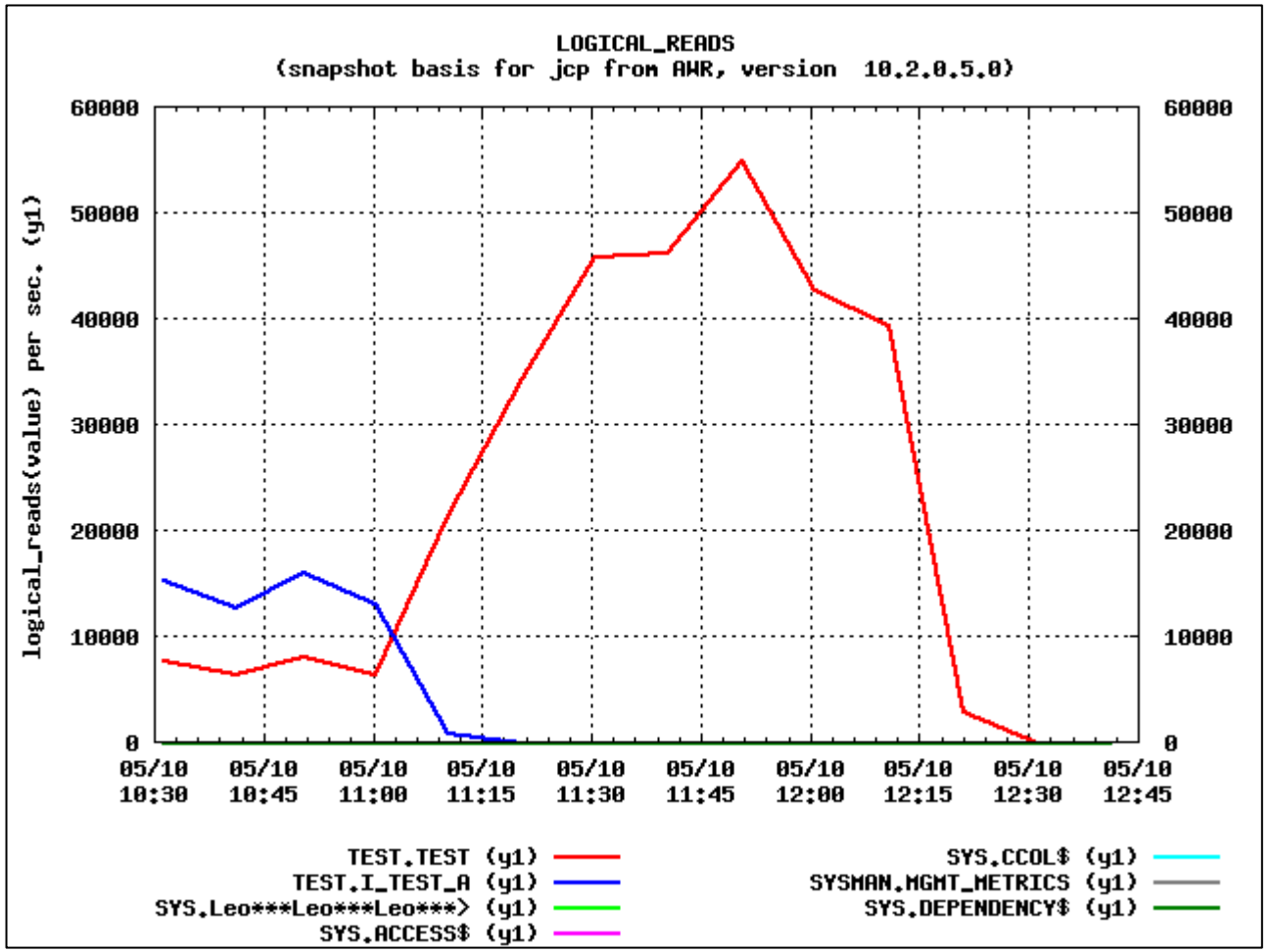
Wir sehen – die CPU war von ca. 11:30 Uhr bis gegen 12:10 zu 100% ausgelastet – aber warum ?

Suche im AWR nach den TOP5 Events – mittels `awr_5_top_events_graph`



Keinerlei Waits zu sehen – das Problem liegt also scheinbar im CPU-Bereich.
CPU-Lasten werden meistens durch LOGICAL_READ‘s ausgelöst.

Ein Blick auf die Segmente per `awr_seg_stats_bgraph10g`



Bis 11:00 Uhr viele Logical Reads auf dem Index TEST.I_TEST_A aber danach nahezu ausschließlich auf der Tabelle TEST.TEST

Postbank

Über `awr_sqltus10g` sehen wir uns nun einmal die Ausführungspläne CPU-intensiver Abfragen an.

```
---- GROUP=10578, SNAP_ID <= 10578, SNAP_TIME <= 05.10.2010 10:31:00, SQL_ID=0qgh9bfzasq6h
---- DISK_READS=0, BUFFER_GETS=13495425, DIRECT_WRITES=0, EXECUTIONS=4498475, ROWS_PROCESSED=4498475, PARSE_CALLS=31
---- DISK_READS_PER_EX=0, BUFFER_GETS_PER_EX=3, DIRECT_WRITES_PER_EX=0, ROWS_PROCESSED_PER_EX=1, PARSE_CALLS_PER_EX=0
---- DISK_READS_PER_ROW=0.0000, BUFFER_GETS_PER_ROW=3.0000
---- CPU_TIME (sec.)=231.7984, ELAPSED_TIME (sec.)=246.6521, WAIT_TIME (sec.)=14.8537
---- CPU_TIME_PER_EX (sec.)=0.0001, ELAPSED_TIME_PRO_EX (sec.)=0.0001, WAIT_TIME_PRO_EX (sec.)=0.0000
---- PLSQL_EXEC_TIME (sec.)=0.0000, JAVA_EXEC_TIME (sec.)=0.0000
---- PLSQL_EXEC_TIME_PER_EX (sec.)=0.0000, JAVA_EXEC_TIME_PRO_EX (sec.)=0.0000
---- APPLICATION_WAIT_TIME (sec.)=0.0000, CONCURRENCY_WAIT_TIME (sec.)=0.0000, CLUSTER_WAIT_TIME (sec.)=0.0000, USER_IO_WAIT_TIME (
---- APPLICATION_WAIT_TIME_PER_EX (sec.)=0.0000, CONCURRENCY_WAIT_TIME_PER_EX (sec.)=0.0000, CLUSTER_WAIT_TIME_PER_EX (sec.)=0.0000
---- MODULE : sqlplus.exe
---- FORCE_MATCHING_SIGNATURE = 12177105831857057174
---- OPTIMIZER_MODE = ALL_ROWS, OPTIMIZER_ENV_HASH_VALUE = 3897423573
SELECT E FROM TEST.TEST WHERE A=:B1 AND ROWNUM <= 1
---- Execution Plan (Plan Hash Value : 1909919657) :
SELECT STATEMENT Optimizer=ALL_ROWS (Cost=2)
  COUNT (STOPKEY) (Bind Peeking used)
    TABLE ACCESS (BY INDEX ROWID) OF TEST (Cost=2 Card=1 Bytes=10 CPU_Cost=15523 IO_Cost=2 Time=1)
      INDEX (RANGE SCAN) OF I_TEST_A (Cost=1 Card=1 CPU_Cost=8171 IO_Cost=1 Time=1)
```

Sieht doch gut aus um 10:31 Uhr – Zugriffe erfolgen per Index (Range Scan).

Postbank

Und zu einem späteren Zeitpunkt – das identische Statement …

11:10 Uhr – FULL TABLE SCAN

```
---- GROUP=10582, SNAP_ID <= 10582, SNAP_TIME <= 05.10.2010 11:10:08, SQL_ID=0qgh9bfzasq6h
---- DISK_READS=0, BUFFER_GETS=12791153, DIRECT_WRITES=0, EXECUTIONS=76232, ROWS_PROCESSED=76231, PARSE_CALLS=1
---- DISK_READS_PER_EX=0, BUFFER_GETS_PER_EX=167.79, DIRECT_WRITES_PER_EX=0, ROWS_PROCESSED_PER_EX=1, PARSE_CALLS_PER_EX=0
---- DISK_READS_PER_ROW=0.0000, BUFFER_GETS_PER_ROW=167.7946
---- CPU_TIME (sec.)=381.3858, ELAPSED_TIME (sec.)=457.2505, WAIT_TIME (sec.)=75.8646
---- CPU_TIME_PER_EX (sec.)=0.0050, ELAPSED_TIME_PRO_EX (sec.)=0.0060, WAIT_TIME_PRO_EX (sec.)=0.0010
---- PLSQL_EXEC_TIME (sec.)=0.0000, JAVA_EXEC_TIME (sec.)=0.0000
---- PLSQL_EXEC_TIME_PER_EX (sec.)=0.0000, JAVA_EXEC_TIME_PRO_EX (sec.)=0.0000
---- APPLICATION_WAIT_TIME (sec.)=0.0000, CONCURRENCY_WAIT_TIME (sec.)=0.0000, CLUSTER_WAIT_TIME (sec.)=0.0000, USER_IO_WAIT_TIME
---- APPLICATION_WAIT_TIME_PER_EX (sec.)=0.0000, CONCURRENCY_WAIT_TIME_PER_EX (sec.)=0.0000, CLUSTER_WAIT_TIME_PER_EX (sec.)=0.00
---- MODULE : sqlplus.exe
---- FORCE_MATCHING_SIGNATURE = 12177105831857057174
---- OPTIMIZER_MODE = ALL_ROWS, OPTIMIZER_ENV_HASH_VALUE = 1481392096
SELECT E FROM TEST.TEST WHERE A=:B1 AND ROWNUM <= 1
---- Execution Plan (Plan Hash Value : 3931117773) :
SELECT STATEMENT Optimizer=ALL_ROWS (Cost=144)
  COUNT (STOPKEY) (Bind Peeking used)
    TABLE ACCESS (FULL) OF TEST (Cost=144 Card=1 Bytes=10 CPU_Cost=34548016 IO_Cost=138 Time=2)
```

Alternativer Suchansatz – wer nicht lange Listen durchsuchen will … `awr_obj_sqltus10g`

```
-- Database Alias : jcp
-- Oracle Server Version :  10.2.0.5.0
-- Script awr_obj_sqltus10g.sql (Product TuTool 6.1.0 : www.tutool.de)
-- Start Time : 26.10.10 14:06:23
-- Input Parameters :
-- begin_time='01.01.2000'
-- end_time=''
-- num_stmt=''
-- group='s'
-- with_sqltext=''
-- trunc_sqltext_to=''
-- with_exec_plan=''
-- object_owner=''
-- object_name='TEST'
-- object_type=''
-- criteria='CT'
```

```
---- GROUP=10581, SNAP_ID <= 10581, SNAP_TIME <= 05.10.2010 11:00:05, SQL_ID=0qgh9bfza..
---- DISK_READS=0, BUFFER_GETS=12202257, DIRECT_WRITES=0, EXECUTIONS=4067419, ROWS_PRO..
---- DISK_READS_PER_EX=0, BUFFER_GETS_PER_EX=3, DIRECT_WRITES_PER_EX=0, ROWS_PROCESSED..
---- CPU_TIME (sec.)=211.0315, ELAPSED_TIME (sec.)=236.9029, WAIT_TIME (sec.)=25.8713
---- CPU_TIME_PER_EX (sec.)=0.0001, ELAPSED_TIME_PRO_EX (sec.)=0.0001, WAIT_TIME_PRO_E..
---- PLSQL_EXEC_TIME (sec.)=0.0000, JAVA_EXEC_TIME (sec.)=0.0000
---- PLSQL_EXEC_TIME_PER_EX (sec.)=0.0000, JAVA_EXEC_TIME_PRO_EX (sec.)=0.0000
---- APPLICATION_WAIT_TIME (sec.)=0.0000, CONCURRENCY_WAIT_TIME (sec.)=0.0000, CLUSTER..
---- APPLICATION_WAIT_TIME_PER_EX (sec.)=0.0000, CONCURRENCY_WAIT_TIME_PER_EX (sec.)=0..
---- MODULE : sqlplus.exe
---- FORCE_MATCHING_SIGNATURE = 12177105831857057174
---- OPTIMIZER_MODE = ALL_ROWS, OPTIMIZER_ENV_HASH_VALUE = 3897423573
SELECT E FROM TEST.TEST WHERE A=:B1 AND ROWNUM <= 1
---- Execution Plan (Plan Hash Value : 1909919657) :
SELECT STATEMENT Optimizer=ALL_ROWS (Cost=2)
  COUNT (STOPKEY) (Bind Peeking used)
    TABLE ACCESS (BY INDEX ROWID) OF TEST (Cost=2 Card=1 Bytes=10 CPU_Cost=15523 IO_Co..
      INDEX (RANGE SCAN) OF I_TEST_A (Cost=1 Card=1 CPU_Cost=8171 IO_Cost=1 Time=1)
```

Auf zur Reparatur ….

Immer noch falsch …

Postbank

```
-- Database Alias : jcp
-- Oracle Server Version :  10.2.0.5.0
-- Script explain_plan10g.sql (Product TuTool 6.1.0 : www.tutool.de)
-- Start Time : 26.10.10 14:17:52

Non-Default System Optimizer Environment:

optimizer_index_cost_adj = 9999

SQL Statement:

1  explain plan set statement_id = 'TUTOOL_20101026141754' into sys.plan_table for
2* select e from test.test where a=:b1 and rownum <= 1

Plan Hash Value: 3931117773

0    SELECT STATEMENT Optimizer=ALL_ROWS (Cost=142 Card=1 Bytes=10 CPU_Cost=494..
1  0   COUNT (STOPKEY)
2  1     TABLE ACCESS (FULL) OF TEST (TABLE) (Cost=142 Card=1 Bytes=10 CPU_Cost..
```

Immer noch falsch (2) …

Postbank

```
Predicate Information (identified by operation id):

    1 - filter (ROWNUM<=1)
    2 - filter ("A"=TO_NUMBER(:B1))

Query Block Name / Object Alias (identified by operation id):

     1 - SEL$1           /
     2 - SEL$1           / TEST@SEL$1

Outline Data:

/*+
BEGIN_OUTLINE_DATA
FULL(@"SEL$1" "TEST"@"SEL$1")
OUTLINE_LEAF(@"SEL$1")
ALL_ROWS
OPT_PARAM('optimizer_index_cost_adj' 9999)
OPTIMIZER_FEATURES_ENABLE('10.2.0.5')
IGNORE_OPTIM_EMBEDDED_HINTS
END_OUTLINE_DATA
*/

Dynamic Sampling = no, SQL Profile not used, Stored Outline not used
```

Wir starten nun das Script `sql_prof_with_hints_from_awr10g.sql` um den alten Ausführungsplan aus dem AWR wieder herzustellen.

```
-- Database Alias : jcp
-- Oracle Server Version :  10.2.0.5.0
-- Script sql_prof_with_hints_from_awr10g.sql (Product TuTool 6.1.0 : www.tutool.de)
-- Start Time : 26.10.10 14:32:10
-- Input Parameters :
-- sql_id='0qgh9bfzasq6h'
-- plan_hash_value='1909919657'
-- sql_profile='TEST_1001'
-- sqltune_category=''
-- force_match='true'




sql_profile TEST_1001 (category DEFAULT) to fix execution plan with hash_value = 1909919657 for
sql_id = 0qgh9bfzasq6h created
```

Die Kategorie DEFAULT wirkt sofort ohne spezielle Einstellung.

Und wieder alles OK …

Postbank

```
-- Database Alias : jcp
-- Oracle Server Version :  10.2.0.5.0
-- Script explain_plan10g.sql (Product TuTool 6.1.0 : www.tutool.de)
-- Start Time : 26.10.10 16:51:34

Non-Default System Optimizer Environment:
optimizer_index_cost_adj = 9999

SQL Statement:

  1  explain plan set statement_id = 'TUTOOL_20101026165137' into sys.plan_table for
  2* select e from test.test where a=:b1 and rownum <= 1

Plan Hash Value: 1909919657

    0       SELECT STATEMENT Optimizer=HINT: ALL_ROWS (Cost=200 Card=1 Bytes=10 CPU_Cost=155..
    1     0   COUNT (STOPKEY)
    2     1     TABLE ACCESS (BY INDEX ROWID) OF TEST (TABLE) (Cost=200 Card=1 Bytes=10 CPU_..
    3     2       INDEX (RANGE SCAN) OF I_TEST_A (INDEX) (Cost=100 Card=1 CPU_Cost=817062 IO..

Predicate Information (identified by operation id):

    1 - filter (ROWNUM<=1)
    3 - access ("A"=TO_NUMBER(:B1))

Query Block Name / Object Alias (identified by operation id):

     1 - SEL$1           /
     2 - SEL$1           / TEST@SEL$1
     3 - SEL$1           / TEST@SEL$1
```

Und wieder alles OK (2) …

```
. . .

Outline Data:

/*+
BEGIN_OUTLINE_DATA
INDEX_RS_ASC(@"SEL$1" "TEST"@"SEL$1" ("TEST"."A"))
OUTLINE_LEAF(@"SEL$1")
ALL_ROWS
OPT_PARAM('optimizer_index_cost_adj' 9999)
OPTIMIZER_FEATURES_ENABLE('10.2.0.4')
IGNORE_OPTIM_EMBEDDED_HINTS
END_OUTLINE_DATA
*/


Dynamic Sampling = no, SQL Profile "TEST_1001" used, Stored Outline not used
```

Löschen mittels:

```
begin
   sys.DBMS_SQLTUNE.DROP_SQL_PROFILE('TEST_1001');
end;
/
```

Die nächste Aufgabe

Postbank

Ein leider auch alltägliches Szenario:

Eine Abfrage einer gekauften Anwendung ist inperformant.

Optimizer wählt einen schlechten Ausführungsplan (ggf. Abhängig der Suchparameter).

Zugriff z.B. über Index wäre deutlich schneller und das Setzen von DB-Parametern wie `optimizer_index_cost_adj` ist nicht möglich.

Tuning per “Hidden Hint”

Postbank

In Oracle 10.2 gibt es über die allgemein bekannten Möglichkeit zum Tunen
(SQL-Tuning-Sets oder statische Hints im Statement) noch eine weitere (versteckte) Möglichkeit:

Vorgehensweise – Step by step:

1. Mittels
   ```
   select distinct  sid from v$mystat;
   ```
   die eigene `SID` herausfinden.

2. Das "*gute*" Statement in dieser Session ausführen.

3. In TUTOOL als User SYS mit dem PRESCRIPT
   ```
   alter session set current_schema=TEST;
   alter session set use_stored_outlines=DEFAULT;
   ```

4. Sessions Skript: `prev_sid_sql` - Eingabe der Session SID aus 1.
   Ergebnis: Der `Hash-Value` des ausgeführten SQL-Statements

Tuning per “Hidden Hint”

Postbank

5. SQL Tuning Skript: `one_exec_plan_sqlarea10g`
   Eingabe des `Hash-Values` aus 4.
   Ergebnis: `SQL-ID` , Hash-Value und Child-Number des Statement
   Info: Bind-Peaking Used

6. SQL Tuning Skript: `one_exec_xplan_sqlarea10`
   Eingabe der `SQL-ID` aus 5.
   Ergebnis: Ausführungsplan mit `Plan-Hash-Value` und Outline-Informationen
   Outline-Infos kopieren

---

7. ERSTELLUNG DER OUTLINE MIT HINTS
   Stored Outlines
   Skript: `create_outln_sqltext8i`
   Aktion: Im Fenster Hint-Teil hinzufügen Save

8. ERSTELLUNG DER OUTLINE OHNE HINTS
   Stored Outlines
   Skript: `create_outln_sqltext8i`
   Aktion: Im Fenster Hint-Teil wieder entfernen Save

Tuning per “Hidden Hint”

Postbank

9. Stored Outlines - Skript: `insert_hidden_hints_in_outln8i`
   Eingabe: Name der Outline OHNE Hints
   Danach Eingabe: Name der Outline MIT Hints Danach ist die Outline MIT Hints „weg“

---

10. Check – In neuer Session
```
alter session set use_stored_outlines=DEFAULT;
```
    Statement ausführen
```
select NAME,OWNER,CATEGORY,USED from dba_outlines;
```
    Geht auf USED

LOGON-Trigger für die Outline-Kategorie. Ist auch bei Kategorie DEFAULT notwendig!

```
CREATE OR REPLACE TRIGGER POSTBANK_DB_USER_LOGON
AFTER LOGON ON DATABASE
DECLARE
 vsql varchar2(200);
BEGIN
    vsql := 'alter session set use_stored_outlines=DEFAULT';
    execute immediate vsql;
EXCEPTION
   WHEN OTHERS THEN
      NULL;
END;
/
```

?

?

?

?

Jens-Christian Pokolm
Postbank Systems AG
Baunscheidtstraße 8
D-53113 Bonn
Telefon: +49 (0) 228 – 920 63155
E-Mail: jens-christian.pokolm@postbank.de