robotron®

DOAG Regionaltreffen Dresden

SQL ausführen – der Optimizer hat einen Plan

Marco Mischke – 19.03.2013

ORACLE Platinum Partner

Agenda

robotron®

Optimizer	• Aufgaben • Möglichkeiten • Randbedingungen
Zugriffspfade	• Welche gibt es • Wie funktionieren diese
Ausführungs-pläne	• Aufbau • Interpretation • Besondere Möglichkeiten
Gut zu wissen	• Indexe • Statistiken • Cursor Sharing

SQL ausführen

robotron®

SQL → Optimizer → Ergebnis

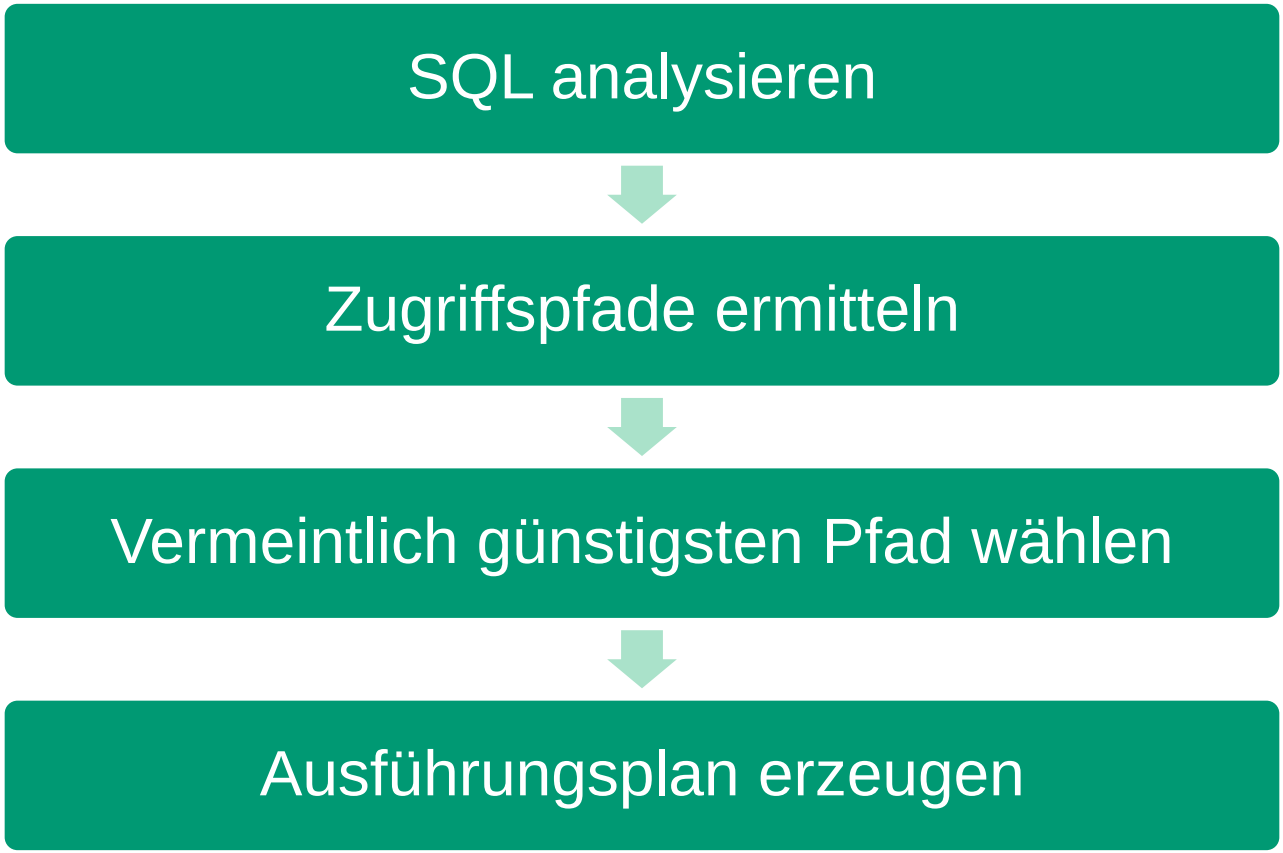
```
Select
  a.a, a.b, b.a
From
  mytable_a a,
  mytable_b b
Where
  a.id = b.id
```

Hier geschieht ein Wunder

a.a	a.b	b.a
A	B	C
AA	CC	BB
BC	BA	CB
CBA	CBO	ABA

Optimizer – Was macht der eigentlich?

robotron®



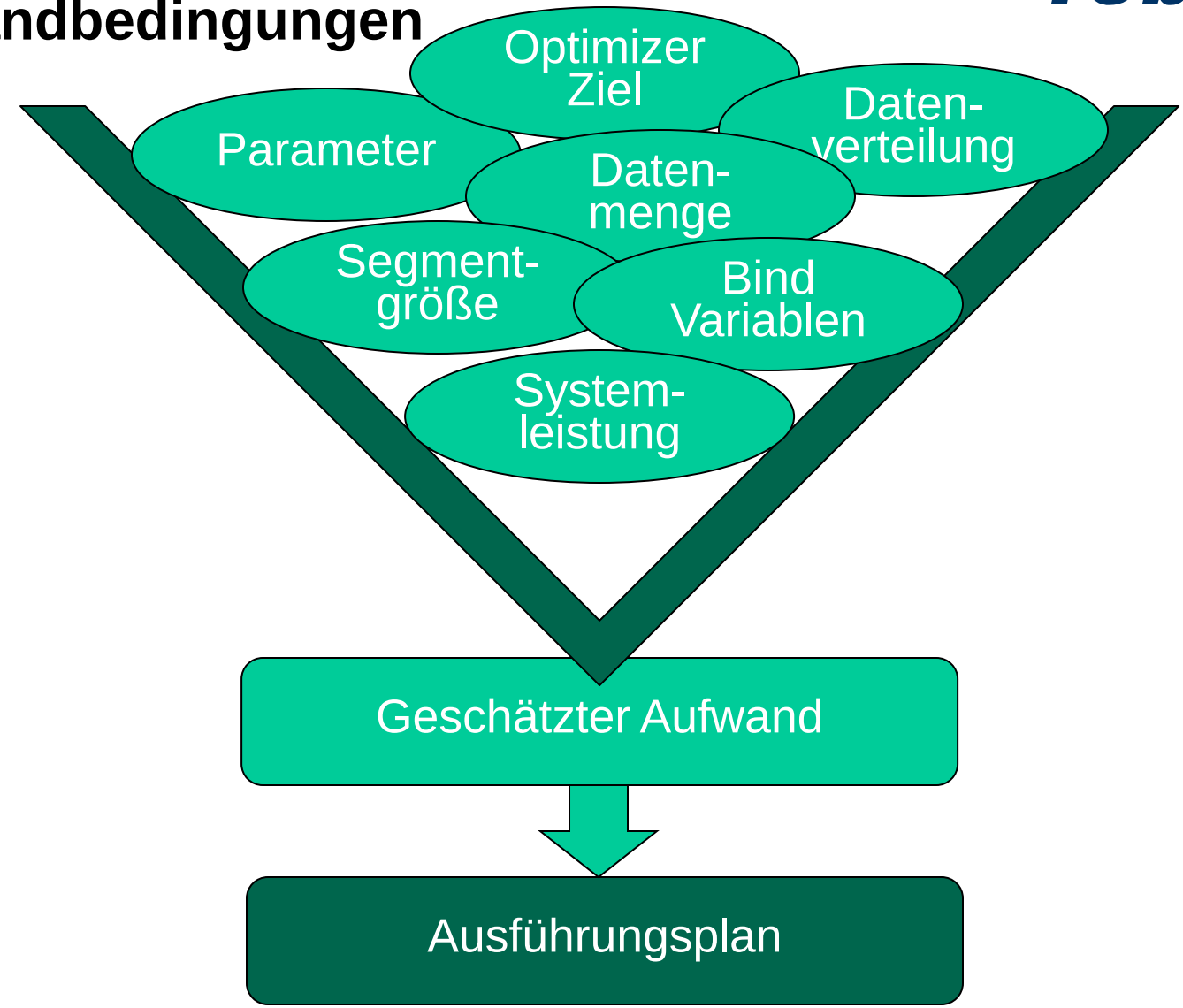
Optimizer – als die Welt noch in Ordnung war

robotron®

- Pre Oracle 8
  - „Rule of thumb“ – Rule Based Optimizer (Kurzfassung)
    1. Single Row by ROWID / Hash Cluster
    2. Unique Index
    3. Composite Index
    4. Single-Column Index
    5. Index Range Scan
    6. Index Full Scan
    7. Full Table Scan

Optimizer – Randbedingungen

robotron®



Optimizer – die heutige Welt

robotron®

- Post Oracle 8 - Cost Based Optimizer
  - Unterstützt neue Features
    - Bitmap Indexe
    - Hash Joins
    - …
  - schätzt den wahrscheinlichen Arbeitsaufwand
    - Objektstatistiken (Anzahl Zeilen, …)
    - Systemstatistiken (I/O Leistung, …)
  - Der günstigste Plan wird genommen
    - Kosten enthalten I/O, CPU, Netzwerkressourcen
    - Kosten dienen nur zum internen Vergleich der Pläne!

Agenda

robotron®

Optimizer	• Aufgaben • Möglichkeiten • Randbedingungen
Zugriffspfade	• Welche gibt es • Wie funktionieren diese
Ausführungs-pläne	• Aufbau • Interpretation • Besondere Möglichkeiten
Gut zu wissen	• Indexe • Statistiken • Cursor Sharing

Optimizer – Zugriffspfade

robotron®

singleblock I/O

ROWID	über physikalische Adresse
Index Unique Scan	über eindeutigen Wert
Index Range Scan	über Wertebereich oder nicht eindeutigen Wert
Index Skip Scan	über zusammengesetzten Index
Index Full Scan	um Daten zu sortieren

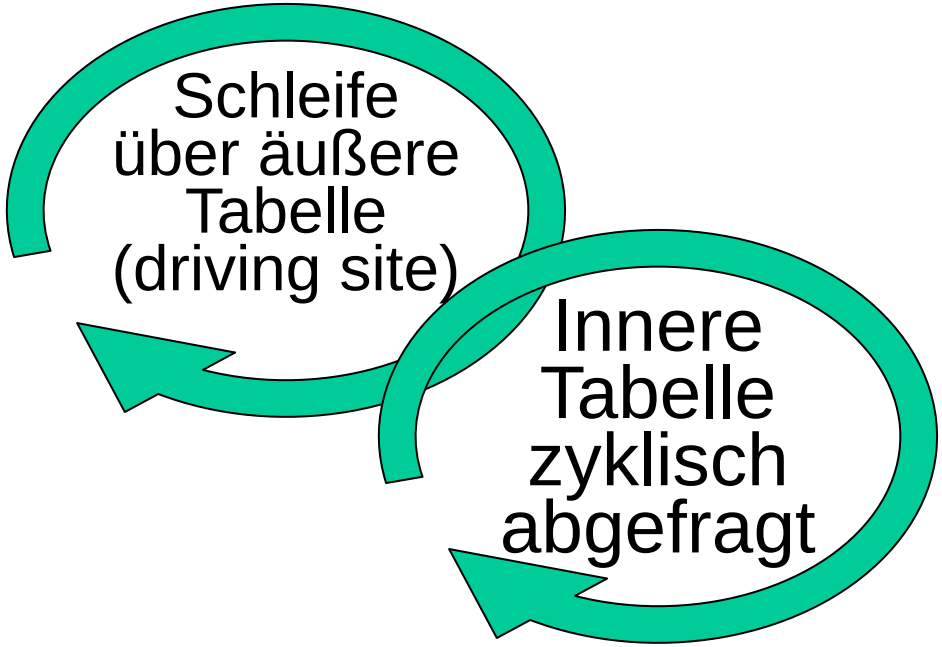
multiblock I/O

Index Fast Full Scan	alle selektierten Daten im Index
Full Table Scan	einmal alles bitte

Optimizer – Join Methoden
Nested Loop

robotron®

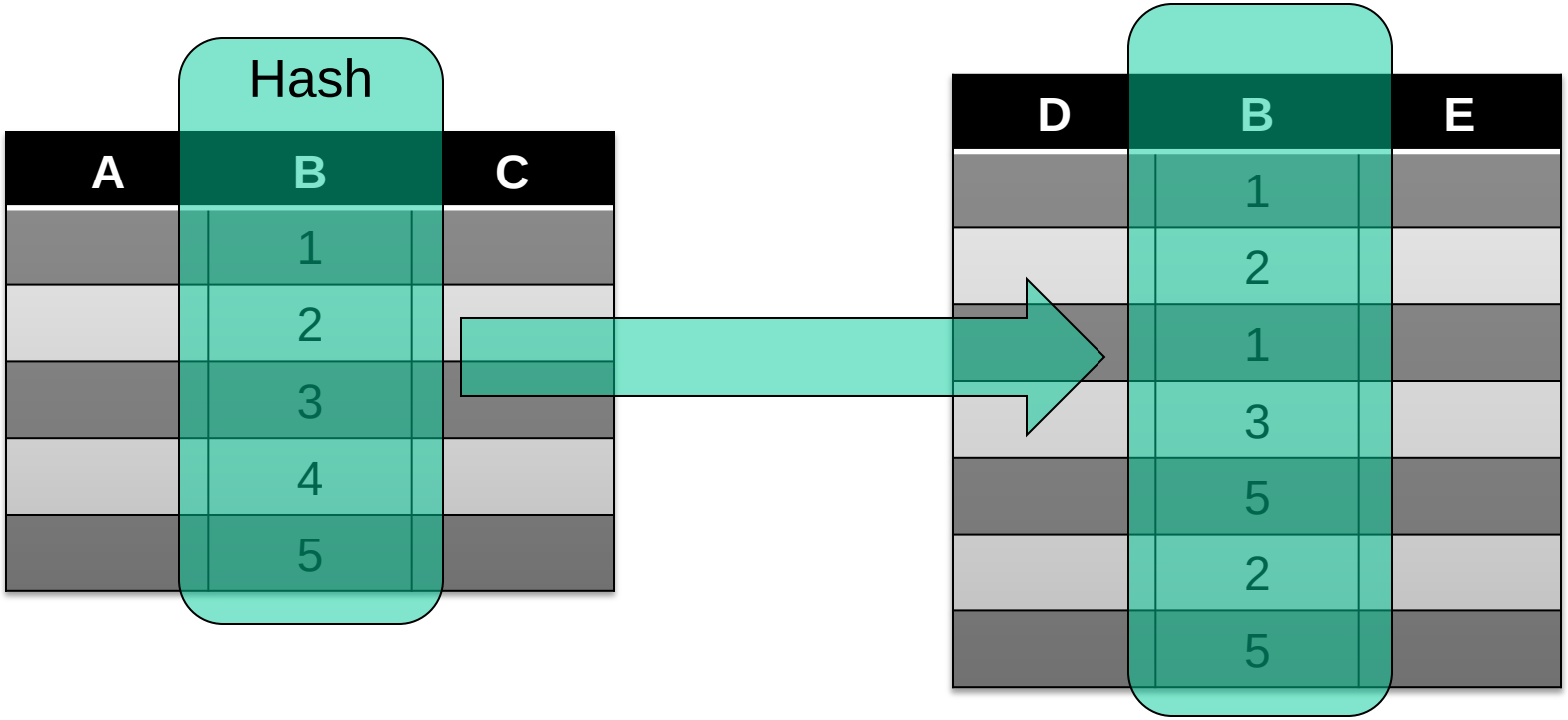
- Nested Loop
  - Geschachtelte Schleife
  - Äußere Schleife (driving site) hat wenig Datensätze
  - Erste Datensätze werden schnell geliefert



Optimizer – Join Methoden
Hash Join

robotron®

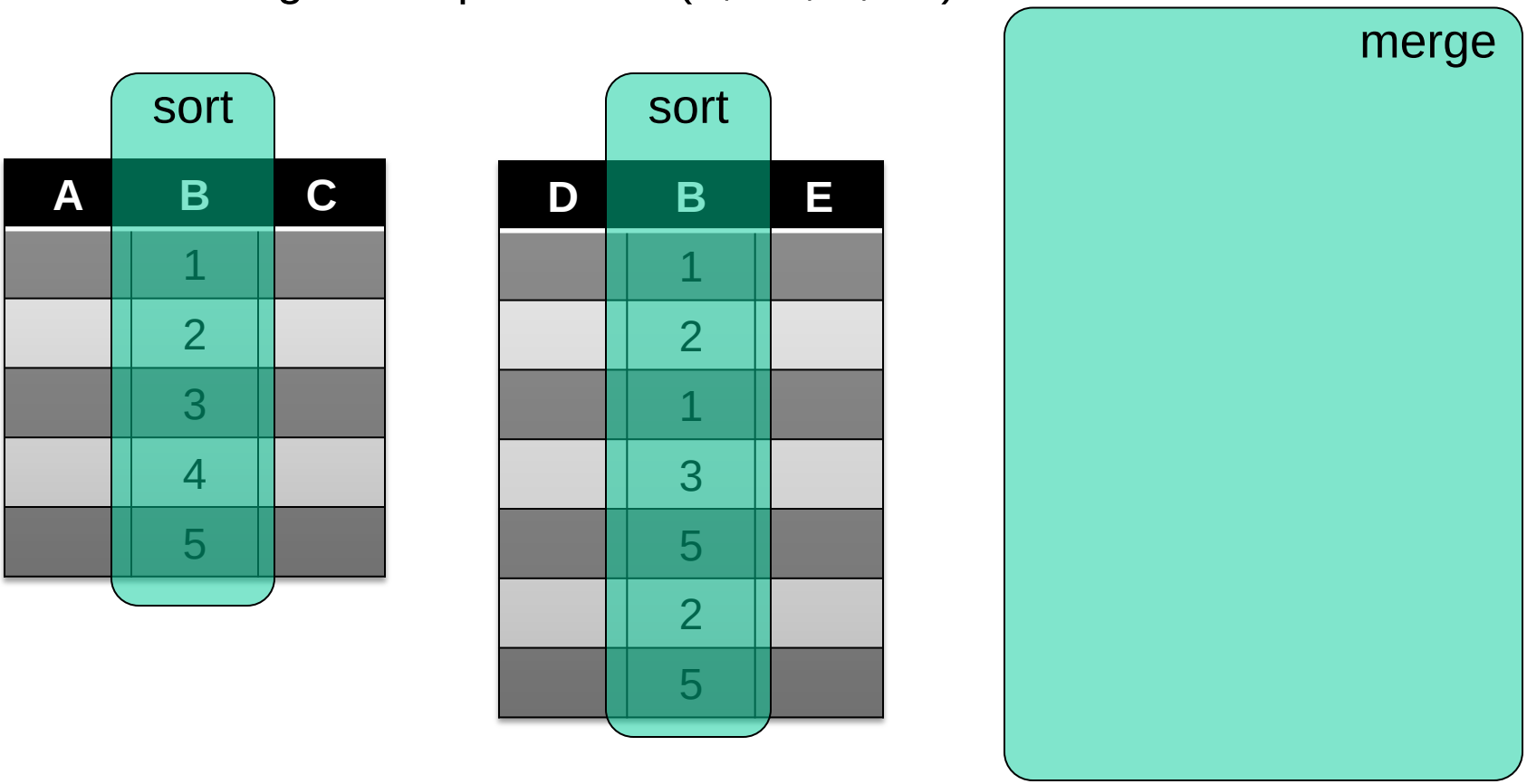
- Hash Join
  - Für Joins relativ großer Datenmengen
  - Alle Datensätze werden auf einmal geliefert



Optimizer – Join Methoden
Sort Merge Join

robotron®

- Sort Merge Joins
  - Für ungleich-Operatoren (<, <=, >, >=)



Optimizer – UNION vs. IN vs. OR

robotron®

- Der Optimizer kann SQLs umbauen
  - `select … from TAB where A=1 or A=2 or A=3`
  - `select … from TAB where A in (1, 2, 3)`
  -
```
select … from TAB where A=1
union all
select … from TAB where A=2
union all
select … from TAB where A=3
```

Agenda

robotron®

Optimizer	• Aufgaben • Möglichkeiten • Randbedingungen
Zugriffspfade	• Welche gibt es • Wie funktionieren diese
Ausführungs-pläne	• Aufbau • Interpretation • Besondere Möglichkeiten
Gut zu wissen	• Indexe • Statistiken • Cursor Sharing

Ausführungspläne – was sind das?

robotron®

```
-------------------------------------------------------------------------------------------------
| Id  | Operation                  | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |                     |       |       |   574 (100)|          |
|   1 |  HASH GROUP BY             |                     |     5 |   255 |   574   (10)| 00:00:07 |
|*  2 |   HASH JOIN                |                     |    72 |  3672 |   573   (10)| 00:00:07 |
|   3 |    VIEW                    | VW_GBC_5            |    72 |  2160 |   570   (10)| 00:00:07 |
|   4 |     HASH GROUP BY          |                     |    72 |   648 |   570   (10)| 00:00:07 |
|   5 |      PARTITION RANGE ALL   |                     |   918K|  8075K|   530    (3)| 00:00:07 |
|   6 |       TABLE ACCESS FULL    | SALES               |   918K|  8075K|   530    (3)| 00:00:07 |
|   7 |    VIEW                    | index$_join$_002    |    72 |  1512 |     3   (34)| 00:00:01 |
|*  8 |     HASH JOIN              |                     |       |       |             |          |
|   9 |      INDEX FAST FULL SCAN  | PR                  |       |       |             |          |
|  10 |      INDEX FAST FULL SCAN  | PR                  |       |       |             |          |
-------------------------------------

Predicate Information (identified by
-------------------------------------

   2 - access("P"."PROD_ID"="ITEM_1")
   8 - access(ROWID=ROWID)
```

OPERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			574
HASH		GROUP BY	574
HASH JOIN			573
Zugriffsprädikate			
P.PROD_ID=ITEM_1			
VIEW	VW_GBC_5		570
HASH		GROUP BY	570
PARTITION RANGE		ALL	530
TABLE ACCESS	SALES	FULL	530
VIEW	index$_join$_002		3
HASH JOIN			
Zugriffsprädikate			
ROWID=ROWID			
INDEX	PRODUCTS_PK	FAST FU...	1
INDEX	PRODUCTS_PROD...	FAST FU...	1

Ausführungspläne – Wo bekomme ich die her?

robotron®

- EXPLAIN PLAN FOR
  - Keine Binds, evtl. andere Umgebung → geratener Plan
- DBMS_XPLAN
  - Plan aus dem Cache
  - `.DISPLAY_CURSOR` → Plan des letzten Statements
  - `.DISPLAY_CURSOR(sqlid, child#)` → ein beliebiger Plan
- SQL*Plus
  - `set autotrace traceonly` → benutzt im Hintergrund `EXPLAIN PLAN FOR`
- SQL Trace / tkprof
  - Tracefile enthält den ermittelten Plan sofern Hard Parse oder `dbms_monitor.[…]_trace_enable(… plan_stat=>'ALL_EXECUTIONS')`
  - **Achtung!** „explain=user/pwd“ macht nur EXPLAIN PLAN FOR

Ausführungspläne – Wahrheit und Dichtung

robotron®

- Angaben in Ausführungsplan zu
  - Anzahl Zeilen
  - Anzahl Bytes
  - Ausführungszeit

  Entsprechen den Schätzungen
- Tatsächliche Werte können stark abweichen
- Werden ermittelt durch erweiterte Statistiken
  - `alter [session | system] set statistics_level=all;`
  - `select /*+ GATHER_PLAN_STATISTICS */ …`
- Werden ausgewertet mit
  - `tkprof`
  - `dbms_xplan.display_cursor`

Ausführungspläne – Wahrheit und Dichtung - tkprof

robotron®

```
SQL> exec dbms_monitor.session_trace_enable(waits=>true, binds=>true,
                                            plan_stat=>'ALL_EXECUTIONS')

PL/SQL-Prozedur erfolgreich abgeschlossen.

SQL> SELECT prod_category, AVG(amount_sold)
  2  FROM sales s, products p
  3  WHERE p.prod_id = s.prod_id
  4  GROUP BY prod_category;

Abgelaufen: 00:00:01.95

SQL> exec dbms_monitor.session_trace_disable;

PL/SQL-Prozedur erfolgreich abgeschlossen.

tkprof marco_ora_12330.trc marco_ora_12330.trc.tkp sys=no
```

Ausführungspläne – Wahrheit und Dichtung - tkprof

robotron®

```
Rows (1st) Row Source Operation
---------- ---------------------------------------------------
         5 HASH GROUP BY (cr=1641 pr=1095 pw=1095 time=1830994 us cost=8
                          size=150 card=5)
    918843  HASH JOIN    (cr=1641 pr=1095 pw=1095 time=3068320 us cost=7
                          size=300 card=10)
    918843   PARTITION RANGE ALL PARTITION: 1 28
                         (cr=1635 pr=0 pw=0 time=5077500 us cost=4
                          size=90 card=10)
    918843    TABLE ACCESS FULL SALES PARTITION: 1 28          Tatsächliche Menge
                         (cr=1635 pr=0 pw=0 time=1980264 us cost=4
                          size=90 card=10)                          Schätzung
        72   VIEW  index$_join$_002
                         (cr=6 pr=0 pw=0 time=4386 us cost=3  size=1512
                          card=72)
        72    HASH JOIN  (cr=6 pr=0 pw=0 time=4094 us)
        72     INDEX FAST FULL SCAN PRODUCTS_PK
                         (cr=3 pr=0 pw=0 time=187 us cost=1 size=1512
                          card=72)
        72     INDEX FAST FULL SCAN PRODUCTS_PROD_CAT_IX
                         (cr=3 pr=0 pw=0 time=454 us  cost=1 size=1512
                          card=72)
```

Ausführungspläne – Wahrheit und Dichtung - DBMS_XPLAN

robotron®

```
SQL> alter session set statistics_level=all;

Session wurde geändert.

SQL> SELECT prod_category, AVG(amount_sold)
  2  FROM sales s, products p
  3  WHERE p.prod_id = s.prod_id
  4  GROUP BY prod_category;

Abgelaufen: 00:00:15.22

SQL> select * from table(dbms_xplan.display_cursor(format=>'ALLSTATS LAST'));
```

Ausführungspläne – Wahrheit und Dichtung - DBMS_XPLAN

robotron®

```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                 | Starts | E-Rows | A-Rows |   A-Time   |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                      |      1 |        |      5 |00:00:15.11 |
|   1 |  HASH GROUP BY               |                      |      1 |      5 |      5 |00:00:15.11 |
|*  2 |   HASH JOIN                  |                      |      1 |     10 |    918K|00:00:13.21 |
|   3 |    PARTITION RANGE ALL       |                      |      1 |     10 |    918K|00:00:05.21 |
|   4 |     TABLE ACCESS FULL        | SALES                |     28 |     10 |    918K|00:00:02.18 |
|   5 |    VIEW                      | index$_join$_002     |      1 |     72 |     72 |00:00:00.01 |
|*  6 |     HASH JOIN                |                      |      1 |        |     72 |00:00:00.01 |
|   7 |      INDEX FAST FULL SCAN    | PRODUCTS_PK          |      1 |     72 |     72 |00:00:00.01 |
|   8 |      INDEX FAST FULL SCAN    | PRODUCTS_PROD_CAT_IX |      1 |     72 |     72 |00:00:00.01 |
-------------------------------------------------------------------------------------------------------

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

   2 - access("P"."PROD_ID"="S"."PROD_ID")
   6 - access(ROWID=ROWID)
```

Geschätzte Menge (estimated rows)

Tatsächliche Menge (actual rows)

Agenda

robotron®

Optimizer	• Aufgaben • Möglichkeiten • Randbedingungen
Zugriffspfade	• Welche gibt es • Wie funktionieren diese
Ausführungs-pläne	• Aufbau • Interpretation • Besondere Möglichkeiten
Gut zu wissen	• Indexe • Statistiken • Cursor Sharing

Ausführungspläne – was geht besser? Indexe und NULL

robotron®

```
-------------------------------------------------------------
| Id | Operation         | Name | Rows | Bytes | Cost (%CPU)|
-------------------------------------------------------------
|  0 | SELECT STATEMENT  |      | 1    | 40    | 3 (0)      |
|* 1 | TABLE ACCESS FULL | EMP  | 1    | 40    | 3 (0)      |
-------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("EMP"."MGR") IS NULL)
```

```
create index IX_MGR
on EMP (MGR, 0);
```

Ausführungspläne – was geht besser? Indexe und NULL

robotron®

```
-------------------------------------------------------------------------
| Id | Operation                    | Name    | Rows | Bytes | Cost |
-------------------------------------------------------------------------
| 0  | SELECT STATEMENT             |         | 1    | 40    | 2 (0)|
| 1  | TABLE ACCESS BY INDEX ROWID| EMP     | 1    | 40    | 2 (0)|
|* 2 |  INDEX RANGE SCAN            | IX_MGR | 1    |       | 1 (0)|
-------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
2 - access("MGR" IS NULL)
```

Ausführungspläne – was geht besser? GROSS/klein Schreibung

robotron®

```
-----------------------------------------------------------------
| Id  | Operation                          | Name               |
-----------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                    |
|   1 |  NESTED LOOPS                      |                    |
|   2 |   NESTED LOOPS                     |                    |
|   3 |    NESTED LOOPS                    |                    |
|*  4 |     TABLE ACCESS FULL              | DEPARTMENTS        |
|   5 |     TABLE ACCESS BY INDEX ROWID    | LOCATIONS          |
|*  6 |      INDEX UNIQUE SCAN             | LOC_ID_PK          |
|*  7 |    INDEX RANGE SCAN                | EMP_DE             |
|   8 |   TABLE ACCESS BY INDEX ROWID      | EMPLOY             |
-----------------------------------------------------------------

Predicate Information (identified by operation
-----------------------------------------------

   4 - filter(UPPER("D"."DEPARTMENT_NAME")='SALES')
   6 - access("D"."LOCATION_ID"="L"."LOCATION_ID")
   7 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
```

```
create index
mm_dep_name_upper
on departments (
upper(department_name)
);
```

Ausführungspläne – was geht besser? GROSS/klein Schreibung

robotron®

```
-----------------------------------------------------------------
| Id  | Operation                          | Name               |
-----------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                    |
|   1 |  NESTED LOOPS                      |                    |
|   2 |   NESTED LOOPS                     |                    |
|   3 |    NESTED LOOPS                    |                    |
|   4 |     TABLE ACCESS BY INDEX ROWID    | DEPARTMENTS        |
|*  5 |      INDEX RANGE SCAN              | MM_DEP_NAME_UPPER  |
|   6 |     TABLE ACCESS BY INDEX ROWID    | LOCATIONS          |
|*  7 |      INDEX UNIQUE SCAN             | LOC_ID_PK          |
|*  8 |    INDEX RANGE SCAN                | EMP_DEPARTMENT_IX  |
|   9 |   TABLE ACCESS BY INDEX ROWID      | EMPLOYEES          |
-----------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   5 - access("D"."SYS_NC00005$"='SALES')
   7 - access("D"."LOCATION_ID"="L"."LOCATION_ID")
   8 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
```

Ausführungspläne – was geht besser?

robotron®

```
… where hire_date >
  '01.01.98'
```

```
create index
mm_emp_hire_date on
employees (hire_date);
```

```
-------------------------------------------------------------------
| Id  | Operation                    | Name                       |
-------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                            |
|*  1 |  HASH JOIN                   |                            |
|   2 |   MERGE JOIN                 |                            |
|   3 |    TABLE ACCESS BY INDEX ROWID|                            |
|   4 |     INDEX FULL SCAN          |                            |
|*  5 |    SORT JOIN                 |                            |
|   6 |     VIEW                     | index$_join$_003           |
|*  7 |      HASH JOIN               |                            |
|   8 |       INDEX FAST FULL SCAN   | LOC_CITY_IX                |
|   9 |       INDEX FAST FULL SCAN   | LOC_ID_PK                  |
|* 10 |   TABLE ACCESS FULL          | EMPLOYEES                  |
-------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
   5 - access("D"."LOCATION_ID"="L"."LOCATION_ID")
       filter("D"."LOCATION_ID"="L"."LOCATION_ID")
   7 - access(ROWID=ROWID)
  10 - filter("E"."HIRE_DATE">'01.01.98')
```

Ausführungspläne – was geht besser?

robotron®

… where hire_date > to_date('01.01.98') !!!!

```
-----------------------------------------------------------------
| Id  | Operation                    | Name                     |
-----------------------------------------------------------------
|   0 | SELECT STATEMENT             |                          |
|*  1 |  HASH JOIN                   |                          |
|   2 |   MERGE JOIN                 |                          |
|   3 |    TABLE ACCESS BY INDEX ROWID| DEPARTMENTS              |
|   4 |     INDEX FULL SCAN          | DEPT_LOCATION_IX         |
|*  5 |    SORT JOIN                 |                          |
|   6 |     VIEW                     | index$_join$_003         |
|*  7 |      HASH JOIN               |                          |
|   8 |       INDEX FAST FULL SCAN   | LOC_CITY_IX              |
|   9 |       INDEX FAST FULL SCAN   | LOC_ID_PK                |
|* 10 |   TABLE ACCESS FULL          | EMPLOYEES                |
-----------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
   5 - access("D"."LOCATION_ID"="L"."LOCATION_ID")
       filter("D"."LOCATION_ID"="L"."LOCATION_ID")
   7 - access(ROWID=ROWID)
  10 - filter("E"."HIRE_DATE">'01.01.98')
```

Ausführungspläne – was geht besser?

robotron®

```
-----------------------------------------------------------
| Id  | Operation                     | Name              |
-----------------------------------------------------------
|   0 | SELECT STATEMENT              |                   |
|*  1 |  HASH JOIN                    |                   |
|   2 |   MERGE JOIN                  |                   |
|   3 |    TABLE ACCESS BY INDEX ROWID| DEPARTMENTS       |
|   4 |     INDEX FULL SCAN           | DEPT_LOCATION_IX  |
|*  5 |    SORT JOIN                  |                   |
|   6 |     VIEW                      | index$_join$_003  |
|*  7 |      HASH JOIN                |                   |
|   8 |       INDEX FAST FULL SCAN    | LOC_CITY_IX       |
|   9 |       INDEX FAST FULL SCAN    | LOC_ID_PK         |
|  10 |   TABLE ACCESS BY INDEX ROWID | EMPLOYEES         |
|* 11 |    INDEX RANGE SCAN           | MM_EMP_HIRE_DATE  |
-----------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
   5 - access("D"."LOCATION_ID"="L"."LOCATION_ID")
       filter("D"."LOCATION_ID"="L"."LOCATION_ID")
   7 - access(ROWID=ROWID)
  11 - access("E"."HIRE_DATE">TO_DATE('01.01.98'))
```

Ausführungspläne – was geht besser? Spaltenzusammenhänge

robotron®

```
SQL> select * from emp where job='SALESMAN' and deptno=30;

-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     1 |    40 |     3   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| EMP  |     1 |    40 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------

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

   1 - filter("JOB"='SALESMAN' AND "DEPTNO"=30)
```

```
SQL> begin
  2  dbms_stats.gather_table_stats(
  3    user,
  4    'EMP',
  5    method_opt=>'for columns (job,deptno)');
  6  end;
  7  /
```

Ausführungspläne – was geht besser? Spaltenzusammenhänge

robotron®

```
SQL> select * from emp where job='SALESMAN' and deptno=30;

-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |     4 |   160 |     3   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| EMP  |     4 |   160 |     3   (0)| 00:00:01 |
-----------------------------------------------------------------------------

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

   1 - filter("JOB"='SALESMAN' AND "DEPTNO"=30)
```

Ausführungspläne
Adaptive Cursor Sharing – Lauf 1

robotron®

```
SQL> exec :id := 1;
SQL> select /*TEST*/ * from viel where nummer = :id;

19 Zeilen ausgewählt.

SQL> select sql_id ,child_number,IS_BIND_AWARE, IS_BIND_SENSITIVE,
  2> executions, buffer_gets from v$sql where sql_id= :sql_id;

SQL_ID        CHILD_NUMBER I I   EXECUTIONS   BUFFER_GETS
------------- ------------ - - ------------ ------------
gnacnx2xb9pxj             0 N Y            1           13


-----------------------------------------------------------------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost  |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |         |       |       |     8 |
|   1 |  TABLE ACCESS BY INDEX ROWID| VIEL    |    19 | 38076 |     8 |
|*  2 |   INDEX RANGE SCAN           | IX_VIEL |    19 |       |     1 |
-----------------------------------------------------------------------------
```

Ausführungspläne
Adaptive Cursor Sharing – Lauf 2

robotron®

```
SQL> exec :id := 10;
SQL> select /*TEST*/ * from viel where nummer = :id;

980 Zeilen ausgewählt.

SQL> select sql_id ,child_number,IS_BIND_AWARE, IS_BIND_SENSITIVE,
  2> executions, buffer_gets from v$sql where sql_id= :sql_id;

SQL_ID        CHILD_NUMBER I I   EXECUTIONS   BUFFER_GETS
------------- ------------ - - ------------ ------------
gnacnx2xb9pxj             0 N Y            2          475
```

Cursor bleibt unverändert → Ausreißer und Sondereffekte ausschließen

Ausführungspläne
Adaptive Cursor Sharing – Lauf 3

robotron®

```
SQL> exec :id := 10;
SQL> select /*TEST*/ * from viel where nummer = :id;

980 Zeilen ausgewählt.

SQL> select sql_id ,child_number,IS_BIND_AWARE, IS_BIND_SENSITIVE,
  2> executions, buffer_gets from v$sql where sql_id= :sql_id;

SQL_ID        CHILD_NUMBER I I   EXECUTIONS  BUFFER_GETS
------------- ------------ - - ------------ ------------
gnacnx2xb9pxj             0 N Y            2          475
gnacnx2xb9pxj             1 Y Y            1          403
```

Neuer Child Cursor als „is_bind_aware“

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |       |       |    96 (100)|          |
|*  1 |  TABLE ACCESS FULL| VIEL |   980 |  1917K|    96   (0)| 00:00:02 |
-------------------------------------------------------------------------------
```

Fragen & Antworten

robotron®