Kampf den Jugendsünden – Ein Plädoyer für Code-Reviews

Tobias Wirtz
PAPSTAR GmbH

Über mich

Tobias Wirtz

Senior Software Engineer

- Fachinformatiker für Anwendungsentwicklung
- ORACLE seit 2001 *(8i)*, APEX seit 2014
- Ausbilder IT-Berufe (IHK-geprüft AEVO)

- Freiberuflicher Motorsportjournalist

tobias.wirtz@papstar.de
Twitter: **@tobiw_apex**

PAPSTAR GmbH

- Mittelständisches Handelsunternehmen
- Hauptsitz in Kall, europaweit 5 Niederlassungen
- Sortiment umfasst mehr als 7000 Artikel
- Zentrales Rechenzentrum in Kall
- Eigene Software-Entwicklungsabteilung
- IT-Ausbildungsbetrieb seit 2001

PAPSTAR

Agenda

PAPSTAR

Agenda

- Code-Reviews?
- PL/Scope
- Code-Beispiele (mit Best Practices)

PAPSTAR

Code-Reviews?

Was ist ein Code-Review?

- Manuelle Prüfung der Arbeitsergebnisse von Softwareentwicklung
- „Der Einsatz von Reviews führt zu einer deutlichen Reduktion von Fehlern.“

  Quelle: https://de.wikipedia.org/wiki/Review_(Softwaretest)
- Bandbreite: sehr formal bis vollständig informell

Was passiert beim Code-Review?

- Fachliche Prüfung
  - Schreibtischtest
  - Debugging

Was passiert beim Code-Review?

- Einhaltung von Entwicklungsstandards
  - Verwendung von Kommentaren
  - Einrückungen
  - Variablenbenennung
  - Variablendeklaration

Was passiert beim Code-Review?

- Code-Optimierung
  - Verwendung neuer Features
  - Ablösung alter Funktionalitäten
  - Performancetuning

Wann ist ein Code-Review sinnvoll?

- Produktivschaltung von neuem Code
  - Vier-Augen-Prinzip
  - Bei neuen/unerfahrenen Entwicklern: gemeinsamer Review
- Migration
  - Vorher: Alte Funktionalitäten umbauen
    - In neuer Version deprecated oder sogar desupported
  - Nachher: Neue Features nutzen
- Geänderte Entwicklungsstandards
- Nach festen Zeitintervallen

PAPSTAR

PL/Scope

PL/Scope

- ORACLE-Tool zur Code-Analyse
- Seit Version 11.1.0.7 verfügbar
  - Keine Installation notwendig
- Default: deaktiviert
  - Aktivierung mit ALTER SESSION
- Integration in SQL Developer

PL/Scope

- Metadaten für alle Bezeichner und SQLs
  - Generierung erfolgt beim Compile-Vorgang
- Abrufbar über Views
  - Analyse mit SQL-Statements
- Ideal zur Prüfung von Namenskonventionen

PL/Scope

```
SELECT *
   FROM PLSQL_REGELN;
```

TYP	DATENTYP	EXPRESSION
VARIABLE	DATE	\ADT
CONSTANT	NUMBER	\ACN
FORMAL IN	NUMBER	\AP_
VARIABLE	NUMBER	\AN
CONSTANT	VARCHAR2	\ACV
FORMAL IN	VARCHAR2	\AP_
VARIABLE	VARCHAR2	\AV

PL/Scope

- Nachteile von PL/Scope
  - Erhöhte Compilierzeit
  - Zusätzlicher Speicherbedarf
  - Keine Analysemöglichkeiten
    - Anonyme Blöcke
    - Dynamisches SQL

PAPSTAR

Code-Beispiele (mit Best Practices)

Code-Beispiele

```
1  CREATE OR REPLACE Function TW_SF_BENUTZER_NAME_DURCHWAHL
2  (
3   P_Benutzer             IN VARCHAR2
4  )
5  RETURN VARCHAR2
6  IS
7  -- ********************************************************************************************
8  -- Die Funktion gibt den Benutzernamen inklusive angehangener Telefonnummer zurück, sollte der
9  -- Benutzer eine Rufnummer im internen Telefonnetz von PAPSTAR Kall besitzen
10 -- ********************************************************************************************
11 --
12 -- Autor:    Tobias Wirtz
13 -- Erstellt: 12. April 2011
14 --
15 -- geändert:
16 --
17 -- ********************************************************************************************
18
19    vName                   VARCHAR2(50 CHAR);
20    vRufnummerPAPSTARKall   CONSTANT VARCHAR2(10 CHAR) := '+49 244183';
21
22 BEGIN
23
24    SELECT CASE WHEN SUBSTR(Telefon, 1, 10) = vRufnummerPAPSTARKall
25                THEN Name || ' (' || SUBSTR(Telefon, -3) || ')'
26                ELSE Name
27           END
28      INTO vName
29      FROM BENUTZER
30     WHERE Benutzer = P_Benutzer;
31
32    RETURN vName;
33
34    EXCEPTION
35       WHEN NO_DATA_FOUND THEN
36       BEGIN
37          RETURN P_Benutzer;
38       END;
39       WHEN OTHERS THEN
40       BEGIN
41          RETURN P_Benutzer;
42       END;
43 END;
```

Laut Regeln:

\ACV

Code-Beispiele

```
WITH Regeln as (SELECT *
                  FROM PLSQL_REGELN)
SELECT u.Zeile,
       u.Name,
       u.Typ,
       u.Datentyp,
       Regeln.Expression
  FROM (SELECT Name,
               Type Typ,
               Nachfolger Datentyp,
               Line Zeile
          FROM (SELECT ai.Name,
                       CASE WHEN LEAD(Usage) OVER(ORDER BY USAGE_ID) = 'REFERENCE'
                            THEN LEAD(Name) OVER(ORDER BY USAGE_ID)
                            ELSE NULL
                       END Nachfolger,
                       ai.Type,
                       ai.Usage,
                       ai.Usage_ID,
                       ai.Line,
                       ai.Col
                  FROM user_identifiers ai
                 WHERE Object_Name = 'TW_SF_BENUTZER_NAME_DURCHWAHL')
         WHERE Type IN (SELECT DISTINCT Typ FROM Regeln)
           AND Usage = 'DECLARATION'
       ) u
   LEFT OUTER JOIN Regeln ON Regeln.Typ = u.Typ
                         AND (NVL(Regeln.Datentyp, 'NIX') = NVL(u.Datentyp, 'NIX')
                            OR u.Datentyp IS NULL AND REGEXP_LIKE(u.Name, Regeln.Expression))
  WHERE NOT REGEXP_LIKE(u.Name, Regeln.Expression) OR Regeln.Expression IS NULL
```

ZEILE	NAME	TYP	DATENTYP	EXPRESSION
20	VRUFNUMMERPAPSTARKALL	CONSTANT	VARCHAR2	\ACV

Code-Beispiele

```
CREATE OR REPLACE TRIGGER TG_LIEFERANT_BU
BEFORE INSERT OR UPDATE ON LIEFERANT
FOR EACH ROW
DECLARE
    dt_datum    date;

BEGIN

    SELECT SYSDATE
      INTO dt_datum
      FROM DUAL;

    IF INSERTING THEN
        :new.Datum_Neuanlage    := dt_datum;
    ELSE
        :new.Datum_Aenderung    := dt_datum;
    END IF;

END;
```

Auf überflüssige Variablendeklarationen möglichst verzichten

Code-Beispiele

```
v_AKTION VARCHAR2(50);
v_MAIL_TEXT LONG;
v_MENGEN_NICHT_GEPLANT LONG;
v_EMAIL_EMPFAENGER VARCHAR2(50);
v_EMAIL_EMPFAENGER_CC1 VARCHAR2(50);
v_EMAIL_EMPFAENGER_CC2 VARCHAR2(50);
v_EMAIL_EMPFAENGER_CC3 VARCHAR2(50);
v_EMAIL_EMPFAENGER_CC1_CHECK NUMBER;
v_EMAIL_EMPFAENGER_CC2_CHECK NUMBER;
v_EMAIL_EMPFAENGER_CC3_CHECK NUMBER;
v_ARTIKEL_NICHT_GEPLANT_CHECK NUMBER;
v_SAISON_KZ NUMBER;


v_DISPLAYCHECK NUMBER;
v_ARTIKELSAISONCHECK NUMBER;
v_AUFTRAGNR_REAL_CHECK NUMBER;
v_MENGE_ANGELEGT_CHECK NUMBER;
v_ISTMENGEN_AUFTEILUNG_CHECK NUMBER;
v_ISTMENGE_CHECK NUMBER;
V_KOPF_ANGELEGT_CHECK NUMBER;
V_SAISON_CHECK NUMBER;
v_ANLAGE_CHECK NUMBER;
v_ISTMENGE_UEBERNAHME NUMBER;
v_DISPLAYNR NUMBER;
v_DISPLAY_COUNT_1 NUMBER;
v_DISPLAY_COUNT_2 NUMBER;

v_ISTMENGE_CALC NUMBER;
v_ISTMENGE_REST_CALC NUMBER;
v_ISTMENGE_EINTRAG_CALC NUMBER;
v_SOLLMENGE_CALC NUMBER;
v_ISTMENGE_LOOP_STOP NUMBER;


v_AUFTRAGSNUMMER NUMBER;   -- ADDITION + SOLLMENGE CHECK
v_ARTIKELNUMMER NUMBER;       -- ADDITION + SOLLMENGE CHECK
v_KUNDENNR NUMBER;
v_ZKONTONR NUMBER;
V_SAMKONTONR NUMBER;
v_SATZART NUMBER;
v_IKTONR NUMBER;
v_VKE NUMBER;
```

```
v_VKE NUMBER;
v_SEQ_NEUE_AUFTRAGSNR NUMBER;
v_LIEFERDATUM DATE;
v_LIEFERDATUM_CHECK NUMBER;

v_PLANER_AUFTRAGSNUMMER NUMBER;
v_PLANER_AUFTRAGSNUMMER_CHECK NUMBER;
v_PLANER_KUNDENNR NUMBER;

v_ARTIKELCHECK NUMBER;


TYPE t_DispoDaten IS REF CURSOR;
TYPE t_DisplayCheck IS REF CURSOR;
TYPE t_PlanerDaten IS REF CURSOR;
TYPE tIstmengenAufteilung IS REF CURSOR;
TYPE t_PlanerArtikelnr IS REF CURSOR;                -- SOLLMENGE CHECK
TYPE t_PlanerSollmengeCheck IS REF CURSOR;           -- SOLLMENGE CHECK

c_DISPO_DATEN t_DispoDaten;
c_PLANER_DATEN t_PlanerDaten;
c_ISTMENGEN_AUFTEILUNG tIstmengenAufteilung;
c_DISPLAY_CHECK t_DisplayCheck;
c_PLANER_ARTIKELNR t_PlanerArtikelnr;                --SOLLMENGE CHECK
c_PLANER_SOLLMENGECHECK t_PlanerSollmengeCheck;      --SOLLMENGE CHECK

v_ISTMENGE NUMBER;                              --SOLLMENGE CHECK
v_SOLLMENGE NUMBER;                              --SOLLMENGE CHECK
v_SOLLMENGEUEBERCHECK NUMBER;                         --SOLLMENGE CHECK
v_SOLLMENGEUEBER  LONG;                               --SOLLMENGE CHECK

v_EIGENMARKECHECK       NUMBER;

BEGIN
```

Code-Beispiele

```
INSERT INTO KUNDE_KOND_DETAIL_KOPF
    (
    Firma,
    KundenNr,
    Laufzeit_Von,
    Laufzeit_Bis,
    Kuendigungszeitpunkt,
    Benutzer_Neuanlage
    )
VALUES
(
    rec.Firma,
    rec.KundenNr,
    ADD_MONTHS(rec.Laufzeit_Von, 12),
    ADD_MONTHS(rec.Laufzeit_Bis, 12),
    ADD_MONTHS(rec.Kuendigungszeitpunkt, 12),
    P_Benutzer
);

SELECT ID
  INTO nID
  FROM KUNDE_KOND_DETAIL_KOPF
 WHERE Firma                 = rec.Firma
   AND KundenNr              = rec.KundenNr
   AND Laufzeit_Von          = rec.Laufzeit_Von
   AND Laufzeit_Bis          = rec.Laufzeit_Bis
   AND Kuendigungszeitpunkt  = rec.Kuendigungszeitpunkt
   AND Benutzer_Neuanlage    = P_Benutzer;
```

Code-Beispiele

```
INSERT INTO KUNDE_KOND_DETAIL_KOPF
     (
     Firma,
     KundenNr,
     Laufzeit_Von,
     Laufzeit_Bis,
     Kuendigungszeitpunkt,
     Benutzer_Neuanlage
     )
VALUES
(
     rec.Firma,
     rec.KundenNr,
     ADD_MONTHS(rec.Laufzeit_Von, 12),
     ADD_MONTHS(rec.Laufzeit_Bis, 12),
     ADD_MONTHS(rec.Kuendigungszeitpunkt, 12),
     P_Benutzer
)
RETURNING ID
     INTO nID;
```

Returning Clause ersetzt SQL zum Nachlesen

Code-Beispiele

```
CREATE OR REPLACE TRIGGER SYSADM.TG_CI_AUFTRAG_POS
FOR INSERT ON AUFTRAG_POS REFERENCING NEW AS NEW OLD AS OLD
COMPOUND TRIGGER
    ------------------------------------------------------------
    nIndex                      BINARY_INTEGER := 0;
    ------------------------------------------------------------
    TYPE t_Firma                IS TABLE OF AUFTRAG_POS.FIRMA%TYPE                  INDEX BY BINARY_INTEGER;
    nFirma                      t_Firma;
    ------------------------------------------------------------
    TYPE t_AuftragsNr           IS TABLE OF AUFTRAG_POS.AuftragsNr%TYPE             INDEX BY BINARY_INTEGER;
    nAuftragsNr                 t_AuftragsNr;
    ------------------------------------------------------------
    TYPE t_PosNr                IS TABLE OF AUFTRAG_POS.PosNr%TYPE                  INDEX BY BINARY_INTEGER;
    nPosNr                      t_PosNr;
    ------------------------------------------------------------
    TYPE t_PREIS_VK             IS TABLE OF AUFTRAG_POS.PREIS_VK%TYPE               INDEX BY BINARY_INTEGER;
    nPREIS_VK                   t_PREIS_VK;
    ------------------------------------------------------------
    TYPE t_PREIS_KENNZEICHEN    IS TABLE OF AUFTRAG_POS.PREIS_KENNZEICHEN%TYPE      INDEX BY BINARY_INTEGER;
    nPREIS_KENNZEICHEN          t_PREIS_KENNZEICHEN;
    ------------------------------------------------------------
    TYPE t_RABATT1              IS TABLE OF AUFTRAG_POS.RABATT1%TYPE                INDEX BY BINARY_INTEGER;
    nRABATT1                    t_RABATT1;
    ------------------------------------------------------------
    TYPE t_RABATT1_KENNZEICHEN  IS TABLE OF AUFTRAG_POS.RABATT1_KENNZEICHEN%TYPE    INDEX BY BINARY_INTEGER;
    nRABATT1_KENNZEICHEN        t_RABATT1_KENNZEICHEN;
    ------------------------------------------------------------
    TYPE t_RABATT2              IS TABLE OF AUFTRAG_POS.RABATT2%TYPE                INDEX BY BINARY_INTEGER;
    nRABATT2                    t_RABATT2;
    ------------------------------------------------------------
    TYPE t_RABATT2_KENNZEICHEN  IS TABLE OF AUFTRAG_POS.RABATT2_KENNZEICHEN%TYPE    INDEX BY BINARY_INTEGER;
    nRABATT2_KENNZEICHEN        t_RABATT2_KENNZEICHEN;
    ------------------------------------------------------------
    TYPE t_BENUTZER_NEUANLAGE   IS TABLE OF AUFTRAG_POS.BENUTZER_NEUANLAGE%TYPE    INDEX BY BINARY_INTEGER;
    vBENUTZER_NEUANLAGE         t_BENUTZER_NEUANLAGE;
    ------------------------------------------------------------
```

Code-Beispiele

```
else

    -- Wenn die Bestellmenge größer als ein Vielfaches vom Palettenfaktor ist,
    -- muss eine zusätzliche Position erzeugt werden
    --
    nIndex                          := nIndex + 1;

    nFirma(nIndex)                  := :new.Firma;
    nAuftragsNr(nIndex)             := :new.AuftragsNr;
    nPosNr(nIndex)                  := :new.PosNr;
    nPREIS_KENNZEICHEN(nIndex)      := :new.PREIS_KENNZEICHEN;

    nARTIKELNR(nIndex)              := :new.ARTIKELNR;
    vBEZEICHNUNG(nIndex)            := :new.BEZEICHNUNG;

    nRABATT1(nIndex)                := :new.RABATT1;
    nRABATT1_KENNZEICHEN(nIndex)    := :new.RABATT1_KENNZEICHEN;
    nRABATT2(nIndex)                := :new.RABATT2;
    nRABATT2_KENNZEICHEN(nIndex)    := :new.RABATT2_KENNZEICHEN;
    vBENUTZER_NEUANLAGE(nIndex)     := :new.BENUTZER_NEUANLAGE;
```

Code-Beispiele

```
CREATE OR REPLACE TRIGGER SYSADM.TG_CI_AUFTRAG_POS
FOR INSERT ON AUFTRAG_POS REFERENCING NEW AS NEW OLD AS OLD
COMPOUND TRIGGER
    ------------------------------------------------------------
    nIndex                      BINARY_INTEGER := 0;
    ------------------------------------------------------------
    TYPE t_AUFTRAG_POS          IS TABLE OF AUFTRAG_POS%ROWTYPE          INDEX BY BINARY_INTEGER;
    rAUFTRAG_POS                t_AUFTRAG_POS;
    ------------------------------------------------------------
```

```
        else

            -- Wenn die Bestellmenge größer als ein Vielfaches vom Palettenfaktor ist,
            -- muss eine zusätzliche Position erzeugt werden
            --
            nIndex                                  := nIndex + 1;

            rAUFTRAG_POS(nIndex).Firma              := :new.Firma;
            rAUFTRAG_POS(nIndex).AuftragsNr         := :new.AuftragsNr;
            rAUFTRAG_POS(nIndex).PosNr              := :new.PosNr;
            rAUFTRAG_POS(nIndex).Preis_Kennzeichen  := :new.PREIS_KENNZEICHEN;

            rAUFTRAG_POS(nIndex).ArtikelNr          := :new.ARTIKELNR;
            rAUFTRAG_POS(nIndex).Bezeichnung        := :new.BEZEICHNUNG;

            rAUFTRAG_POS(nIndex).RABATT1            := :new.RABATT1;
            rAUFTRAG_POS(nIndex).RABATT1_KENNZEICHEN:= :new.RABATT1_KENNZEICHEN;
            rAUFTRAG_POS(nIndex).RABATT2            := :new.RABATT2;
            rAUFTRAG_POS(nIndex).RABATT2_KENNZEICHEN:= :new.RABATT2_KENNZEICHEN;
            rAUFTRAG_POS(nIndex).BENUTZER_NEUANLAGE := :new.BENUTZER_NEUANLAGE;
```

Code-Beispiele

```
-- Auftrag in DESADV_AUFTRAGS_KOPF protokollieren
-- !! NICHT im Testbetrieb

v_Fehler := -20200;
if P_Test != 'Ja'
then
    UPDATE desadv_auftrags_kopf
                SET  Tour            = rec_AUFTRAG.tour_nr,
                     Tour_Datum      = rec_AUFTRAG.tour_datum,
                   Liefer_Datum    = rec_AUFTRAG.lieferdatum,
                   Desadv_Datum    = sysdate
             WHERE AuftragsNr = rec_AUFTRAG.auftragsnr
             AND   Firma = 02;
    IF SQL%ROWCOUNT = 0
    THEN
       INSERT INTO desadv_auftrags_kopf
       (
             Firma,
             AuftragsNr,
             Tour,
             Tour_Datum,
             Liefer_Datum,
             Desadv_Datum
       )
       VALUES
       (
              02,
             rec_AUFTRAG.auftragsnr,
             rec_AUFTRAG.tour_nr,
             rec_AUFTRAG.tour_datum,
             rec_AUFTRAG.lieferdatum,
             sysdate
       );
    END IF;
```

Code-Beispiele

```
-- Auftrag in DESADV_AUFTRAGS_KOPF protokollieren
-- !! NICHT im Testbetrieb

v_Fehler := -20200;
if P_Test != 'Ja'
then

    MERGE INTO desadv_auftrags_kopf z
    USING (SELECT rec_AUFTRAG.Tour_nr       as Tour,
                  rec_AUFTRAG.Tour_Datum    as Tour_Datum,
                  rec_AUFTRAG.Lieferdatum   as Lieferdatum,
                  rec_AUFTRAG.AuftragsNr    as AuftragsNr,
                  2                         as Firma
             FROM DUAL) q
    ON (q.Firma         = z.Firma
    AND q.AuftragsNr    = z.AuftragsNr)
    WHEN MATCHED THEN
    UPDATE
       SET Tour             = q.Tour,
           Tour_Datum       = q.Tour_Datum,
           Liefer_Datum     = q.Lieferdatum,
           Desadv_Datum     = SYSDATE
    WHEN NOT MATCHED THEN
    INSERT
    (   Firma,
        AuftragsNr,
        Tour,
        Tour_Datum,
        Liefer_Datum,
        Desadv_Datum
    )
    VALUES
    (
        q.Firma,
        q.AuftragsNr,
        q.Tour,
        q.Tour_Datum,
        q.Lieferdatum,
        SYSDATE
    );
```

Merge statt getrenntem Insert / Update

Code-Beispiele

```
    v_Langbezeichnung           VARCHAR2(200 BYTE);
    v_Zustaendigkeit_Einkauf    VARCHAR2(50 CHAR);
    v_Zustaendigkeit_Marketing  VARCHAR2(50 CHAR);
    v_Email                     EMAIL.Empfaenger%TYPE;
    v_Email_CC1                 EMAIL.CarbonCopy1%TYPE;
    v_Email_CC3                 EMAIL.CarbonCopy3%TYPE;
    v_Email_CC4                 EMAIL.CarbonCopy4%TYPE;
    v_Email_CC5                 EMAIL.CarbonCopy5%TYPE;
    v_Email_CC6                 EMAIL.CarbonCopy6%TYPE;
    v_MailBetreff               VARCHAR2(200 CHAR);
    v_MailText                  VARCHAR2(200 CHAR);
    v_Exist                     NUMBER;
    n_Artikelstatus_Alt         NUMBER;

begin

    -- notwendige Artikel-Infos holen
    v_Exist := 1;
    BEGIN
        SELECT Langbezeichnung,
               DECODE(Zustaendigkeit_Einkauf,     '0', 'DUMMY',   NVL(TRIM(Zustaendigkeit_Einkauf),    'DUMMY')),
               DECODE(Zustaendigkeit_Marketing,   '0', 'DUMMY',   NVL(TRIM(Zustaendigkeit_Marketing), 'DUMMY')),
               Artikelstatus
          INTO v_Langbezeichnung,
               v_Zustaendigkeit_Einkauf,
               v_Zustaendigkeit_Marketing,
               n_Artikelstatus_Alt
          FROM ARTIKEL
          WHERE ArtikelNr  = P_ArtikelNr
            AND Firma      = 2
            AND ((Artikelstatus = 1 AND P_Artikelstatus_Neu = 2)
             OR  (Artikelstatus = 2 AND P_Artikelstatus_Neu = 1));
```

Code-Beispiele

ARTIKEL: Created: 24/03/2019 10:46:02 Last DDL: 28/04/2019 10:00:21

Primary Key: FIRMA, ARTIKELNR

Columns | Indexes | Constraints | Triggers | Data | Script | Grants | Synonyms | Partitions | Subpartitions | Stats/Size | Referential | Used By | Auditing

Column Name	I.	PK	Index Pos	Null?	Data Type	Default	Def. On Null	Histogram	Num Distinct
FIRMA	1	1	2, 1, 1, 1, 1, 1, 1, 1, 1,	N	NUMBER (2)		☐	None	19
ARTIKELNR	2	2	1, 2, 3	N	NUMBER (8)		☐	None	39790
KURZBEZEICHNUNG	3			Y	VARCHAR2 (50 Char)		☐	None	9795
LANGBEZEICHNUNG	4		2	Y	VARCHAR2 (250 Char)		☐	None	150524
WARENGRUPPE	5		2	Y	NUMBER (7)		☐	None	711

Code-Beispiele

```
    v_Langbezeichnung           ARTIKEL.Langbezeichnung%TYPE;
    v_Zustaendigkeit_Einkauf    ARTIKEL.Zustaendigkeit_Einkauf%TYPE;
    v_Zustaendigkeit_Marketing  ARTIKEL.Zustaendigkeit_Marketing%TYPE;
    v_Email                     EMAIL.Empfaenger%TYPE;
    v_Email_CC1                 EMAIL.CarbonCopy1%TYPE;
    v_Email_CC3                 EMAIL.CarbonCopy3%TYPE;
    v_Email_CC4                 EMAIL.CarbonCopy4%TYPE;
    v_Email_CC5                 EMAIL.CarbonCopy5%TYPE;
    v_Email_CC6                 EMAIL.CarbonCopy6%TYPE;
    v_MailBetreff               VARCHAR2(200 CHAR);
    v_MailText                  VARCHAR2(200 CHAR);
    v_Exist                     NUMBER;
    n_Artikelstatus_Alt         NUMBER;

begin

    -- notwendige Artikel-Infos holen
    v_Exist := 1;
    BEGIN
        SELECT Langbezeichnung,
               DECODE(Zustaendigkeit_Einkauf,     '0', 'DUMMY',   NVL(TRIM(Zustaendigkeit_Einkauf),    'DUMMY')),
               DECODE(Zustaendigkeit_Marketing,   '0', 'DUMMY',   NVL(TRIM(Zustaendigkeit_Marketing), 'DUMMY')),
               Artikelstatus
          INTO v_Langbezeichnung,
               v_Zustaendigkeit_Einkauf,
               v_Zustaendigkeit_Marketing,
               n_Artikelstatus_Alt
          FROM ARTIKEL
         WHERE ArtikelNr   = P_ArtikelNr
           AND Filiale     = P_Filiale
           AND Artikelstatus <> P_Artikelstatus_Neu
           AND (Artikelstatus = 2 AND P_Artikelstatus_Neu = 1);
```

Niemals VARCHAR2(BYTE) bei Unicode verwenden

SELECT … INTO von Tabellenspalten in Variablen mit statischen Definitionen dringend vermeiden

Code-Beispiele

```
SELECT mo.Jahr,
       mo.Monat,
       ab.stunden,
       mo.servicenr,
       mo.Vorgaenger,
       mo.kundenNr,
       mo.firma,
       abgeschlossen_JN,
       NVL (
           NVL (mo.Stundenlohn, st.StundenLohn),
           NVL (
               (SELECT VALUE
                  FROM PARAMETER
                 WHERE     section = 'ServiceAbrechnung'
                       AND parameter = 'Stundensatz'
                       AND benutzer = 'Alle'),
               9.19))
           Stundenlohn
```

Code-Beispiele

```
SELECT mo.Jahr,
       mo.Monat,
       ab.stunden,
       mo.servicenr,
       mo.Vorgaenger,
       mo.kundenNr,
       mo.firma,
       abgeschlossen_JN,
       COALESCE(mo.Stundenlohn,
                st.StundenLohn,
                (SELECT VALUE
                   FROM PS_PARAMETER
                  WHERE section     = 'ServiceAbrechnung'
                    AND parameter   = 'Stundensatz'
                    AND benutzer    = 'Alle'),
                9.19)
         Stundenlohn
```

COALESCE anstatt NVL-Verschachtelungen

PAPSTAR

Fragen ?

Quellen / Weitergehende Informationen

- DOAG Red Stack Magazin, September 2017
  - Sabine Heimsath: Schöner Coden – PL/SQL analysieren mit PL/Scope

https://www.its-people.de/files/admin/Artikel/sabine-heimsath-schoener-coden-plsql-analysieren-mit-plscope_2.pdf

WWW · PAPSTAR · COM