click-click
IT Solutions

# Speed up your Query Strategien zur Optimierung von SQL-Queries

Juni 2012

Ulrike Brenner

click-click
IT Solutions

# Ulrike Brenner

- Seit 1999 Oracle Entwicklerin
- Oracle DB 8-11gR2
- SQL
  PL/SQL
  Oracle Forms/Reports
  APEX
- Entwicklerin
  Teamleiterin
  Kundenbetreuerin

# Max Gehälter pro DepId

- Mitarbeiter mit den höchsten Gehältern in ihrer Abteilung

```
HR.EMPLOYEES
EMPLOYEE_ID NOT NULL NUMBER
FIRST_NAME           VARCHAR2
LAST_NAME   NOT NULL VARCHAR2
SALARY               NUMBER

DEPARTMENT_ID        NUMBER
```

# Max Gehälter pro DepId

```
SELECT E1.FIRST_NAME
     , ...
     , E1.DEPARTMENT_ID
  FROM HR.EMPLOYEES E1
 WHERE E1.SALARY = (
    SELECT MAX(E2.SALARY)
      FROM HR.EMPLOYEES E2
     WHERE E2.DEPARTMENT_ID =
           E1.DEPARTMENT_ID
OR ( E2.DEPARTMENT_ID IS NULL
 AND E1.DEPARTMENT_ID IS NULL
     ))
```

# Max Gehälter pro DepId

```
-----------------------------------------------
| Id| Operation                 | Name      | Cost|
-----------------------------------------------
|  0 |SELECT STATEMENT          |           |   39|
|  1 | FILTER                   |           |     |
|  2 |   TABLE ACCESS FULL      |EMPLOYEES  |    3|
|  3 |   SORT AGGREGATE         |           |     |
|  4 |    TABLE ACCESS FULL     |EMPLOYEES  |    3|
-----------------------------------------------
```

# Max Gehälter pro DepId

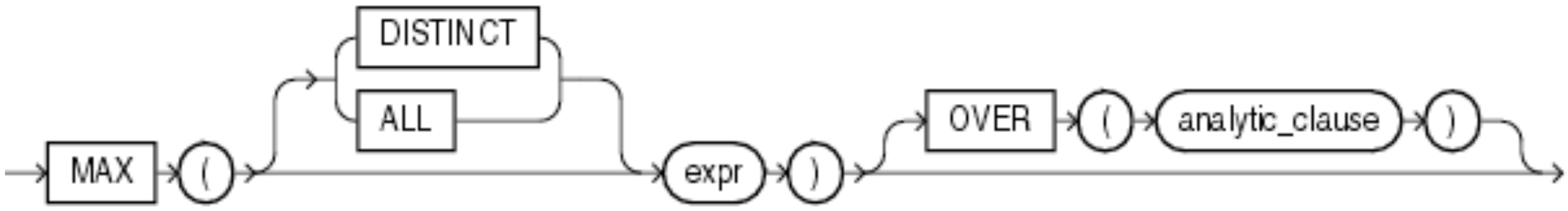
```
SELECT E1.FIRST_NAME
     , E1.LAST_NAME
     , E1.SALARY
     , E1.DEPARTMENT_ID
  FROM HR.EMPLOYEES E1
 WHERE E1.SALARY = (
    SELECT MAX(E2.SALARY)
      FROM HR.EMPLOYEES E2
     WHERE
          NVL(E2.DEPARTMENT_ID,-1) =
          NVL(E1.DEPARTMENT_ID,-1)
                )
```

# Max Gehälter pro DepId

```
-----------------------------------------------
|Id| Operation                    | Name      |Cost|
-----------------------------------------------
| 0|SELECT STATEMENT              |           |   8|
| 1| HASH JOIN                    |           |   8|
| 2|  VIEW                        |VW_SQ_1    |   4|
| 3|   FILTER                     |           |    |
| 4|    HASH GROUP BY             |           |   4|
| 5|     TABLE ACCESS FULL        |EMPLOYEES  |   3|
| 6|  TABLE ACCESS FULL           |EMPLOYEES  |   3|
-----------------------------------------------
```

# analytic Function MAX

maximaler Wert innerhalb einer Gruppe




```
SELECT LAST_NAME
     , SALARY
     , DEPARTMENT_ID
     , MAX(SALARY) OVER
       (PARTITION BY DEPARTMENT_ID)
FROM HR.EMPLOYEES
```

http://docs.oracle.com/cd/E11882_01/server.112/e26088/functions098.htm#i89072

# analytic Function MAX

| LAST_NAME | SALARY | DEP_ID | MAX_SAL |
|---|---|---|---|
| Whalen | 4400 | 10 | 4400 |
| Hartstein | 13000 | 20 | 13000 |
| Fay | 6000 | 20 | 13000 |
| Raphaely | 11000 | 30 | 11000 |
| Khoo | 3100 | 30 | 11000 |
| ... | | | |
| Colmenares | 2500 | 30 | 11000 |
| ... | | | |
| Grant | 7000 | | 7000 |

# Max Gehälter pro DepId

```
SELECT  FIRST_NAME
      , LAST_NAME
      , SALARY
      , DEPARTMENT_ID
FROM  (SELECT FIRST_NAME
      , ...
      , MAX(SALARY) OVER
          (PARTITION BY
           DEPARTMENT_ID)
        AS MAX_SAL
    FROM HR.EMPLOYEES) E
WHERE E.SALARY = E.MAX_SAL
```

# Max Gehälter pro DepId

```
-------------------------------------------------
|Id|Operation                    | Name      | Cost|
-------------------------------------------------
| 0|SELECT STATEMENT             |           |    4|
| 1| VIEW                        |           |    4|
| 2|  WINDOW SORT                |           |    4|
| 3|   TABLE ACCESS FULL|EMPLOYEES|    3|
-------------------------------------------------
```

# Max Gehälter pro DepId

- OR – Verknüpfung
  -> eine "="-Verknüpfung
  -> analytic function MAX
- Costs von 39
  -> 8
  -> 4
- 2 full table scans
  -> 1 full table scan

# Tipps (I)

- Kenne das Datenmodell
- Kenne die Daten
- Selektiere nur Spalten, die du brauchst

# Tipps (I)

~~~~
SELECT *
  FROM HR.EMPLOYEES
~~~~

```
SELECT  FIRST_NAME
      , LAST_NAME
      , SALARY
FROM  HR.EMPLOYEES
```

# Tipps (I)

```
SELECT E1.LAST_NAME
  FROM HR.EMPLOYEES E1
 WHERE EXISTS
       ( SELECT *
           FROM HR.EMPLOYEES E2
          WHERE E1.DEPARTMENT_ID =
                E2.DEPARTMENT_ID)

WHERE EXISTS
       ( SELECT 1
           FROM HR.EMPLOYEES E2
          WHERE E1.DEPARTMENT_ID =
                E2.DEPARTMENT_ID)
```

# Tipps (I)

- Joine nur Tabellen, die du brauchst
- Schränke früh die Datenmenge ein
- Verwende “AND” und “=” Verknüpfungen

# höchsten 5 Gehälter

- Mitarbeiter mit den höchsten 5 Gehältern (mehrere Mitarbeiter können das selbe verdienen)

**HR.EMPLOYEES**

| | | |
|---|---|---|
| EMPLOYEE_ID | NOT NULL | NUMBER |
| FIRST_NAME | | VARCHAR2 |
| LAST_NAME | NOT NULL | VARCHAR2 |
| SALARY | | NUMBER |
| DEPARTMENT_ID | | NUMBER |

# höchsten 5 Gehälter

```
SELECT E1.FIRST_NAME, E1.LAST_NAME
     , E1.SALARY
     , E1.DEPARTMENT_ID
  FROM (SELECT SALARY
             , ROWNUM POS
          FROM (SELECT DISTINCT
                       SALARY
                  FROM HR.EMPLOYEES
                 ORDER BY SALARY DESC)
                ) E_SAL
     , HR.EMPLOYEES E1
WHERE E1.SALARY = E_SAL.SALARY
  AND E_SAL.POS <= 5
```

# höchsten 5 Gehälter

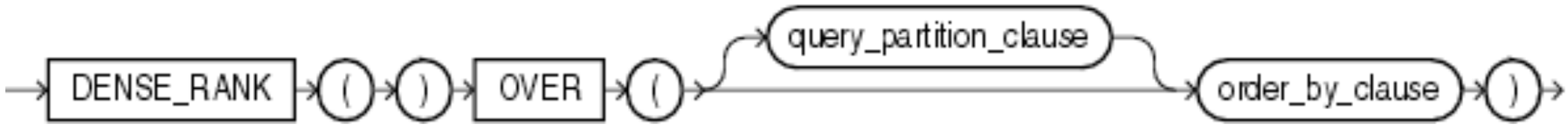
```
SELECT E1.FIRST_NAME, E1.LAST_NAME
     , E1.SALARY
     , E1.DEPARTMENT_ID
  FROM (SELECT SALARY
             , ROWNUM POS
          FROM (SELECT DISTINCT
                       SALARY
                  FROM HR.EMPLOYEES
                 ORDER BY SALARY DESC)
       ) E_SAL
     , HR.EMPLOYEES E1
 WHERE E1.SALARY = E_SAL.SALARY
   AND E_SAL.POS <= 5
```

# höchsten 5 Gehälter

```
-------------------------------------------------------
|Id|  Operation                    |  Name    |Cost|
-------------------------------------------------------
|  0|SELECT  STATEMENT             |          |    9|
|  1|  HASH JOIN                   |          |    9|
|  2|   VIEW                       |          |    5|
|  3|    COUNT                     |          |     |
|  4|     VIEW                     |          |    5|
|  5|      SORT UNIQUE             |          |    4|
|  6|       TABLE ACCESS FULL|EMPLOY..|    3|
|  7|   TABLE ACCESS FULL          |EMPLOY..|    3|
-------------------------------------------------------
```

# DENSE_RANK

verdichteter Rang innerhalb einer Gruppe
abhängig von der Sortierung




```
SELECT LAST_NAME
     , SALARY
     , DEPARTMENT_ID
     , DENSE_RANK() OVER (
         PARTITION BY DEPARTMENT_ID
         ORDER BY SALARY DESC)
  FROM HR.EMPLOYEES
```

http://docs.oracle.com/cd/E11882_01/server.112/e26088/functions052.htm#i1064409

# DENSE_RANK

| LAST_NAME | SALARY | DEP_ID | EMP_RANK |
|---|---|---|---|
| Whalen | 4400 | 10 | 1 |
| Hartstein | 13000 | 20 | 1 |
| Fay | 6000 | 20 | 2 |
| Raphaely | 11000 | 30 | 1 |
| Khoo | 3100 | 30 | 2 |
| Baida | 2900 | 30 | 3 |
| Tobias | 2800 | 30 | 4 |
| Himuro | 2600 | 30 | 5 |
| Colmenares | 2500 | 30 | 6 |
| ... | | | |
| Grant | 7000 | | 1 |

# DENSE_RANK

```
SELECT  LAST_NAME
      , SALARY
      , DENSE_RANK()  OVER
        (ORDER BY SALARY DESC)
  FROM HR.EMPLOYEES
```

```
LAST_NAME  SALARY  EMP_RANK
---------  ------  --------
King         24000         1
Kochhar      17000         2
De Haan      17000         2
Russell      14000         3
Partners     13500         4
```

# höchsten 5 Gehälter

```
SELECT  E_RANK.FIRST_NAME
      , E_RANK.LAST_NAME
      , E_RANK.SALARY
      , E_RANK.DEPARTMENT_ID
  FROM (SELECT DENSE_RANK() OVER
                (ORDER BY SALARY DESC)
                 EMP_RANK
             , FIRST_NAME
             , LAST_NAME
             , SALARY
             , DEPARTMENT_ID
          FROM HR.EMPLOYEES) E_RANK
 WHERE E_RANK.EMP_RANK <= 5
```

# höchsten 5 Gehälter

```
-------------------------------------------------
|Id| Operation                    | Name      | Cost|
-------------------------------------------------
| 0 |SELECT STATEMENT             |           |    4|
| 1 | VIEW                        |           |    4|
| 2 |  WINDOW SORT                |           |     |
|                PUSHED RANK|                 |    4|
| 3 |   TABLE ACCESS FULL|EMPLOYEES|         |    3|
-------------------------------------------------
```

# COALESCE statt NVL

- NVL (Parameter1, Parameter2)
  Parameter1 wird durch Parameter2 ersetzt, wenn Parameter1 NULL ist
- COALESCE (Param1, Param2, Param3,...)
  liefert den ersten Parameter der NOT NULL ist
- Verwende COALESCE, da es nur so viele Parameter evaluliert, bis der erste Parameter der NOT NULL ist, gefunden wurde

# COALESCE statt NVL

```
CREATE OR REPLACE FUNCTION
Fehlerhafte_Funktion
RETURN VARCHAR2
AS
     vZahl NUMBER;    --FEHLER!!!
BEGIN
     vZahl := 'abc';
     RETURN(vZahl);
END;
```

# COALESCE statt NVL

```
SELECT NVL('Hallo',
           Fehlerhafte_Funktion)
  FROM DUAL;

ORA-06502: PL/SQL: numeric or value
error: character to number
conversion error
ORA-06512: at
"xxx.FEHLERHAFTE_FUNKTION", line 6
06502. 00000 -  "PL/SQL: numeric or
value error%s"
```

# COALESCE statt NVL

```
SELECT COALESCE('Hallo',
                Fehlerhafte_Funktion)
  FROM DUAL;


ERGEBNIS_COALESCE
-----------------
Hallo
```

# CASE statt NVL2

NVL2 (Parameter1, Parameter2, Parameter3)

**Wenn Parameter1 NOT NULL**

**dann wird Parameter2 zurückgegeben,**
**sonst Parameter3**

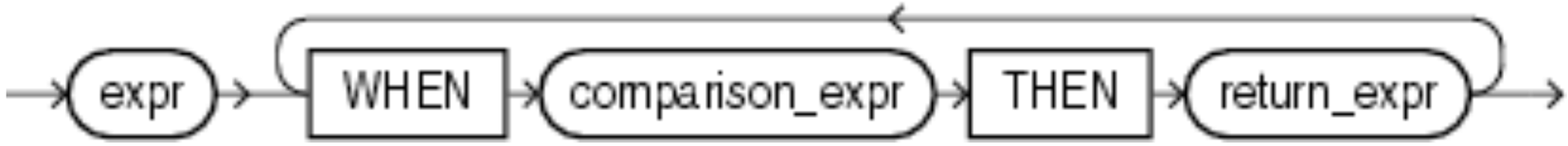
# CASE statt NVL2

```
SELECT LAST_NAME
     , DEPARTMENT_ID
     , NVL2( DEPARTMENT_ID
           , 'Abt.: '||DEPARTMENT_ID
           , 'KEINE ABTEILUNG')
  FROM EMPLOYEES
```

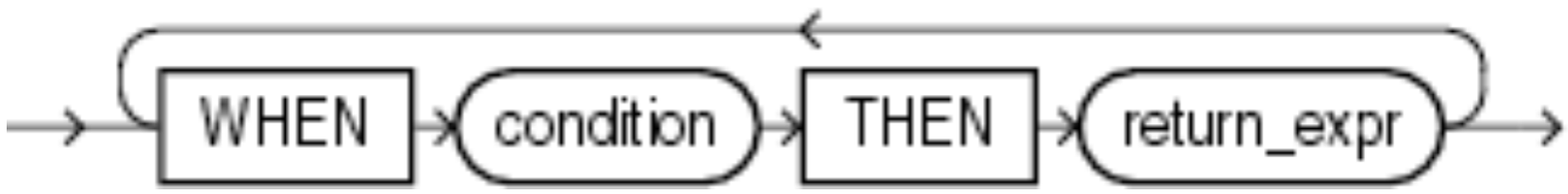
```
LAST_NAME DEP_ID NVL2
--------- ------ ----------------
King           90 Abt.: 90
Kochhar        90 Abt.: 90
Grant             KEINE ABTEILUNG
```

# CASE statt NVL2

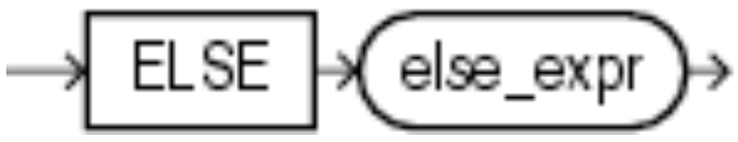

**simple_case_expression::=**




**searched_case_expression::=**




**else_clause::=**




http://docs.oracle.com/cd/B28359_01/server.111/b28286/expressions004.htm#SQLRF20037

# CASE statt NVL2

```
CASE
     WHEN DEPARTMENT_ID IS NOT NULL
     THEN 'Abt.: '||DEPARTMENT_ID
     ELSE 'KEINE ABTEILUNG'
END
```

```
LAST_NAME DEP_ID NVL2
--------- ------ ----------------
King           90 Abt.: 90
Kochhar        90 Abt.: 90
Grant             KEINE ABTEILUNG
```

# CASE statt NVL2

```
SELECT NVL2('Hallo','wenn NOT NULL'
                   ,Fehlerhafte_Funktion)
  FROM DUAL;


ORA-06502: PL/SQL: numeric or value
error: character to number conversion
error
ORA-06512: at
"xxx.FEHLERHAFTE_FUNKTION", line 6
06502. 00000 -  "PL/SQL: numeric or
value error%s"
```

# CASE statt NVL2

```
SELECT CASE
            WHEN 'Hallo' IS NOT NULL
            THEN 'wenn NOT NULL'
            ELSE FEHLERHAFTE_FUNKTION
       END
  FROM DUAL
```

```
ERGEBNIS_CASE
--------------
wenn NOT NULL
```

# Funktionen & WHERE

```
SELECT *
  FROM HR.EMPLOYEES
 WHERE LAST_NAME = 'King';


-----------------------------------------
|Id| Operation           | Name         | Cost|
-----------------------------------------
|0| SELECT STATEMENT     |              |    2|
|1|  TABLE ACCESS BY
        INDEX ROWID       |EMPLOYEES     |    2|
|2|   INDEX RANGE SCAN    |EMP_NAME_IX   |    1|
-----------------------------------------
```

# Funktionen & WHERE

```
SELECT *
  FROM HR.EMPLOYEES
 WHERE UPPER(LAST_NAME) = 'KING';
```

```
-----------------------------------------
|Id| Operation                 | Name     | Cost|
-----------------------------------------
| 0 |SELECT STATEMENT          |          |    3|
| 1 |  TABLE ACCESS FULL       |EMPLOYEES |    3|
-----------------------------------------
```

# Funktionen & WHERE

```
SELECT *
  FROM HR.EMPLOYEES
WHERE INITCAP(LAST_NAME) = 'King'

WHERE LOWER(LAST_NAME) = 'king'

WHERE SUBSTR(PHONE_NUMBER,1,
INSTR(PHONE_NUMBER,'.',1,1)-1) = ..

WHERE TO_CHAR(HIRE_DATE,‘YYYY’) =..
```

**schalten vorhandene Indices aus**

# Funktionen & WHERE

- Zusätzliche indizierte Spalte befüllen, z.B. mit Datenbank-Trigger
- Function based index (pro und contra) auch eigene Funktionen möglich

- Index wird auch ausgeschalten, wenn wir zu einer Zahl 0 addieren, bzw. NULL zu einem Character-Feld dazuhängen

# und ohne Funktionen?

```
CREATE TABLE UB_EMPLOYEES
( EMPLOYEE_ID   VARCHAR2(3)
, FIRST_NAME    VARCHAR2(20)
, LAST_NAME     VARCHAR2(25)
, CONSTRAINT UB_EMP_PK
                PRIMARY KEY
               (EMPLOYEE_ID)
);
```

# und ohne Funktionen?

```
SELECT *
  FROM HR.UB_EMPLOYEES
 WHERE EMPLOYEE_ID = 101;
```

```
-----------------------------------------
|Id| Operation                | Name     | Cost|
-----------------------------------------
| 0 |SELECT STATEMENT         |          |    3|
| 1 | TABLE ACCESS FULL|UB_EMP...|    3|
-----------------------------------------
```

# und ohne Funktionen?

```
SELECT *
  FROM HR.UB_EMPLOYEES
 WHERE TO_NUMBER(EMPLOYEE_ID) = 101
```

# und ohne Funktionen?

```
SELECT *
  FROM HR.UB_EMPLOYEES
 WHERE TO_NUMBER(EMPLOYEE_ID) = 101

 WHERE EMPLOYEE_ID = '101'

-------------------------------------------
|Id| Operation                | Name       |Cost|
-------------------------------------------
| 0 |SELECT STATEMENT         |            |   1|
| 1 | TABLE ACCESS BY
          INDEX ROWID          |UB_EMP...   |   1|
| 2 |   INDEX UNIQUE SCAN      |UB_EMP_PK   |   1|
-------------------------------------------
```

# Context Switch

- PL/SQL- Funktion in einer SQL-Abfrage erzeugt einen Context Switch.
- Einmalig keine besondern Kosten...
- v('APP_USER')

# Context Switch

```
CREATE OR REPLACE
FUNCTION getUser
RETURN VARCHAR2
AS
BEGIN
     RETURN('ULI');
END;
```

- UB_EMP: 1 000 000 Datensätze
- Spalte USER_NAME (1000 Treffer)

# Context Switch

```
SELECT COUNT(*)
FROM UB_EMP

WHERE USER_NAME = getUser
```

- 2,9 sec

```
WHERE USER_NAME = 'ULI'
```

- 0,024 sec

# Context Switch

- Subquery

```
SELECT COUNT(*)
   FROM UB_EMP
  WHERE USER_NAME =
          (SELECT getUser FROM DUAL)
```

# Context Switch

- factored Subquery

```
WITH UB_NAME AS
 ( SELECT getUser Name
     FROM DUAL
 )
SELECT  COUNT(*)
   FROM  UB_EMP
       ,  UB_NAME
  WHERE  UB_NAME.NAME =
         UB_EMP  .USER_NAME
```

# Context Switch

- scalar Subquery

```
SELECT E.LAST_NAME
     , FUNCTION(E.DEP_ID)
  FROM EMPLOYEES E
```

```
SELECT E.LAST_NAME
     , ( SELECT FUNCTION(E.DEP_ID)
           FROM DUAL )
  FROM EMPLOYEES E
```

# Tipps (2)

- Prüfe jeden full table scan (bei 'kleinen Tabellen' kann ein full table scan schnell sein)
- Aktualisiere die Optimizer Statistics
- Erzeuge pro Anwendungsfall eine View
- Analysiere auch 'kleine' Erweiterungen
- Minimiere die Anzahl der Aufrufe der selben Tabelle

# Tipps (2)

```
SELECT
   ( SELECT COUNT(*)
       FROM EMPLOYEES
      WHERE SALARY < 15000) S1
,  ( SELECT COUNT(*)
       FROM EMPLOYEES
      WHERE SALARY >= 15000) S2
FROM DUAL
```

# Tipps (2)

```
SELECT
      COUNT(
         CASE WHEN SALARY < 15000
              THEN 1
              ELSE NULL
         END) S1
   , COUNT(
         CASE WHEN SALARY >= 15000
              THEN 1
              ELSE NULL
         END) S2
     FROM EMPLOYEES
```

# Speed up your Query

- analytic Functions MAX und DENSE_RANK
- COALESCE und CASE

- Funktionen angewand auf Spalten, schalten Indices aus
- Funktionen im Select erzeugen Context Switches

click-click
IT Solutions

# Ulrike Brenner


## click-click IT Solutions

Welschgasse 8
A-1230 Wien

ulrike.brenner@click-click.at
http://www.click-click.at
http://www.wirsindapex.at