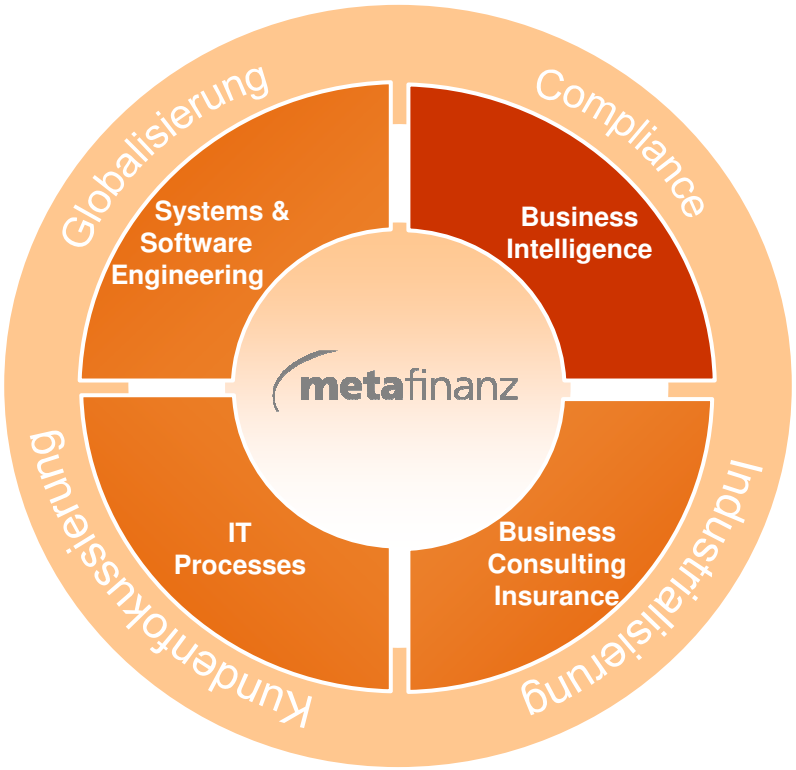
2010
DOAG
Konferenz + Ausstellung

MS-SQL als DWH-Quelle, angebunden per JDBC/OpenSource-Framework ORAJDBCLINK

Günther Herzog

20 Jahre dynamic X²cellence

metafinanz GmbH ist ein Software- und Beratungshaus mit Schwerpunkt in der Finanzdienstleistungsbranche. Als Generator effizienter Business-Prozesse übernehmen wir End-to-End-Verantwortung und unterstützen unsere Kunden in ihren Projektvorhaben.



Ihr Partner für:

- **Datawarehousing**
  (Oracle, OWB, Informatica)
- **Business Intelligence**
  (SAS, Cognos, Oracle, Microsoft)
- Schwerpunkte:
  Datenqualität, Sicherheit und **Compliance**
- Offene **Schulungen** und maßgeschneiderte **Coachings**

Partnerships:

ORACLE Microsoft COGNOS INFORMATICA SAS

Gliederung

Gliederung

Motivation

- Anbindung einer zusätzlichen Datenquelle an ein ORACLE-Data Warehouse notwendig.
- Kurzfristige Umsetzung notwendig, keine Zeit für Produktevaluation
- Schonung Projektbudget
- Benötigte Datenmenge eher gering.

Gliederung

Features

- Einbindungsmöglichkeit für alle per JDBC (Java DataBase Connectivity) erreichbaren Datenbanken wie z.B.:
  - ORACLE vor Version 8
  - Microsoft SQL-Server
  - MySQL
  - IBM DB2
  - Sybase
- JDBC-Treiber für nicht-ORACLE-DBs werden in der ORACLE-Datenbank abgelegt
- JDBC-Treiber für (alte) ORACLE-DBs bereits in ORACLE enthalten
- Gleichzeitiger Zugriff auf mehrere Datenbanken möglich
- Für jeden Datenbanktyp eigener JDBC-Treiber notwendig

Gliederung

Voraussetzung

Aktivierte JavaOption JVM

(deswegen auf 10gXE nicht lauffähig)

Installation

1. Anlegen eines Framework-Users JDBCLINK als SYSDBA:

```
create user jdbclink identified by jdbclink;
```

2. Benötigte Grants wie das JavaUserPriv und das JavaSysPriv:

```
GRANT JAVAUSERPRIV, JAVASYSPRIV TO JDBCLINK;
```

2. Laden der benötigten JAVA Jar-Dateien der JDBC-Treiber mit loadjava (unter Windows)

```
loadjava -resolve -verbose -user jdbclink/jdbclink@orcl jcifs-1.3.13.jar
jtds-1.2.2.jar
```

3. Anlegen aller benötigten Javaklassen, Typen und Objekten als User JDBCLINK.

```
sqlplus jdbclink/jdbclink@ora @initoracletoany.sql
```

4. Es sind die Objektberechtigungen für die User, die das Framework benutzen dürfen zu vergeben:

```
grant all on jdbclink.jcursor to hr;
grant all on jdbclink.jcall to hr;
grant all on jdbclink.jdbc_dblink to hr;
```

5. Das Verschlüsselungs-Package compilieren (eine Eigenentwicklung des Projekts)

```
sqlplus jdbclink/jdbclink@ora @utl_decrypt.sql
```

6. Die Datenbankzugriffe auf die JDBC-Quellen in die Config-Tabelle `JDBC_DBLINK` eintragen:

Ohne Verschlüsselung des Passwortes

```
INSERT INTO JDBC_DBLINK
(DATA_SOURCE_NAME,URL,DBUSER,DBPASSWORD,DRIVER) VALUES
('mssql2000','jdbc:jtds:sqlserver://oradb:1433','oracle','o
racle','net.sourceforge.jtds.jdbc.Driver');
```

Mit Verschlüsselung des Passwortes

```
INSERT INTO JDBC_DBLINK
(DATA_SOURCE_NAME,URL,DBUSER,DBPASSWORD,DRIVER) VALUES
('mssql2000','jdbc:jtds:sqlserver://oradb:1433','oracle',UT
L_DECRYPT.ENCRYPT('oracle'),'net.sourceforge.jtds.jdbc.Driv
er');
```

Gliederung

Daten abrufen

Für jedes SELECT wird ein Record Type angelegt der in einer separaten Funktion die Daten per Pipelined Table Function zurückgibt. Typischerweise wird pro JDBC-Datenquelle ein Package angelegt und die Funktionen beinhalten den Namen der Quelltabelle.

```
create or replace PACKAGE HR.MSSQL AS
  TYPE EMPLOYEES_RECORD IS RECORD
  (
    "EMPLOYEE_ID"  NUMBER(6,0),
    "FIRST_NAME"   VARCHAR2(20 BYTE),
    "LAST_NAME"    VARCHAR2(25 BYTE),
    "EMAIL"        VARCHAR2(25 BYTE),
    "INSERT_DATE"    DATE
  );

  TYPE EMPLOYEES_TABLE IS TABLE OF EMPLOYEES_RECORD;

  FUNCTION employees RETURN employees_table pipelined;

END MSSQL;
/
```

Daten abrufen

```
CREATE OR REPLACE PACKAGE BODY HR.MSSQL AS
  FUNCTION EMPLOYEES RETURN EMPLOYEES_TABLE PIPELINED AS
    V_CURSOR jdbclink.JCURSOR:= jdbclink.JCURSOR
           ('select EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL from EMPLOYEES',
'MSSQL2000',1);
    V_RECORD EMPLOYEES_RECORD;
BEGIN
  v_cursor.init; -- open connection, and prepare query
  v_cursor.open; -- execute query
  while v_cursor.dofetch = 1 loop -- fetch query results into record
    V_RECORD.EMPLOYEE_ID      := V_CURSOR.GET_NUMBER(1);
    V_RECORD.FIRST_NAME       := V_CURSOR.GET_STRING(2);
    V_RECORD.LAST_NAME        := V_CURSOR.GET_STRING(3);
    V_RECORD.EMAIL            := V_CURSOR.GET_STRING(4);
    pipe row (v_record); -- pipe row to the query
  end loop;
  v_cursor.close;  -- close resources
EXCEPTION
  WHEN OTHERS THEN -- if something happens
    V_CURSOR.CLOSE;  -- close resources
    RAISE;
  END EMPLOYEES;
END MSSQL;
```

Daten abrufen

Ein Abruf der Daten ist über den Aufruf der Table-Function zu realisieren.

```
SELECT EMPLOYEE_ID
     , FIRST_NAME
     , LAST_NAME
     , EMAIL
     , SYSDATE

FROM TABLE(HR.MSSQL.EMPLOYEES);
```

Performance

Übertragung aller Daten:

```
SELECT EMPLOYEE_ID
     , FIRST_NAME
     , LAST_NAME
     , EMAIL
FROM TABLE(HR.MSSQL.EMPLOYEES)
```

Ebenso Übertragung aller Daten:

```
SELECT EMPLOYEE_ID
     , FIRST_NAME
     , LAST_NAME
     , EMAIL
FROM TABLE(HR.MSSQL.EMPLOYEES)
where LAST_NAME = 'Baer';
```

- Filterung sollte auf der Quelldatenbank erfolgen

Parameterfilter per Bind-Variable

```
FUNCTION EMPLOYEES_BY_LAST_NAME(P_LAST_NAME IN VARCHAR2) RETURN
EMPLOYEES_TABLE PIPELINED as
v_cursor jdbclink.jcursor:= jdbclink.jcursor('select EMPLOYEE_ID,
LAST_NAME from employees where LAST_NAME= ?','MSSQL2000',2);
v_record EMPLOYEES_RECORD;
BEGIN
v_cursor.init;
v_cursor.bind(1, P_LAST_NAME);
v_cursor.open;
while v_cursor.dofetch = 1 loop
    V_RECORD.EMPLOYEE_ID      := V_CURSOR.GET_NUMBER(1);
    V_RECORD.LAST_NAME        := V_CURSOR.GET_STRING(2);
pipe row (v_record);
end loop;
v_cursor.close;
exception
when others then
  v_cursor.close;
  raise;
END EMPLOYEES_BY_LAST_NAME;
```

Füllen einer Staging-Tabelle

```
FUNCTION LOAD_EMPLOYEES_STG RETURN NUMBER AS

BEGIN
  DELETE STG_EMPLOYEES; --vorherige Daten löschen

  INSERT INTO STG_EMPLOYEES
        (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, INSERT_DATE)
  SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, SYSDATE
  FROM TABLE(MSSQL.EMPLOYEES);
  COMMIT;
  return 1;

EXCEPTION
    WHEN OTHERS THEN
     ROLLBACK; --Falls es Probleme gibt, sind nach einem Rollback
               -- die vorherigen Daten wieder verfügbar.
     RETURN 0;

END LOAD_EMPLOYEES_STG;
```

Anbinden von mehr als einer JDBC-Datenquelle

- Unterstützung mehrerer gleichzeitig nutzbarer JDBC-Connections
- Pro JDBC-Data-Source ein Eintrag in der Tabelle JDBC_DBLINK
- Datenquelle wird per DATA_SOURCE_NAME erkannt und benutzt
- Empfehlenswert: für jede angebundene Datenbank eigenes Package

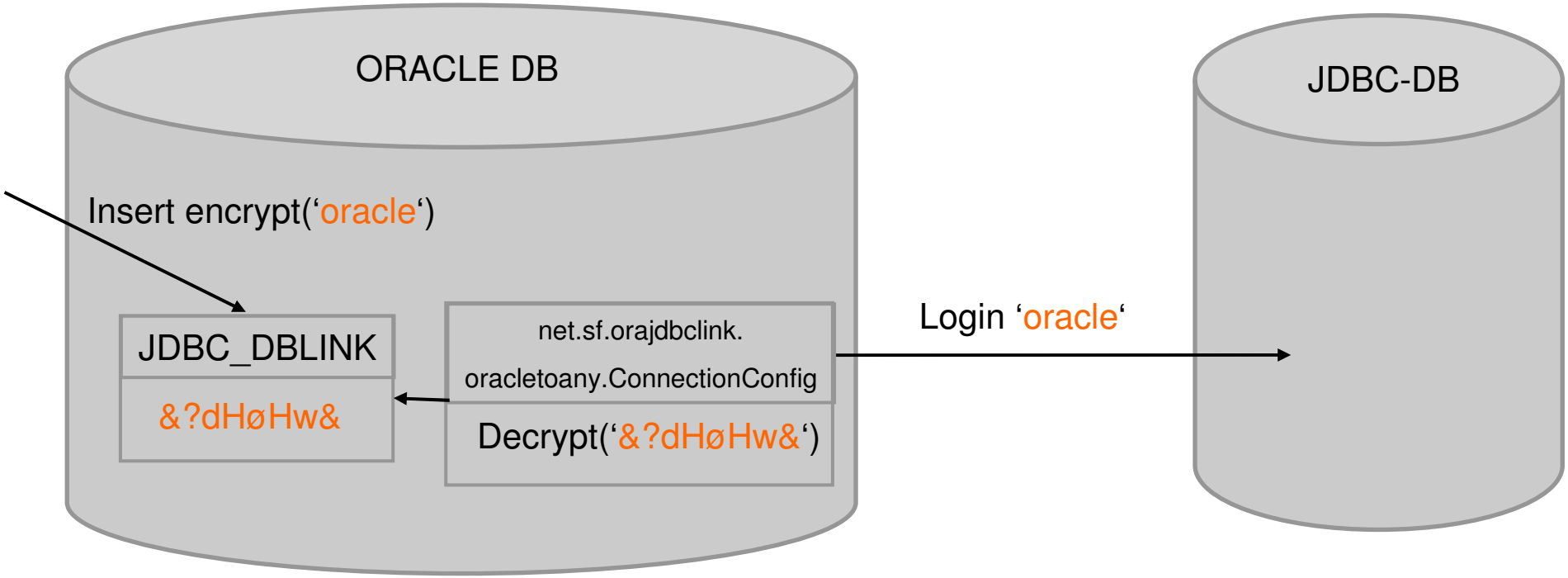
Tabelle JDBC_DBLINK

DATA_SOURCE_NAME	URL	DBUSER	DBPASSWORD	DRIVER
MSSQL2000	jdbc:jtds:sqlserver://orcl:1433	oracle	&?dHøHw&	net.sourceforge.jtds.Driver
MYSQL	jdbc:mysql://mysqlserver:3306/mydb	myuser	c&T?dH&ø	com.mysql.jdbc.Driver

Gliederung

Verschlüsselung des Passwortes

- Passwort soll in der Config-Tabelle JDBC_DBLINK nicht im Klartext zu lesen sein
- Verschlüsseltes Passwort soll nicht vom Framework-User entschlüsselt werden können.



Verschlüsselungspackage

- Verwendung Package DBMS_OBFUSCATION_TOOLKIT
- Verwendung 3DES-Algorithmus
- DBMS_OBFUSCATION_TOOLKIT. DES3ENCRYPT (Verschlüsseln)
- DBMS_OBFUSCATION_TOOLKIT. DES3DECRYPT (Entschlüsseln)
- Seit ORACLE 8i
- Key: Mindestlänge 128bit d.h. 16 Zeichen
- Passwortlänge muss ein Vielfaches der Länge von 8 Byte sein
- Vor Verschlüsselung evtl. Verlängerung des Passwortes notwendig

```
i := 8 - MOD( length('PASS'), 8 );
LV_PASSWORD2ENCRYPT := 'PASS'||RPAD(CHR(i), i, CHR(i));
```

Verschlüsselungspackage

Als Verschlüsselung werden die Prozeduren DES3ENCRYPT bzw. DES3DECRYPT des Packages DBMS_OBFUSCATION_TOOLKIT in dem der 3DES-Algorithmus verwendet wird. Dieses Package steht seit der ORACLE Datenbank Version 8i zur Verfügung.

Der Key für die Verschlüsselung muss eine Mindestlänge von 128bit d.h. 16 Zeichen haben.

Die Länge einer Zeichenkette, die verschlüsselt werden soll (hier das Passwort), muss ein Vielfaches der Länge von 8 Byte besitzen, wobei das selbstentwickelte Package UTL_DECRYPT kürzere Zeichenketten vor der Verschlüsselung verlängert und beim entschlüsseln wieder dementsprechend kürzt. Zunächst die Verlängerung:

```
i := 8 - MOD( length('PASS'), 8 );
LV_PASSWORD2ENCRYPT := 'PASS'||RPAD(CHR(i), i, CHR(i));
```

Hier wird zuerst die Anzahl der Stellen ermittelt, die der Zeichenkette "PASS" fehlen, um auf eine Länge zu gelangen, die einem Vielfachen der Länge von 8 Byte entspricht. Anschließend wird die ermittelte Anzahl an aufzufüllenden Stellen als ASCII-Code interpretiert und der CHR Funktion übergeben.

Die CHR Funktion liefert zu einem ASCII-Code das entsprechende Zeichen aus der ASCII-Tabelle zurück. Danach wird die "PASS"- Zeichenkette so oft um das Zeichen aus der ASCII-Tabelle erweitert, bis die Gesamtlänge der Zeichenkette ein Vielfaches der Länge von 8 Byte besitzt.

Verschlüsselungspackage

- Nach Entschlüsselung Verkürzung notwendig

Für die Entschlüsselung der Zeichenkette gilt das analoge Verfahren. Dabei muss der Klartext um die erweiterten Zeichen aus der ASCII-Tabelle gekürzt werden:.

```
V_LAENGE    := LENGTH(V_KLARTEXT);
V_KLARTEXT  := RPAD(V_KLARTEXT,V_LAENGE-ASCII(SUBSTR(V_KLARTEXT,
  V_LAENGE)));
```

Bei dieser Vorgehensweise wird die Zeichenkette immer nur um Steuerzeichen wie z. B. ^E oder ^G erweitert, die in Zeichenketten nur selten vorkommen.

Verschlüsselung des JDBC-Passwortes

- Erweiterung Konstruktormethode von Java-Klasse net.sf.orajdbclink.oracletoany.ConnectionConfig

```
//Verschlüsselungsaufruf vorbereiten
CallableStatement callableStatement = conn.prepareCall("{? = call
  UTL_DECRYPT.DECRYPT(?)}");
//ausgehenden Parameter registrieren
callableStatement.registerOutParameter(1, java.sql.Types.VARCHAR);
//bereits geholtes Passwort als Bind-Variable setzen
callableStatement.setString(2, password);
callableStatement.execute();
//Entschlüsseltes PW abholen
password= callableStatement.getString(1);
callableStatement.close();
```

Verschlüsselung-Key

- Ablage Verschlüsselungspackage im ORAJDBCLINK-Schema
- Verschlüsselungskey im Verschlüsselungspackage
- Objektberechtigung nur für User ORAJDBCLINK
- Wrap des Package-Bodys wg. SELECT ANY DICTIONARY - Recht denkbar

Vorteil:

- Passwort sicher verschlüsselt
- Verschlüsselungs-Key kann von anderen Usern nicht im UTL_DECRYPT-Package gelesen werden

Nachteil

- Stets Verwendung desselben Verschlüsselungskeys

Gliederung

Nachteile

- Nicht flexibel genug bzw. ungeeignet für Adhoc-Abfragen
- Jedes neue SELECT benötigt eigene Prozedur
- SQL als String hinterlegt

Vorteile

- Komplett in der Datenbank zu verwalten
- Kann an eigene Bedürfnisse angepasst werden
- Open Source
- Kostenlos
- Anbindung mehrerer JDBC-Datenquellen möglich
- Schnell einzubinden

ORAJDBCLINK

Nettes kleines Framework, das bei kleineren Datenaufkommen sehr schnell und unproblematisch neue Datenquellen an ein Data Warehouse anbinden kann ohne in kommerzielle Produkte investieren zu müssen.

Metafinanz auf der DOAG 2010

- 17.11.2010 / 10:00 Uhr / Raum Helsinki
  Einführung OWB Java API

- 17.11.2010 / 12:00 Uhr / Raum Helsinki
  Vergleich von OWB Repositories mittels Skripting

- 17.11.2010 / 16:00 Uhr / Raum Helsinki
  Arbeiten mit Template Mappings

- 18.11.2010 / 10:00 Uhr / Raum Stockholm
  The beast - Umstieg von einer E20K nach Exadata Version V2

- 18.11.2010 / 12:00 Uhr / Raum Stockholm
  ORACLE Change Data Capture (CDC) - Die Änderungen entscheiden

- 18.11.2010 / 15:00 Uhr / Raum Stockholm
  MS-SQL als DWH-Quelle, angebunden per JDBC-Framework ORAJDBCLINK

Interesse? Fragen? Austausch?

Besuchen Sie unseren Stand…

Aktuelle Schulungen:
OWB11gR2 New Features
&
Skripting mit OMB*Plus
Mehr Info an unserem Stand!

Sie finden uns auf Ebene 1, Stand 112 (neben der Mercator Lounge)
Mehr Informationen und Downloads unter: http://owb.metafinanz.de

Herzlichen Dank!




metafinanz
Informationssysteme GmbH
Leopoldstr. 146
80804 München
phone: +49 89 360531-0
fax: +49 89 360531-15
kontakt@metafinanz.de
www.metafinanz.de

UTL_DECRYPT Header

```
create or replace PACKAGE UTL_DECRYPT AS

  -- encrypt/verschlsseln
  FUNCTION ENCRYPT(p_password2encrypt IN VARCHAR2, p_key IN VARCHAR2 DEFAULT NULL)
  RETURN VARCHAR2;

  -- decrypt/entschlsseln
  FUNCTION DECRYPT(p_password2decrypt IN VARCHAR2, p_key IN VARCHAR2 DEFAULT NULL)
  RETURN VARCHAR2;

END UTL_DECRYPT;
```

UTL_DECRYPT Body 1/2

```
create or replace
PACKAGE BODY UTL_DECRYPT AS

  -- Verschluesseln
  FUNCTION ENCRYPT(P_PASSWORD2ENCRYPT IN VARCHAR2, P_KEY IN VARCHAR2 DEFAULT NULL)
  RETURN VARCHAR2 AS

  -- V_KEY muss mindestens 128BIT d.h. 16Byte lang sein
  V_KEY              VARCHAR2(50) := '1234567890123456';
  V_VERSCHLUESSELT    VARCHAR2(50);
  LV_PASSWORD2ENCRYPT VARCHAR2(50);
  LN_8BYTE           NUMBER;
BEGIN
-- Überpruefen, ob p_key mindestens 16Byte lang ist
  IF length(NVL(P_KEY,V_KEY)) < 16 THEN
      LOG_HANDLING.ERROR(V_MODULE
                        ,V_SOURCE
                        ,V_IDENTIFIER
                        ,'Key zum Verschluesseln zu kurz! (Mindestens 16 Byte)'
                        );
  END IF;
-- Dass Passwort muss ein Vielfaches von 8Zeichen sein
  ln_8byte := 8 - MOD( length(P_PASSWORD2ENCRYPT), 8 );
  LV_PASSWORD2ENCRYPT := P_PASSWORD2ENCRYPT||RPAD(CHR(ln_8byte), ln_8byte, CHR(ln_8byte));

  V_VERSCHLUESSELT := DBMS_OBFUSCATION_TOOLKIT.DES3ENCRYPT(INPUT_STRING  => LV_PASSWORD2ENCRYPT, KEY_STRING => NVL(P_KEY,V_KEY) );
  --DBMS_OUTPUT.PUT_LINE('Verschluesselt: '||v_verschluesselt);

   RETURN v_verschluesselt;

   EXCEPTION
   WHEN OTHERS THEN
  RAISE;
END ENCRYPT;
```

UTL_DECRYPT Body 2/2

```
 FUNCTION DECRYPT(p_password2decrypt IN VARCHAR2, p_key IN VARCHAR2 DEFAULT NULL)  RETURN VARCHAR2 AS
V_KEY        varchar2(50) := '1234567890123456';
  V_KLARTEXT    VARCHAR2(50);
  V_LAENGE      NUMBER;

BEGIN
  -- Entschluesseln ...
  V_KLARTEXT := DBMS_OBFUSCATION_TOOLKIT.DES3DECRYPT(INPUT_STRING => P_PASSWORD2DECRYPT,
KEY_STRING => NVL(P_KEY,V_KEY) );

  --Klaartext um Verlängerung kürzen
  V_LAENGE := LENGTH(V_KLARTEXT);
  v_klartext := RPAD(V_KLARTEXT,V_LAENGE -    ASCII(SUBSTR(v_klartext,v_laenge)));
  RETURN v_klartext;
  EXCEPTION
   WHEN OTHERS THEN
  RAISE;
  END DECRYPT;
END UTL_DECRYPT;
```