Versionsvergleich in APEX

APEXconnect, Juni 2015

Sabine Heimsath
its-people GmbH

Sabine Heimsath
Senior Consultant
its-people GmbH

Mail: sabine.heimsath@its-people.de
Twitter: @flederbine

Hinweise

Der folgende Text bezieht sich auf APEX 4.2.6
Für Apex 5.0 dürfte er in weiten Teilen ebenfalls zutreffen, dies ist aber im Einzelfall zu prüfen.

Motivation

Immer wieder kommt es vor, dass zwei Apex-Applikationen verglichen werden müssen, um die Unterschiede zwischen beiden herauszufinden.

Der bisherige in Apex mitgelieferte Vergleich innerhalb der APEX-Entwicklungsoberfläche (unter Cross Application Reports) taugt dafür nur bedingt, da er die ‚Signatur' der Komponenten vergleicht. Bei der Signatur (= Spalte COMPONENT_SIGNATURE) handelt es sich um einen String, der sich aus einer Auswahl Attributen (teilweise verkürzt) für je nach Objekttyp zusammensetzt und dadurch nicht immer alle wesentlichen Änderungen enthält.

Die Idee:
APEX-Metadaten nutzen

Da der Textvergleich der Exportfiles[1] nicht ohne Tücken ist, ergab sich die Idee, die Metadaten direkt in der Datenbank zu vergleichen und das Ergebnis benutzerfreundlich in einer Apex-Anwendung anzuzeigen.

Die Daten stehen dem User PUBLIC in Form von Views zur Verfügung:
Schema: APEX_040200
Präfix: APEX_

Als Einstiegspunkt sollte man die View APEX_DICTIONARY wählen, denn hier werden alle APEX-Views, die zugehörige Spalten UND jeweils eine Erläuterung angezeigt.
Will man nur die View sehen, setzt man den Filter auf COMMENT_TYPE = ‚View'

Empfehlung:
Sehen Sie sich auch das SQL der View APEX_DICTIONARY an.

Im Application Builder findet man das Dictionary rechts im Kasten ‚Tasks' unter dem Link ‚Application Express Views'. Hier bietet sich dann das ‚Report' Format an, da es im Gegensatz zum ‚Icon' Format auch die Beschreibungen der Views anzeigt.

Die Namen der Views sind normalerweise sehr sprechend. Teilweise werden Abkürzungen verwendet wie z. B.

BC – BREADCRUMBS
DA – DYNAMIC ACTIONS
SUPP – SUPPORTING
TEMP –TEMPLATES
VAL – VALIDATIONS

Diese und alle weiteren Abkürzungen sind den Beschreibungen (Comments) der Apex-Views im Dictionary erklärt.

[1] siehe Vortrag ‚Auf der Suche nach dem kleinen Unterschied: Versionsvergleich für APEX-Anwendungen' von Sabine Heimsath auf der DOAG 2013

Rechte

Normalerweise funktioniert der Versionsvergleich, wenn die Applikation im gleichen Workspace wie die zu vergleichenden Applikationen liegt.

Auf einem Spielsystem kann man seinem APEX_COMP-User die Rechte für die kompletten Metadaten über

```
grant APEX_ADMINISTRATOR_ROLE to apex_comp
```

geben. Damit kann man natürlich auch viel kaputt machen, daher Vorsicht!

Unterbau: Compare-Views

Nicht alle Views im Dictionary werden für den Applikationsvergleich benötigt. Wir können zum Beispiel alle Views ausklammern, die sich in der Objekt-Hierarchie oberhalb der Applikationen befinden (z. B. alle, die sich auf Workspaces beziehen) oder sich komplett außerhalb der Hierarchie befinden (z. B. die, die zum Team Development gehören).
Grob gesagt: Im Fokus sind alle Views, die mit APEX_APPL beginnen.

Für die Generierung der Compare-Views werden unterschiedliche Informationen benötigt. Diese werden in einer Tabelle namens DEF_DELTA_REPORT abgelegt. Sie enthält Schema und Namen der Quelltabellen, die Information, welche Spalten ignoriert werden sollen, nach welchen Kriterien die Daten gruppiert werden sollen und wie die Apex-Views gejoint werden sollen.

Die Tabelle DEF_PARAM enthält die Kombination zweier Applikations-IDs und eine generierte COMP_ID, über die der Vergleich an verschiedenen Stellen referenziert wird.

Um eine Compare-View für ein Objekt zu bauen, selektieren wir einmal alle Attribute für Applikation A und einmal alle Attribute für Applikation B. Der Join erfolgt über den jeweiligen Schlüssel, der ein Objekt eindeutig beschreibt.

Die folgende Liste der Join-Bedingungen ist nicht vollständig und auch nicht fix, denn es gibt bei manchen Objekten mehrere Möglichkeiten, ihre Identität zu definieren. Am häufigsten bietet sich der Name an, aber manchmal auch die Sequence oder im Fall der Page die ID, da der Benutzer selbst Einfluss auf diese ID hat.

APEX_APPLICATION_LOV_ENTRIES	a.LIST_OF_VALUES_NAME = b.LIST_OF_VALUES_NAME and a.DISPLAY_SEQUENCE = b.DISPLAY_SEQUENCE
APEX_APPLICATION_PAGE_BUTTONS	a.PAGE_ID = b.PAGE_ID and a.BUTTON_NAME = b.BUTTON_NAME
APEX_APPLICATION_PAGES	a.PAGE_ID = b.PAGE_ID
APEX_APPLICATION_PAGE_ITEMS	a.PAGE_ID = b.PAGE_ID and a.ITEM_NAME = b.ITEM_NAME
APEX_APPLICATION_COMPUTATIONS	a.COMPUTATION_ITEM = b.COMPUTATION_ITEM and a.COMPUTATION_SEQUENCE = b.COMPUTATION_SEQUENCE

APEX_APPLICATION_ITEMS	a.ITEM_NAME = b.ITEM_NAME
APEX_APPLICATION_PROCESSES	a.PROCESS_NAME = b.PROCESS_NAME
APEX_APPLICATION_SHORTCUTS	a.SHORTCUT_NAME = B.SHORTCUT_NAME
APEX_APPLICATION_AUTH	a.AUTHENTICATION_SCHEME_NAME = b.AUTHENTICATION_SCHEME_NAME
APEX_APPLICATION_LISTS	a.LIST_NAME = b.LIST_NAME
APEX_APPLICATION_LIST_ENTRIES	a.LIST_NAME = b.LIST_NAME and a.ENTRY_TEXT = b.ENTRY_TEXT
APEX_APPLICATION_LOVS	a.LIST_OF_VALUES_NAME = b.LIST_OF_VALUES_NAME
APEX_APPLICATION_NAV_BAR	a.DISPLAY_SEQUENCE = b.DISPLAY_SEQUENCE
APEX_APPLICATION_PAGE_BRANCHES	a.PAGE_ID = b.PAGE_ID and a.BRANCH_POINT = b.BRANCH_POINT and a.PROCESS_SEQUENCE = b.PROCESS_SEQUENCE
APEX_APPLICATION_PAGE_COMP	a.PAGE_ID = b.PAGE_ID and a.ITEM_NAME = b.ITEM_NAME and a.EXECUTION_SEQUENCE = b.EXECUTION_SEQUENCE
APEX_APPLICATION_PAGE_DB_ITEMS	a.PAGE_ID = b.PAGE_ID and a.ITEM_NAME = b.ITEM_NAME
APEX_APPLICATION_PAGE_PROC	a.PAGE_ID = b.PAGE_ID and a.PROCESS_NAME = b.PROCESS_NAME
APEX_APPLICATION_PAGE_REGIONS	a.PAGE_ID = b.PAGE_ID and a.REGION_NAME = b.REGION_NAME

Die Views werden alle nach demselben Schema aufgebaut:

```
create or replace view #VIEW_NAME# as
with param as (
  select p.APP_ID1, p.APP_ID2, p.COMP_ID
  from def_param p),
eins as (
  select p.comp_id, #SORT# ord, #COL_LIST#
  from #TABELLE1# e, param p
  where application_id = p.app_id1),
zwei as (
  select p.comp_id, #SORT# ord, #COL_LIST#
  from #TABELLE2# z, param p
  where application_id = p.app_id2)
select nvl(a.comp_id,b.comp_id) comp_id
     , #PRE_COL_LIST_A#
     , #PRE_COL_LIST_B#
from eins a
full outer join zwei b
  on a.comp_id = b.comp_id
  and #JOINCLAUSE#
order by 2, 1
```

Die Platzhalter werden aus diversen Quellen mit den entsprechenden Daten aus der Tabelle DEF_DELTA_REPORT gefüllt, wobei die Funktion LISTAGG_CLOB von Carsten Czarski[2] gute Dienste leistet, denn die konkatenierten Spaltennamen kommen bei weitem nicht mit 4000 Zeichen aus!

Beispiel für Variablen für die View auf Tabelle APEX_APPLICATION_PAGES:

#VIEW_NAME#	C_APEX_APPLICATION_PAGES (die resultierende View)
#SORT#	PAGE_ID (das Sortierungs und Gruppierungskriterium)
#TABELLE1#,#TABELLE2#	APEX_APPLICATION_PAGES (die Datenquelle)
#COL_LIST#	Sämtliche Spalten der Tabelle, bis auf unterdrückte Spalten (kann in Definitionstabelle hinterlegt werden)
#PRE_COL_LIST_A#, #PRE_COL_LIST_B#	Die Spaltennamen, dieses Mal mit Präfix A_ oder B_ für die beiden zu vergleichenden Applikationen A und B
#JOINCLAUSE#	a.PAGE_ID = b.PAGE_ID (Zusätzliche Join-Kriterien, wie in der obigen Tabelle)

Für eine spätere Erweiterung der Anwendung kann man analog zu den Compare-Views auch Views erzeugen, die ein Minus in beide Richtungen auf die zusammengehörigen Zeilen machen. (Im weiteren Diff-Views genannt)

Hierauf lassen sich dann auch Statistik-Views aufsetzen, die Auskunft über Art und Anzahl der Differenzen geben. Hier müssen in der #COL_LIST# natürlich die Applikations-ID ausgeschlossen werden, da sonst jede Zeile eine Differenz aufweisen würde.

Das Grundgerüst für jede View sieht so aus:

```
create or replace view #VIEW_NAME# as
with
param as (
  select APP_ID1, APP_ID2, OFFSET_DIFF, COMP_ID
  from def_param),
eins as (
  select p.COMP_ID, #SORT# ord, #COL_LIST#
  from #TABELLE1# e, param p
  where p.app_id1 = e.application_id
  #WHERE1#),
zwei as (
  select p.COMP_ID, #SORT# ord, #COL_LIST#
  from #TABELLE2# z, param p
  where p.app_id2 = z.application_id
  #WHERE2#)
select 'a' quelle, a.* from eins a
minus
select 'a' quelle, b.* from zwei b
union all
```

[2] http://sql-plsql-de.blogspot.de/2014/01/sql-listagg-mit-clob-ausgabe-kein.html

```
select 'b' quelle, b.* from zwei b
minus
select 'b' quelle, a.* from eins a
order by 2, 1
```

Welche Werte sich geändert haben, lässt sich mit der folgenden View feststellen, die die vorherige View referenziert. #BASEVIEW_NAME# ist hierbei der Name der vorherigen View.

```
CREATE OR REPLACE VIEW #VIEW_NAME#
AS
WITH eins AS
( SELECT * FROM #BASEVIEW_NAME# WHERE quelle = ,a'
),
zwei AS
( SELECT * FROM #BASEVIEW_NAME# WHERE quelle = 'b'
)
SELECT a.COMP_ID, a."ORD",
#COL_LIST#
FROM eins a
INNER JOIN zwei b
ON a."ORD" = b."ORD"
AND a.COMP_ID = b.COMP_ID
```

Mittelschicht: Die Zugriffsfunktionen

Die Compare-Views beinhalten alle Informationen, die wir haben wollen, aber in einem sehr unleserlichen Format. Es ist schon im SQL Developer lästig, kilometerweit nach rechts zu scrollen, um die relevanten Unterschiede zu finden, aber noch lästiger ist dies einem sehr breiten Apex-Report.

Sehr viel übersichtlicher wird es, wenn die zu vergleichenden Werte nebeneinander dargestellt werden, in einer Zeile pro Attribut:

	Applikation A	Applikation B
Attributname 1	XY	XX
Attributname 2	1	2
…	…	…

Wie bekommt man die Daten in dieses Format?
Diverse Ansätze mit PIVOT, UNPIVOT, DECODE erwiesen sich bei dieser großen Anzahl von Spalten als ungünstig.
Ein Verfahren, das die Daten erst auf der Oberfläche mit jQuery dreht, ist denkbar, wird aber wieder unschön komplex, wenn man sich eine spätere Darstellung als Baum mit verschachtelten Objekten wünscht.
Vielleicht gäbe es noch eine Möglichkeit, die gewünschte Darstellung mit einem Apex Template zu erreichen. Für Hinweise wäre ich dankbar!

In diesem Fall fiel die Entscheidung auf Pipelined Table Functions[3], die eine Menge von Attribut-Wert-Wert-Tupeln zurückgeben.

Die Definition des Tupels und des zugehörigen Tabellentyps:

```
create or replace TYPE t_av2_row AS OBJECT (
  attr VARCHAR2(4000),
  val1  VARCHAR2(4000),
  val2  VARCHAR2(4000),
  diff  VARCHAR2(1),
  ord_id number
);

create or replace TYPE t_av2_tab IS TABLE OF t_av2_row;
```

Der Aufbau einer Funktion ist simpel, der Aufwand entsteht einmalig bei Generierung der Funktionen.

Ablauf:
Wird die Funktion aufgerufen, wird eine Zeile aus der Compare-View in einen Record eingelesen. Durch den Join in der View enthält sie die Daten des Objekts aus Applikation A und Applikation B. Dann wird der Reihe nach *jeder* Attributname der View mit dem jeweiligen Wert aus Applikation A und Applikation B zurückgegeben.
Die Werte befinden sich in v_werte.a_<attributname> und v_wert.b_<attributname>.

Weiterhin wird festgestellt, ob sich die Werte unterscheiden oder nicht; entsprechend ‚Y' oder ‚N' zurückgegeben.

Schließlich wird noch die COLUMN_ID der Original-Tabelle angehängt, damit der Entwickler die Attribute in der Reihenfolge sortieren kann, in der sie auch in der Original-View angezeigt werden.

Hier ein Beispiel für das Objekt *Page*:

```
FUNCTION all_application_pages (p_comp_id number,
                                p_page_id1 number,
                                p_page_id2 number)
                           RETURN t_av2_tab PIPELINED  AS

  v_werte  c_apex_application_pages%ROWTYPE;

begin

  select * into v_werte
  from c_apex_application_pages
  where comp_id = p_comp_id
  and (nvl(a_page_id,-1) = nvl(p_page_id1,-1))
  and (nvl(b_page_id,-1) = nvl(p_page_id2,-1));
```

[3] http://docs.oracle.com/cd/B28359_01/appdev.111/b28370/tuning.htm#LNPLS01210

```
  pipe row(t_av2_row('PAGE_ID',
                     v_werte.a_PAGE_ID,
                     v_werte.b_PAGE_ID,
                     case when
                         v_werte.a_PAGE_ID = v_werte.b_PAGE_ID
                     then 'N'
                         when v_werte.a_PAGE_ID is null
                          and v_werte.b_PAGE_ID is null
                     then 'N'
                     else 'Y'
                     end,
                     5));  -- COLUMN_ID = Spaltenreihenfolge
  -- usw.
```

Die Generierung der vielen PIPE-ROW-Zeilen wird durch folgendes Statement unterstützt:

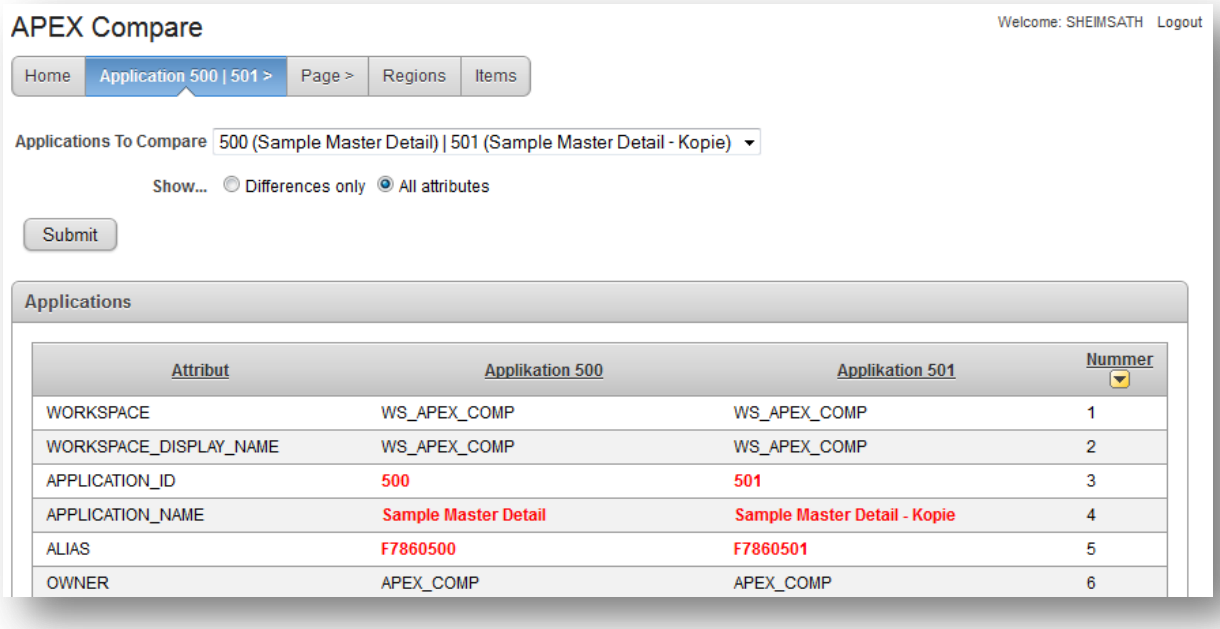
```
select
  case
    when column_name in
        ('APPLICATION_GROUP_ID','AUTHENTICATION_SCHEME_ID',
        'AUTHORIZATION_SCHEME_ID','AUTHORIZATION_SCHEME_ID',
        'PARENT_REGION_ID','TEMPLATE_ID',
        'BREADCRUMB_TEMPLATE_ID','AUTHORIZATION_SCHEME_ID',
        'REPORT_TEMPLATE_ID','REGION_ID',
        'ITEM_LABEL_TEMPLATE_ID','AUTHORIZATION_SCHEME_ID')
    then '--'
    else '  '
  end
  || ' pipe row(t_av2_row('''
  || column_name                  -- Spaltenname in der View
  || ''', v_werte.a_'             -- Wert aus Applikation A
  || substr(column_name,1,28)
  || ', v_werte.b_'               -- Wert aus Applikation B
  || substr(column_name,1,28)
  || ', case when v_werte.a_'     -- Differenz Y/N
  || substr(column_name,1,28)
  || ' = v_werte.b_'
  || substr(column_name,1,28)
  || ' then ''N'' when v_werte.a_'
  || substr(column_name,1,28)
  || ' is null and v_werte.b_'
  || substr(column_name,1,28)
  || ' is null then ''N'' else ''Y'' end, '
  || column_id                    -- Spaltenreihenfolge
  || '));'
from all_tab_cols
where owner    = 'APEX_040200'
and table_name = 'APEX_ APPLICATION_PAGES'
order by column_id;
```

Basis für die Generierung ist die View-Spaltenbeschreibung in der Data-Dictionary-Tabelle ALL_TAB_COLS.

Das erste CASE sorgt dafür, dass IDs, die in der Apex-View bereits zum Objektnamen aufgelöst wurden, unterdrückt werden, indem es Kommentarzeichen vor die Zeile setzt.
Damit kann der Entwickler die ID bei Bedarf schnell wieder einbinden.

Ganz oben:
Die APEX-Applikation

Die Applikation besteht aus einer Reihe von Reports für die einzelnen Objekte, angefangen bei der Applikation:



Wenn der Vergleich für die beiden Applikationen A und B angelegt und ausgewählt ist, kann man über den Aufruf der Funktion die Attribute vergleichen.

Das unterliegende Report-Statement:

```
select   attr ,
  val1 ,
  val2  ,
  decode(diff,'Y','color:red;font-weight:bold;','color:black;')
diff ,
  ord_id
from
table(pkg_diff.all_application(:P20_APPLICATIONS_TO_COMPARE))
where (:P20_DIFFS_ONLY = 0 or diff = 'Y')
```

Über das Page-Item P20_DIFFS_ONLY werden die identischen Werte ein- oder ausgeblendet.

Über die DIFF-Spalte wird die Darstellung der Werte gesteuert, damit dem Benutzer die abweichenden Werte direkt ins Auge fallen. Für die Darstellung müssen für die Spalte VAL1 und VAL2 analog folgende Werte gesetzt werden:

Spalte VAL1:

Column Heading:	Applikation &P20_APP_ID1.
HTML Expression:	#VAL1#

Erweiterung: Labels und Hilfetexte

Die Attributnamen sind leider nicht immer sprechend. Deshalb wäre es schön, die Bezeichnungen, die der Entwickler vom Apex Builder her kennen, zusätzlich einzublenden.[4]

Eine Apex-Page wird auf der Apex-Builder-Page 4301 angelegt. Mit dieser Infomation kann man in der Tabelle APEX_APPLICATION_PAGE_DB_ITEMS die passenden Texte selektieren und mit der Tabelle USER_TAB_COLS (table_name = 'APEX_ APPLICATION_PAGES') joinen.

Das funktioniert bei 21 Items (immerhin!), aber es bleiben auch 25 DB_ITEMS übrig, die den View-Spalten nicht zugeordnet werden können.
Warum ist das so?
Ein Blick in die View-Definition zeigt, dass viele Spalten zwecks besserer Lesbarkeit umbenannt wurden. Das kann unser JOIN leider nicht berücksichtigen:

```
CREATE OR REPLACE FORCE VIEW "APEX_040200"."APEX_APPLICATION_PAGES"
select
  w.short_name                              workspace,
  w.display_name                            workspace_display_name,
  p.flow_id                                 application_id,
  f.name                                     application_name,
  --
  p.id                                      page_id,
  p.name                                     page_name,
  p.STEP_TITLE                              page_title,
  p.user_interface_id,
  p.media_type                              media_type,
  p.TAB_SET                                 tab_set,
  p.ALIAS                                    page_alias,
  decode(substr(p.PAGE_COMPONENT_MAP,1,2),
        '01','Tabular Form',
        '02','Form',
        '03','Report',
        '04','Chart',
        '05','Web Service',
        '06','Navigation Page',
        '07','Tree',
        '08','Calendar'
```

[4] Dies funktioniert nur, wenn der APEX_COMP-User zum Generierungszeitpunkt die Rechte auf die Metadaten des Apex Builders hat. Siehe Überschrift ‚Rechte'.

Trotzdem probieren wir es einmal aus und definieren einen erweiterten Objekttyp:

```
create or replace TYPE T_AV2LH_ROW AS OBJECT (
  attr VARCHAR2(4000),
  val1  VARCHAR2(4000),
  val2  VARCHAR2(4000),
  diff  VARCHAR2(1),
  ord_id number,
  label VARCHAR2(4000),
  help VARCHAR2(4000)
);

create or replace TYPE T_AV2LH_TAB IS TABLE OF T_AV2LH_ROW;
```

Die Generierung der PIPE-ROW-Zeilen wird dadurch ein wenig komplexer:

```
with labels_and_help as
  (
    select db_column_name
    , nvl2(item_label,'q''{'
      || rtrim(item_label,':')
      || '}''','null') item_label
    , nvl2(help_text,
      'q''{'
      ||replace(
         replace(
          replace(help_text,chr(10),'')
        ,chr(13),'')
       ,'&', '&'' || ''')
      || '}''','null') help_text
    from apex_application_page_db_items
    where application_id = 4000 -- APEX Builder
    and page_id          = 4301 -- Das ist die Page, auf der
                                -- eine Page definiert wird
  )
select
  case
    when column_name in
       ('APPLICATION_GROUP_ID','AUTHENTICATION_SCHEME_ID',
        'AUTHORIZATION_SCHEME_ID','AUTHORIZATION_SCHEME_ID',
        'PARENT_REGION_ID','TEMPLATE_ID',
        'BREADCRUMB_TEMPLATE_ID','AUTHORIZATION_SCHEME_ID',
        'REPORT_TEMPLATE_ID','REGION_ID',
        'ITEM_LABEL_TEMPLATE_ID','AUTHORIZATION_SCHEME_ID')
    then '--'
    else '  '
  end
  || ' pipe row(t_av2lh_row('''
  || column_name
  || ''', v_werte.a_'
```

```
  || substr(column_name,1,28)
  || ', v_werte.b_'
  || substr(column_name,1,28)
  || ', case when v_werte.a_'
  || substr(column_name,1,28)
  || ' = v_werte.b_'
  || substr(column_name,1,28)
  || ' then ''N'' when v_werte.a_'
  || substr(column_name,1,28)
  || ' is null and v_werte.b_'
  || substr(column_name,1,28)
  || ' is null then ''N'' else ''Y'' end, '
  || column_id
  || ', '
  || nvl(lh.item_label,'null')                                  -- neu
  ||', '
  || nvl(lh.help_text,'null')                                   -- neu
  ||'));'
from all_tab_cols atc
left outer join labels_and_help lh
on lh.db_column_name = atc.column_name
where owner          = 'APEX_040200'
and table_name       = 'APEX_APPLICATION_PAGES'
order by column_id;
```

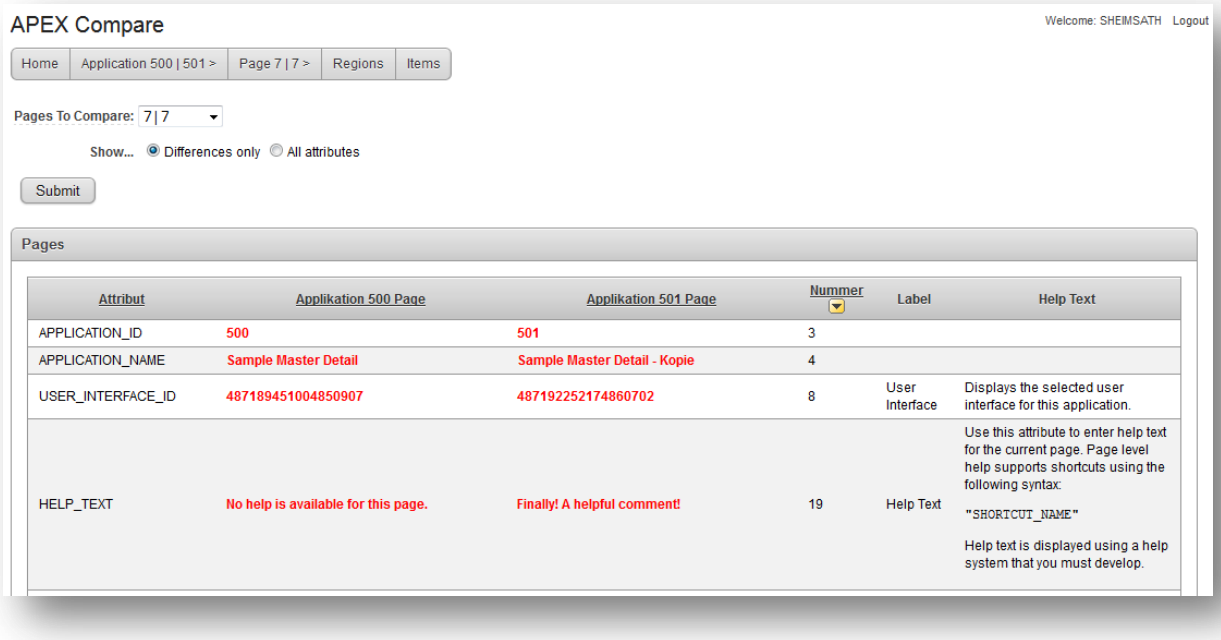
Das Ergebnis sieht dann zum Beispiel so aus (mit q-Operator[5] wegen eventueller Quotes):

```
FUNCTION all_application_pages_2 (p_comp_id number, p_page_id1
number, p_page_id2 number) RETURN t_av2lh_tab PIPELINED  AS

  v_werte  c_apex_application_pages%ROWTYPE;

begin

  select * into v_werte
  from c_apex_application_pages
  where comp_id = p_comp_id
  and (nvl(a_page_id,-1) = nvl(p_page_id1,-1))
  and (nvl(b_page_id,-1) = nvl(p_page_id2,-1));

  pipe row(t_av2lh_row('TAB_SET',
                       v_werte.a_TAB_SET,
                       v_werte.b_TAB_SET,
                      case when
                          v_werte.a_TAB_SET = v_werte.b_TAB_SET
                      then 'N'
                      when v_werte.a_TAB_SET is null
```

[5] http://docs.oracle.com/cd/B28359_01/appdev.111/b28370/fundamentals.htm#i38404

```
                        and v_werte.b_TAB_SET is null
                       then 'N'
                       else 'Y' end,
                       10,
                       q'{Standard Tab Set}',
                       q'{Select a standard tab set to be used
for this page. A standard tab set is associated with a specific
page number. You can use standard tabs to link users to a
specific page.}'));
-- usw.
```

Bindet man diese Funktion in einen Apex-Report ein und setzt ‚Display as' für die neue Spalte auf ‚Standard Report Column', so werden die Hilfetexte sogar mit HTML-Formatierungen angezeigt:



Fazit

Es lohnt sich, ein wenig in den Metadaten herumzustöbern und herauszufinden, wo welche Objekte einer Apex-Applikation verwaltet werden. Das gilt nicht nur für den Vergleich zweier Applikationen, sondern zum Beispiel auch für die Überprüfung von Entwicklungsrichtlinien und bestimmt noch viele andere Anwendungsfelder.

Für die hier entwickelte Beispielapplikation gibt es noch viele wünschenswerte Funktionalitäten, die sich mithilfe der Metadaten realisieren lassen.

Kontakt

Haben Sie Fragen?
Wünschen Sie Informationen, wenn die Applikation heruntergeladen werden kann oder eine neue Version dieses Dokuments bereit steht?
Dann melden Sie sich bitte einfach per Mail an sabine.heimsath@its-people.de.
Ihre Adresse wird nur für diesen Zweck verwendet.