



SQL - Optimierung - nur etwas für Entwickler?

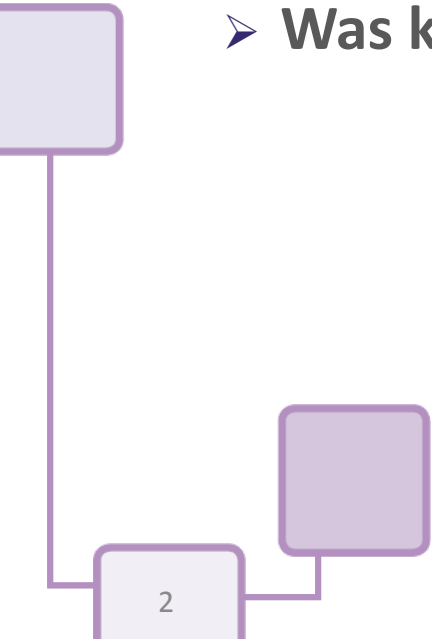
Johannes Ahrends
Geschäftsführer
CarajanDB GmbH



ORACLE
ACE



- **Vorstellung CarajanDB**
- **Ein Beispiel aus der Praxis**
 - ... und wer ist schuld?
- **Index oder nicht Index – das ist doch keine Frage, oder?**
- **Was kann der DBA tun?**
- **Was kann der Entwickler tun?**



- **Experten mit über 30 Jahren Oracle Erfahrung**
- **Spezialisten für**
 - Backup & Recovery
 - Hochverfügbarkeit
 - Healthchecks
 - Performance Optimierung
 - Einsatz von Oracle Standard Edition
 - Oracle in virtuellen Umgebungen und in der Cloud
 - Oracle Migrationen (HW, Unicode, Konsolidierung, Standard Edition)
 - Monitoring (Grid / Cloud Control, HLMM, Foglight, Spotlight)
- **Schulung und Workshops (Oracle, Toad)**





„DIE SPEZIALISTEN FÜR ORACLE DATENBANKEN, VIRTUALISIERUNG UND CLOUD COMPUTING“

TOAD 11.6 UPGRADE SCHULUNG

Dauer : 1 Tag inklusive Übungen

Trennung nach Datenbank Administration oder Entwicklung

THEMEN ALLGEMEIN

- Installation und Anpassung an Ihre Bedürfnisse
- Toad Editor – Tipps und Tricks
- Schema Browser - Welche Funktionen benötigen Sie für die tägliche Arbeit
- Query Builder - Erstellung von komplexen Abfragen
- Toad Data Modeler - Reverse Engineering von Schemata

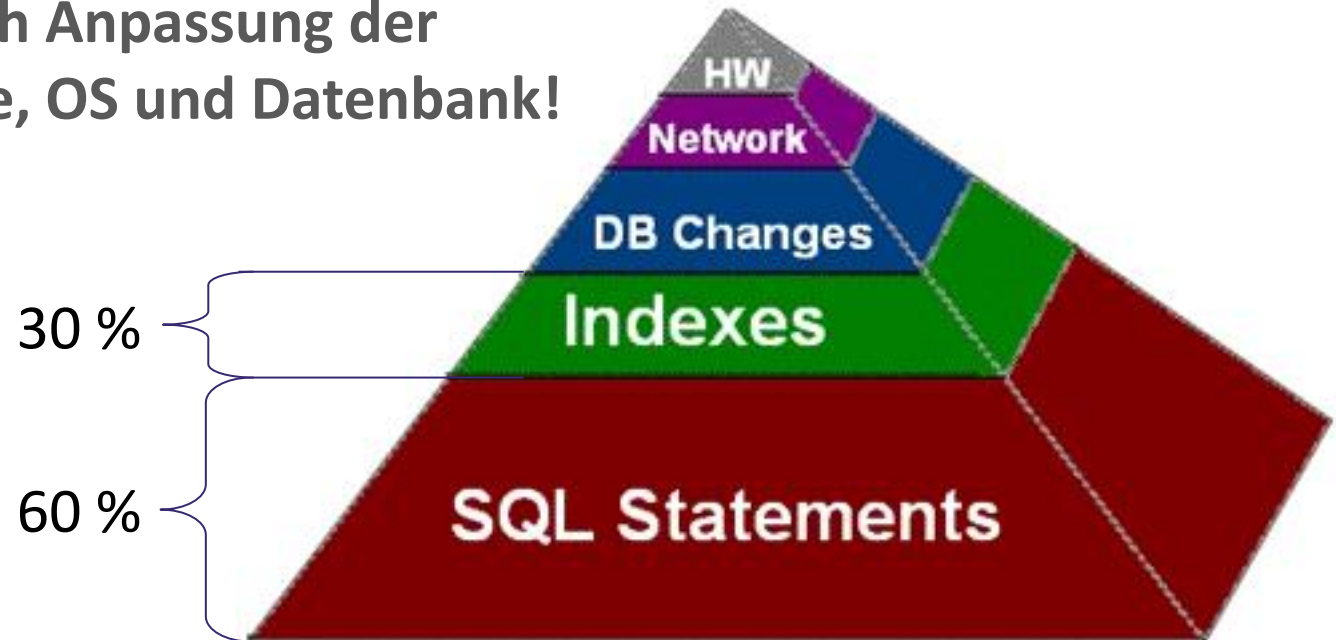
THEMEN DATENBANK ADMINISTRATION

- Database Browser oder Schemabrowser – was ist besser?
- Health Check – welche Abfragen sind für Sie wichtig?
- Diagnose – schnell ein Problem identifizieren und lösen (Toad und Spotlight)

THEMEN ENTWICKLUNG

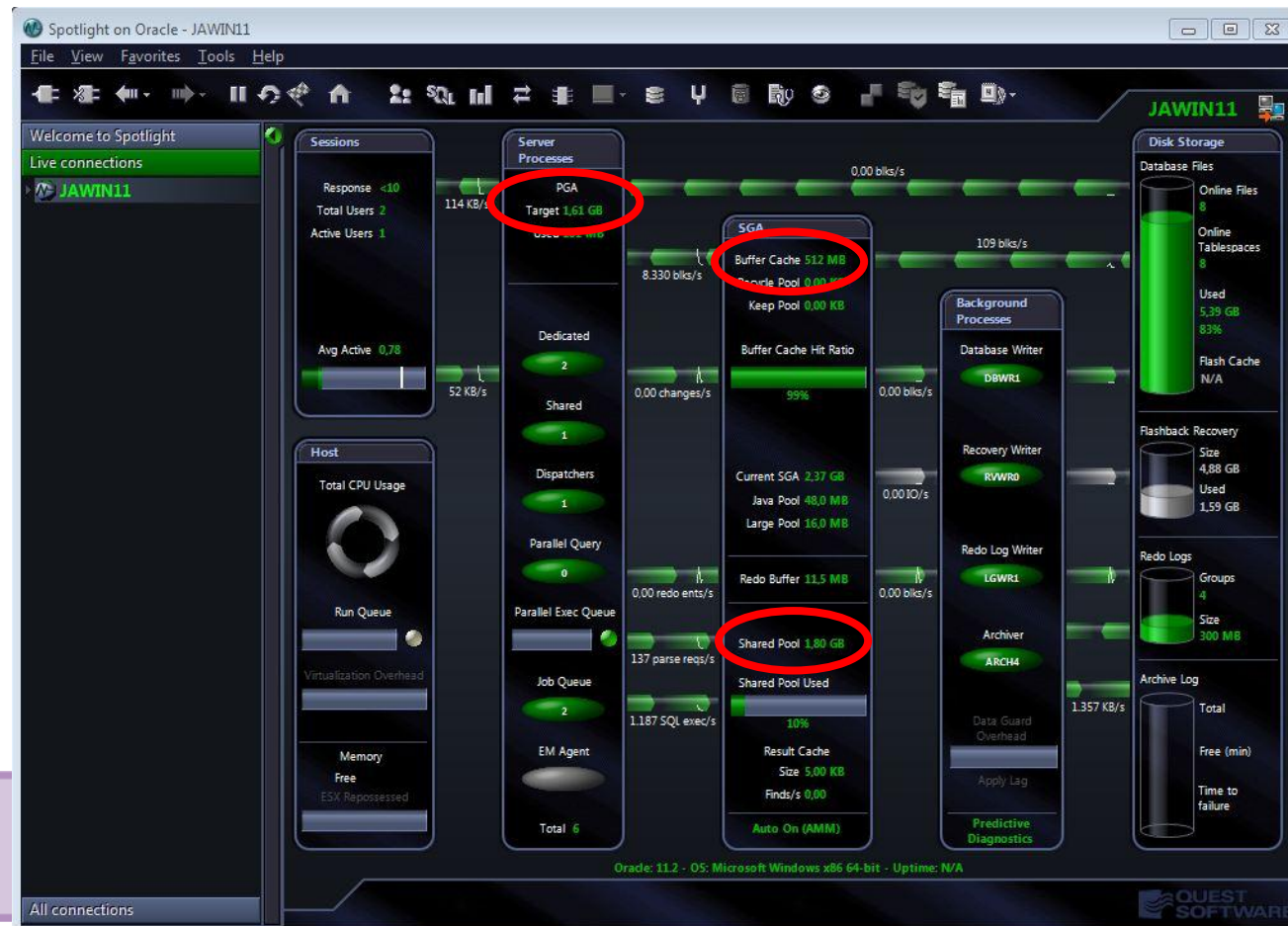
- PL/SQL Programmierung leicht gemacht (Code Templates, Auto Replace)
- Debugging
- Formattierung

- Forrester Research: Root of Database Performance Impact
- 90% Tuningpotential durch Optimierung einzelner Befehle
- 10% durch Anpassung der Hardware, OS und Datenbank!



Ein Beispiel aus der Praxis

➤ Was fällt auf?



Init.ora – Oracle 11g

```
jawin11.__db_cache_size=536870912
jawin11.__java_pool_size=50331648
jawin11.__large_pool_size=16777216
jawin11.__pga_aggregate_target=1728053248
jawin11.__sga_target=2566914048
jawin11.__shared_io_pool_size=0
jawin11.__shared_pool_size=1929379840
jawin11.__streams_pool_size=0
*.memory_target=4G
```

- **Mindestens setzen:**
 - `db_cache_size`
 - `shared_pool_size`
- **Shared Pool kann im laufenden Betrieb kaum freigegeben werden**
 - `ALTER SYSTEM FLUSH SHARED_POOL`

Spotlight: Oracle Parameter

The screenshot shows the 'Spotlight on Oracle - JAWIN11' application window. The 'Configuration & Memory' tab is active, and the 'Oracle Parameters' sub-tab is selected. A list of parameters is displayed, with 'db_cache_size' highlighted. An 'Edit Oracle Parameter' dialog box is open, showing the following details:

- Parameter name: db_cache_size
- Parameter value: 1500M
- Description: Size of DEFAULT buffer pool for standard block size buffers
- Scope: Both

The dialog has 'Update' and 'Cancel' buttons. The background table lists various Oracle parameters with their current values and descriptions.

Parameter Name	Value	Default	Scope	Description
db_4k_cache_size				
db_8k_cache_size				
db_16k_cache_size				
db_32k_cache_size				
db_block_buffers				
db_block_checking				
db_block_checksum				
db_block_size				
db_cache_advice	ON	YES	IMMEDIATE	Buffer cache sizing advice
db_cache_size	1577058304	YES	IMMEDIATE	Size of DEFAULT buffer pool for standard block size buffers
db_create_file_dest		YES	IMMEDIATE	default database location
db_create_online_log_dest_1		YES	IMMEDIATE	online log/controlfile destination
db_create_online_log_dest_2		YES	IMMEDIATE	online log/controlfile destination
db_create_online_log_dest_3		YES	IMMEDIATE	online log/controlfile destination
db_create_online_log_dest_4		YES	IMMEDIATE	online log/controlfile destination
db_create_online_log_dest_5		YES	IMMEDIATE	online log/controlfile destination
db_domain		NO	NO	directory part of global name
db_file_multiblock_read_count	101	YES	IMMEDIATE	db block to be read in one I/O operation
db_file_name_convert		YES	NO	datafile name conversion
db_files	200	YES	NO	max allowable # of datafiles
db_flash_cache_file		YES	NO	flash cache file for default

➤ Ist dieser Index sinnvoll?

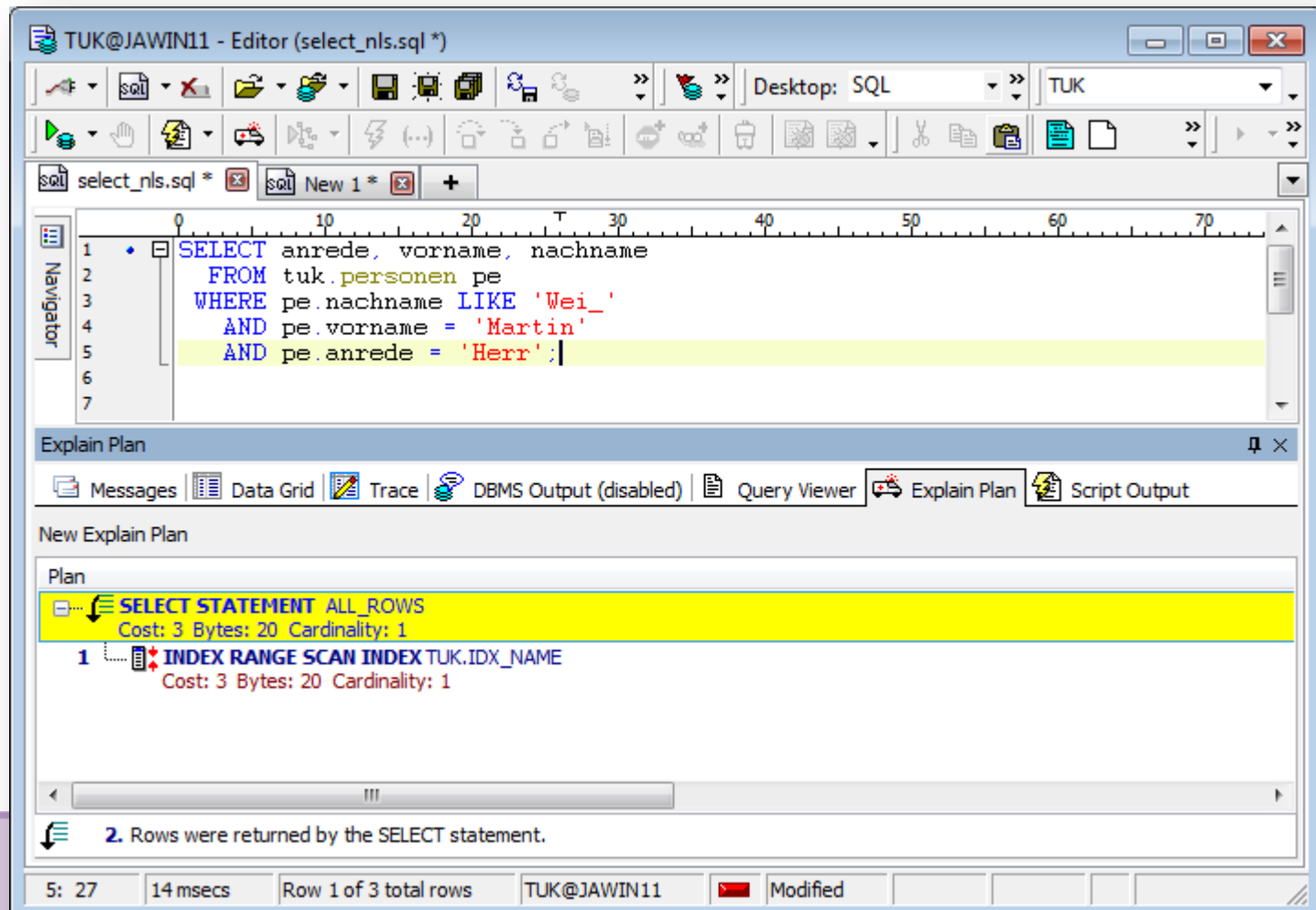
```
CREATE INDEX idx_name  
ON personen (anrede, vorname, nachname);
```

➤ Abfrage 1:

```
SELECT anrede, vorname, nachname  
FROM tuk.personen pe  
WHERE pe.nachname LIKE 'Wei_'  
AND pe.vorname = 'Martin'  
AND pe.anrede = 'Herr';
```

1_select_index.sql

Index Range Scan



The screenshot shows the Toad SQL Editor interface. The main window displays an SQL query in the editor:

```
1 SELECT anrede, vorname, nachname
2 FROM tuk.personen pe
3 WHERE pe.nachname LIKE 'Wei_'
4 AND pe.vorname = 'Martin'
5 AND pe.anrede = 'Herr';
```

Below the editor, the 'Explain Plan' window is open, showing the execution plan for the query. The plan is as follows:

```
Plan
1 SELECT STATEMENT ALL_ROWS
  Cost: 3 Bytes: 20 Cardinality: 1
  1 INDEX RANGE SCAN INDEX TUK.IDX_NAME
    Cost: 3 Bytes: 20 Cardinality: 1
```

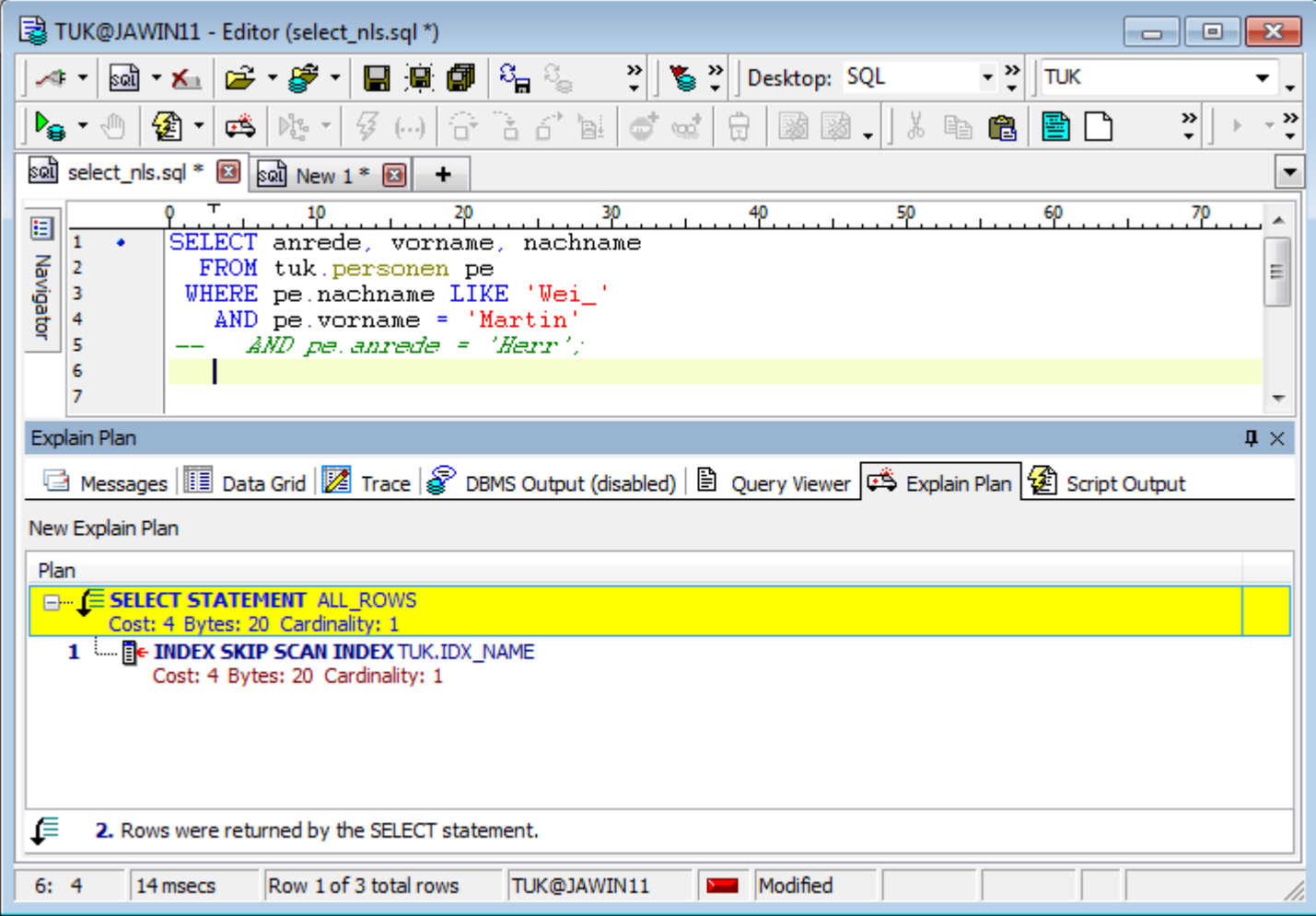
The status bar at the bottom indicates: 5: 27, 14 msec, Row 1 of 3 total rows, TUK@JAWIN11, Modified.



➤ Abfrage 2:

```
SELECT anrede, vorname, nachname  
FROM tuk.personen pe  
WHERE pe.nachname LIKE 'Wei_'  
      AND pe.vorname = 'Martin'  
--    AND pe.anrede = 'Herr';
```

Index Skip Scan



The screenshot shows the Toad SQL Editor interface. The main window displays an SQL query in the 'select_nls.sql' file:

```
1 SELECT anrede, vorname, nachname
2 FROM tuk.personen pe
3 WHERE pe.nachname LIKE 'Wei_'
4 AND pe.vorname = 'Martin'
5 -- AND pe.anrede = 'Herr';
6
7
```

Below the editor, the 'Explain Plan' tab is active, showing the following plan:

New Explain Plan

Plan
SELECT STATEMENT ALL_ROWS Cost: 4 Bytes: 20 Cardinality: 1
1 INDEX SKIP SCAN INDEX TUK.IDX_NAME Cost: 4 Bytes: 20 Cardinality: 1
2. Rows were returned by the SELECT statement.

The status bar at the bottom indicates: 6: 4 | 14 msec | Row 1 of 3 total rows | TUK@JAWIN11 | Modified



Index Benutzung oder nicht?

```
SQL> SELECT anrede, vorname, nachname
2      FROM tuk.personen pe
3      WHERE pe.nachname LIKE 'wei_'
4      AND pe.vorname LIKE 'Martin';
```

ANRED	VORNAME	NACHNAME
Herr	Martin	Weiß
Herr	Martin	Weis
Herr	Martin	Weiz

3 Zeilen ausgewählt.

2_abfrage_ls.bat

➤ Warum Full-Table-Scan?

Program	Machine	OSUser	SerialNum	Server	Status
SPOTLIGHT.EXE					
SQLPLUS.EXE					
TUK	CARAJANDB\CARA	CARAJANDB01\Joh	283	DEDICATED	INACTIVE
TOAD.EXE					
SYSTEM	CARAJANDB\CARA	CARAJANDB01\Joh	121	DEDICATED	ACTIVE
SYSTEM	CARAJANDB\CARA	CARAJANDB01\Joh	77	DEDICATED	INACTIVE
TUK	CARAJANDB\CARA	CARAJANDB01\Joh	423	DEDICATED	INACTIVE
TUK	CARAJANDB\CARA	CARAJANDB01\Joh	33	DEDICATED	INACTIVE
TUK	CARAJANDB\CARA	CARAJANDB01\Joh	125	DEDICATED	INACTIVE

Session	Process	IO	Waits	Current Statement	Open Cursors	Access	Locks	RBS Usage	Long Ops	Statistics
---------	---------	----	-------	-------------------	--------------	--------	-------	-----------	----------	------------

Cached Explain Plan

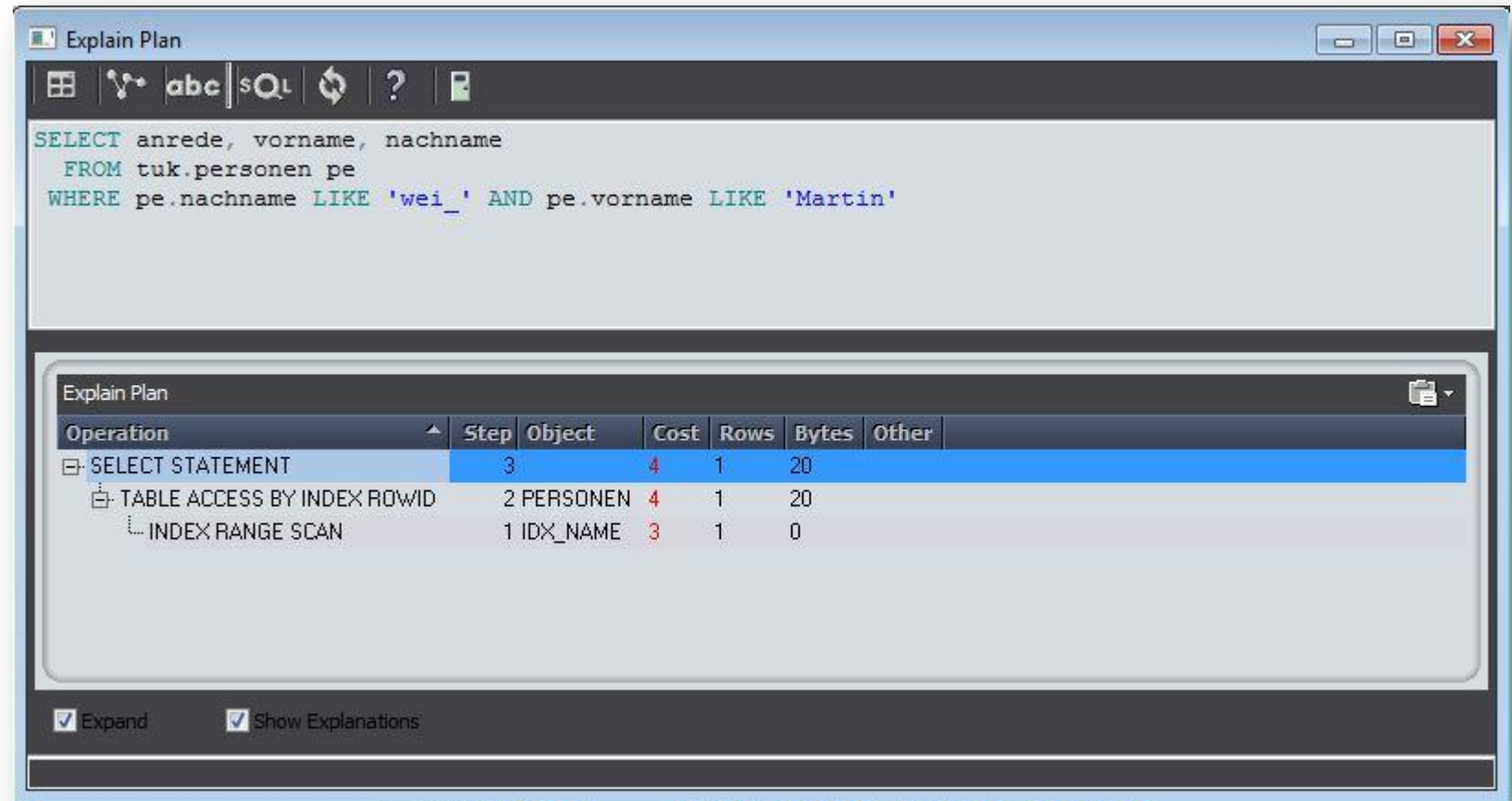
Plan

SELECT STATEMENT ALL_ROWS
Cost: 274

1 **TABLE ACCESS FULL TABLE TUK.PERSONEN**
Cost: 274 Bytes: 20 Cardinality: 1



... und Spotlight?



The screenshot shows the 'Explain Plan' window in Toad SQL Developer. The query being executed is:

```
SELECT anrede, vorname, nachname
FROM tuk.personen pe
WHERE pe.nachname LIKE 'wei_' AND pe.vorname LIKE 'Martin'
```

The execution plan is displayed in a table with the following columns: Operation, Step, Object, Cost, Rows, Bytes, and Other.

Operation	Step	Object	Cost	Rows	Bytes	Other
SELECT STATEMENT	3		4	1	20	
TABLE ACCESS BY INDEX ROWID	2	PERSONEN	4	1	20	
INDEX RANGE SCAN	1	IDX_NAME	3	1	0	

At the bottom of the window, there are two checked options: ☒ Expand and ☒ Show Explanations.



➤ Versuch einer Analyse:

```
SQL> SELECT anrede, vorname, nachname  
2      FROM tuk.personen pe  
3      WHERE pe.nachname LIKE 'wei_'  
4      AND pe.vorname LIKE 'Martin';
```

ANRED	VORNAME	NACHNAME
Herr	Martin	Weiß
Herr	Martin	Weis
Herr	Martin	Weiz

3 Zeilen ausgewählt.

- **ALTER SESSION SET nls_sort=binary_ci;**
 - Sortierung ist unabhängig von Groß- / Kleinschreibung aber abhängig von Akzenten
 - Alternativen:
 - binary_ai → Case und Akzent insensitiv
 - german_ai → Deutsche Sortierung, Case und Akzent insensitiv
 - ...
- **ALTER SESSION SET nls_comp=linguistic;**
 - Filter verwenden die gleiche Funktion wie NLS_SORT, d.h. in diesem Fall ist die WHERE-Clausel unabhängig von Groß- / Kleinschreibung und von Akzenten
 - Alternativen:
 - BINARY oder ANSI

- Entweder Linguistische Suche ausschalten
- ... oder Index auf Linguistische Suche

```
CREATE INDEX idx_name2  
  ON personen (NLSSORT (vorname, 'nls_sort=binary_ci'),  
              NLSSORT (nachname, 'nls_sort=binary_ci'));
```

- Aber jetzt sind z.B. keine Bitmapped Indizes auf den Spalten möglich

2c_nls_index.sql

➤ Initialisierungsparameter:

- `Cursor_sharing = FORCE | SIMILAR) * | EXACT`

➤ Wandelt Literale in Bindevariablen um

```
SELECT ...  
  FROM personen p, auftraege a  
 WHERE p.persid = a.persid  
AND p.vorname = 'Horst'  
   AND p.nachname = 'Ortmann'  
   AND a.aufstatus = 'G';
```

```
SELECT ...  
FROM personen p, auftraege a  
 WHERE      p.persid = a.persid  
           AND p.vorname = : "SYS_B_0"  
           AND p.nachname = : "SYS_B_1"  
           AND a.aufstatus = : "SYS_B_2"
```



- Was passiert, wenn ein Wert nicht gleich verteilt ist?
- Z.B. AUFTRAGSSTATUS
 - Geliefert hoffentlich über 90%
 - Storniert hoffentlich nur 1 – 2 %

```
SELECT ...  
  FROM personen p, auftraege a  
 WHERE p.persid = a.persid  
       AND a.aufstatus = 'G';
```

- Wenn l_aufstatus = 'G' ist, dann besser Full-Tables Scan
- Wenn l_aufstatus = 'S' ist, dann besser Index Range Scan

➤ Aber was ist mit Bindevariablen?

```
SELECT ...  
  FROM personen p, auftraege a  
 WHERE p.persid = a.persid  
       AND a.aufstatus = :l_aufstatus;
```

- Optimizer analysiert die Abfrage und erstellt u.U. einen anderen Ausführungsplan
- Beispiel von Christian Antognini, Autor des Buches „Troubleshooting Oracle Performance“
<http://www.antognini.ch/publications>

3_chris_bind1.bat

Index oder nicht?

```
SELECT anrede, vorname, nachname, geburtstag  
FROM tuk.personen pe  
WHERE pe.nachname like 'Weisse%'  
and pe.anrede = 'Herr';
```

Index Zugriff

The screenshot displays the Toad SQL Developer interface. At the top, a SQL query is entered in the main editor:

```
1 SELECT anrede, vorname, nachname, geburtstag
2 FROM tuk.personen pe
3 WHERE pe.nachname like 'Weisse%'
4 and pe.anrede = 'Herr';
```

Below the query editor, the 'Explain Plan' tab is active. It shows a hierarchical execution plan for the query:

- Plan**
 - 1 SELECT STATEMENT ALL_ROWS**
Cost: 410 Bytes: 756 Cardinality: 27
 - 2 TABLE ACCESS BY INDEX ROWID TABLE TUK.PERSONEN**
Cost: 410 Bytes: 756 Cardinality: 27
 - 1 INDEX RANGE SCAN INDEX TUK.IDX_NAME**
Cost: 379 Cardinality: 31
 - 3. Rows were returned by the SELECT statement.**

The plan indicates that the query uses an index range scan on the `TUK.IDX_NAME` index to access the `TUK.PERSONEN` table.



... oder doch nicht?

The screenshot shows the Toad SQL Developer interface. The top pane contains the following SQL query:

```
1 SELECT anrede, vorname, nachname, geburtstag
2 FROM tuk.personen pe
3 WHERE pe.nachname like 'Weisse%'
4 and pe.anrede = 'Herr';
```

The bottom pane displays the 'Explain Plan' for the query. The 'Plan' section shows the following steps:

- SELECT STATEMENT** ALL_ROWS
Cost: 274 Bytes: 756 Cardinality: 27
- 1 TABLE ACCESS FULL** TABLE TUK.PERSONEN
Cost: 274 Bytes: 756 Cardinality: 27

At the bottom, a summary line states: **2. Rows were returned by the SELECT statement.**

➤ **db_file_multiblock_read_count**

- Anzahl Blöcke die mit einem I/O beim table oder index scan gelesen werden
- Default 128 (ev. 64)
- Je höher der Wert umso wahrscheinlicher ein Full-Table-Scan

➤ **optimizer_index_cost_adj**

- Verhältnis zwischen Full-Table-Scan und Index Benutzung
- Default 100
- Je geringer der Wert umso wahrscheinlicher ein Index Zugriff
- Schlecht bei hohen Clustering Faktoren!

Index oder nicht?

```
SELECT p.persid,  
       p.vorname,  
       p.nachname,  
       m.email  
FROM personen p, mail m  
WHERE p.persid = m.persid  
      AND p.persid = :l_persid;
```

- Beide Tabellen persid als Primary Key!

5_select_mail.sql (TUK3)

Warum kein Index?

The screenshot shows the Toad SQL Editor interface. The main window displays a SQL query:

```
1 SELECT p.persid,  
2       p.vorname,  
3       p.nachname,  
4       m.email  
5 FROM personen p, mail m  
6 WHERE p.persid = m.persid  
7       AND p.persid = :l_persid;
```

Below the query editor, the 'Explain Plan' tab is active, showing the following execution plan:

```
Plan  
1 SELECT STATEMENT ALL_ROWS  
  Cost: 21 Bytes: 54 Cardinality: 1  
2 NESTED LOOPS  
  Cost: 21 Bytes: 54 Cardinality: 1  
3   TABLE ACCESS BY INDEX ROWID TABLE TUK3.PERSONEN  
    Cost: 2 Bytes: 20 Cardinality: 1  
4     INDEX UNIQUE SCAN INDEX (UNIQUE) TUK3.PK_PERSONEN  
      Cost: 1 Cardinality: 1  
5     TABLE ACCESS FULL TABLE TUK3.MAIL  
      Cost: 19 Bytes: 34 Cardinality: 1
```

At the bottom of the plan, it states: "5. Rows were returned by the SELECT statement."

The status bar at the bottom indicates: 5: 24 79 msec TUK3@JAWIN11 Modified



Datentyp Konvertierung

```
CREATE TABLE MAIL
(
  PERSID      VARCHAR2(10 BYTE) ,
  EMAIL        VARCHAR2(50 BYTE) ,
  BEMERKUNG    VARCHAR2(200 BYTE)
)
TABLESPACE USERS;
```

Einfacher Ausführungsplan!?

```
SELECT DISTINCT anrede, vorname, nachname, geburtstag
FROM tuk.kunden
WHERE nachname LIKE 'Wei_'
      AND vorname = 'Martin'
      AND anrede = 'Herr';
```

6_personen.sql
6_kunden.sql



Ein einfacher Ausführungsplan

The screenshot displays the Toad SQL Developer interface. At the top, a SQL query is entered in the editor:

```
1 • SELECT distinct anrede, vorname, nachname, geburtstag
2 FROM tuk.personen
3 WHERE nachname LIKE 'Wei_'
4 AND vorname = 'Martin'
5 AND anrede = 'Herr';
```

Below the editor, the 'Explain Plan' tab is active. It shows a hierarchical execution plan for the query:

- SELECT STATEMENT ALL_ROWS**
Cost: 5 Bytes: 28 Cardinality: 1
 - HASH UNIQUE**
Cost: 5 Bytes: 28 Cardinality: 1
 - TABLE ACCESS BY INDEX ROWID TABLE TUK.PERSONEN**
Cost: 4 Bytes: 28 Cardinality: 1
 - INDEX RANGE SCAN INDEX TUK.IDX_NAME**
Cost: 3 Cardinality: 1

... oder doch nicht?

1 SELECT distinct anrede, vorname, nachname, geburtstag
2 FROM tuk.kunden
3 WHERE nachname LIKE 'Wei_'
4 AND vorname = 'Martin'
5 AND anrede = 'Herr';

Explain Plan

Messages | Data Grid | Trace | DBMS Output (disabled) | Query Viewer | Explain Plan | Script Output

New Explain Plan

Plan

- 15 NESTED LOOPS
Cost: 4.917 Bytes: 142 Cardinality: 1
 - 13 HASH JOIN
Cost: 4.916 Bytes: 118 Cardinality: 1
 - 11 NESTED LOOPS
Cost: 2.650 Bytes: 95 Cardinality: 1
 - 9 NESTED LOOPS
Cost: 2.649 Bytes: 79 Cardinality: 1
 - 7 NESTED LOOPS
Cost: 2.648 Bytes: 69 Cardinality: 1
 - 4 HASH JOIN
Cost: 2.648 Bytes: 69 Cardinality: 1
 - 2 TABLE ACCESS BY INDEX ROWID TABLE TUK.
Cost: 4 Bytes: 25 Cardinality: 1
 - 1 INDEX RANGE SCAN INDEX TUK.IDX_NAME
Cost: 3 Cardinality: 1
 - 3 TABLE ACCESS FULL TABLE TUK.ADRESSEN
Cost: 2.642 Bytes: 15.664.088 Cardinality: 356.
 - 6 TABLE ACCESS BY INDEX ROWID TABLE TUK.TYPE
Cost: 1 Bytes: 10 Cardinality: 1
 - 5 INDEX UNIQUE SCAN INDEX (UNIQUE) TUK.PK
Cost: 0 Cardinality: 1
 - 8 INDEX UNIQUE SCAN INDEX (UNIQUE) TUK.PK_BUNDESLAENDI
Cost: 0 Cardinality: 1
 - 10 TABLE ACCESS BY INDEX ROWID TABLE TUK.BUNDESLAENDI
Cost: 1 Bytes: 16 Cardinality: 1
 - 12 TABLE ACCESS FULL TABLE TUK.ANREDE
Cost: 1 Bytes: 16 Cardinality: 1

33. Rows were returned by the SELECT statement.



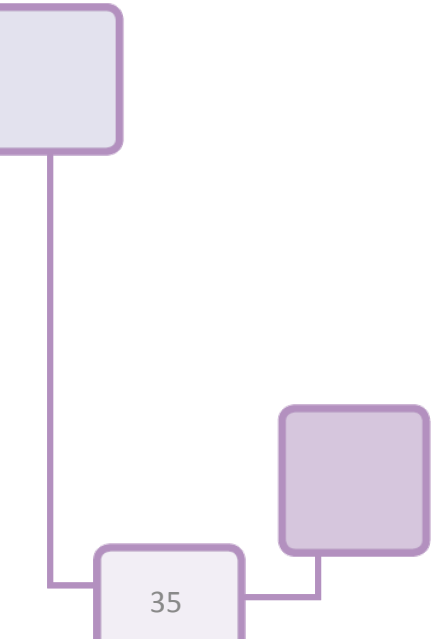
Wo bin ich stutzig geworden?

```
SELECT DISTINCT anrede, vorname, nachname, geburtstag  
FROM tuk.kunden  
WHERE nachname LIKE 'Wei_'  
AND vorname = 'Martin'  
AND anrede = 'Herr';
```



Verwendung von Views

- Views sind gut um komplexe Befehle zu „verstecken“
- Kaskadierende Views sollten vermieden werden



➤ Abhängige Spalten

```
SELECT pe.anrede, pe.vorname, pe.nachname, ad.plz,  
       ad.ort, ad.strasse  
  FROM adressen ad, personen pe  
 WHERE pe.persid = ad.persid  
       AND pe.nachname = :nachname  
       AND ad.plz = :plz  
       AND ad.ort = :ort;
```

7_extended_stat1.sql (TUK3)

Einfache Statistiken

The screenshot displays the Toad SQL Developer interface. At the top, a SQL query is entered in the editor:

```
1 SELECT pe.anrede, pe.vorname, pe.nachname, ad.plz, ad.ort, ad.strasse
2 FROM adressen ad, personen pe
3 WHERE pe.persid = ad.persid
4 AND pe.nachname = :nachname
5 AND ad.plz = :plz
6 AND ad.ort = :ort;
```

Below the editor, the 'Explain Plan' tab is active, showing the execution plan for the query. The plan is as follows:

- SELECT STATEMENT ALL_ROWS**
Cost: 5 Bytes: 140 Cardinality: 2
 - 6 NESTED LOOPS**
 - 4 NESTED LOOPS**
Cost: 5 Bytes: 140 Cardinality: 2
 - 2 TABLE ACCESS BY INDEX ROWID TABLE TUK3.ADRESSEN**
Cost: 3 Bytes: 76 Cardinality: 2
 - 1 INDEX RANGE SCAN INDEX TUK3.IDX_ORT**
Cost: 1 Cardinality: 2
 - 3 INDEX UNIQUE SCAN INDEX (UNIQUE) TUK3.PK_PERSONEN**
Cost: 0 Cardinality: 1
 - 5 TABLE ACCESS BY INDEX ROWID TABLE TUK3.PERSONEN**
Cost: 1 Bytes: 32 Cardinality: 1



Erweiterte Statistiken

The screenshot shows the Toad SQL Editor interface. The main window displays an SQL query in the 'extended_stat1.sql' file. The query is as follows:

```
1 SELECT pe.anrede, pe.vorname, pe.nachname, ad.plz, ad.ort, ad.strasse
2 FROM adressen ad, personen pe
3 WHERE pe.persid = ad.persid
4 AND pe.nachname = :nachname
5 AND ad.plz = :plz
6 AND ad.ort = :ort;
```

Below the query editor, the 'Explain Plan' tab is active. It shows the execution plan for the query. The plan consists of the following steps:

- SELECT STATEMENT ALL_ROWS**
Cost: 75 Bytes: 280 Cardinality: 4
- 4 HASH JOIN**
Cost: 75 Bytes: 280 Cardinality: 4
 - 2 TABLE ACCESS BY INDEX ROWID TABLE TUK3.PERSONEN**
Cost: 6 Bytes: 128 Cardinality: 4
 - 1 INDEX RANGE SCAN INDEX TUK3.IDX_NAME**
Cost: 2 Cardinality: 4
 - 3 TABLE ACCESS FULL TABLE TUK3.ADRESSEN**
Cost: 68 Bytes: 7.828 Cardinality: 206
- 5. Rows were returned by the SELECT statement.**



➤ Erstellen von erweiterten Statistiken

```
SELECT dbms_stats.create_extended_stats (
    OWNNAME      => 'TUK3',
    TABNAME      => 'ADRESSEN',
    EXTENSION    => '(PLZ,ORT)')
FROM DUAL;

BEGIN
    dbms_stats.gather_table_stats (
        OWNNAME      => 'TUK3',
        TABNAME      => 'ADRESSEN'
        ,ESTIMATE_PERCENT => 100
        ,METHOD_OPT    => 'FOR ALL COLUMNS SIZE SKEWONLY'
        ,DEGREE        => NULL
        ,CASCADE        => TRUE
        ,NO_INVALIDATE => FALSE);
END;
```



➤ Löschen von erweiterten Statistiken

```
BEGIN
  dbms_stats.drop_extended_stats (
    OWNNAME      => 'TUK2',
    TABNAME      => 'ADRESSEN',
    EXTENSION    => '(PLZ,ORT)');
END;
```

Was kann / sollte der DBA tun?

- **Automatic Memory Management → Fluch und Segen**
 - Mindestgrößen definieren!
- **Serverparameter:**
 - `cursor_sharing` → immer wieder gern genommen aber besser, der Entwickler benutzt Bindevariablen
 - Bind Peeking beachten
 - `db_file_multiblock_read_count` → kleiner Wert
 - `optimizer_index_cost_adj` → nur Ausnahmsweise verändern
 - Besser: Systemstatistiken regelmäßig aktualisieren
- **Statistiken**
 - Histogramme auf kritische Spalten
 - Ev. Extended Statistiken für abhängige Spalten



Was kann / sollte der Entwickler tun?



- **Benutzung von Bindevariablen**
 - Dadurch kein `cursor_sharing = FORCE` notwendig
 - Literale verwenden, wo es sinnvoll ist
- **Änderungen an der Umgebung protokollieren und dem DBA mitteilen**
 - Z.B. `NLS_SORT` bzw. `NLS_COMP`
- **Verwendung von Funktionen vermeiden**
 - Besser „BETWEEN“ als „TO_CHAR“ bei Datumssuche
- **Nicht zu viel „verstecken“ spielen**
 - Keine View auf View auf View
- **Datentypen beachten**
 - Keine unterschiedlichen Datentypen mit gleichem Namen



- 20. bis 22. November: DOAG Konferenz in Nürnberg



CARAJAN DB

Fragen?

Johannes Ahrends
www.carajandb.com

Johannes.ahrends@carajandb.com



ORACLE
ACE

