Regressionstesten und Testdatenanonymisieung von DWH Ladestrecken

DOAG Koferenz Nürnberg 2016
Christian König, CGI Business Intelligence Expert
17.November 2016

CGI

Experience the commitment®

Referent: Christian König

Business Intelligence Expert,
CGI (Germany) GmbH & Co. KG

Schwerpunkte

- Konzeption und Implementierung von komplexen Datenbewirtschaftungen im Bereich BI&DWH
- Oracle Warehouse Builder, Oracle Data Integrator, Informatica, Ab Initio & SQL
- Ab Initio Certified Technician
- Informatica DI PC Developer Certified Specialist
- Branchenfokus Telekommunikation, Banking & Finance, Retail

Inhalte

CGI

Testen von Operativen vs. Dispositiven Systemen

Operative Systeme	Dispositive Systeme (DWH)
Testen gegen Spezifikation	Unklare Spezifikationen von Anforderungen und Quellsystemen
Unit-Tests etc.	Datengetrieben
Low-Level Programmiersprachen	High-Level Tools
Security wichtig	Security eher auf Tool – Ebene (Konfiguration)
Wasserfall	Iterative Entwicklung, Regressionstests

CGI

Generierung von künstlichen Testdaten

- Manuelles Erstellen von Testdaten (kleine Datenmenge)
  - Texteditor
  - Excel (Beispieldaten können auch als Dokumentation dienen)
  - Import:
    - Insert Statements
    - DB Tool (TOAD, SQL Developer, ..)
    - ETL Tool
    - External Table
- Generieren von Testdaten mit SQL (beliebig große Datenmengen)

```
Create table t as
Select
    level as zahl
from dual connect by level <= 10;
```

CGI

Generierung von künstlichen Testdaten

```
Create table t as
Select
    level as zahl
  , mod(level, 3)+1 as zahl_1_3
  , round(dbms_random.value(1,9)) as Rand_1_9
  , add_months(date'2016-01-01', level-1)+level-1  as Datum
  , lpad('X', level, 'Y') as string
  , dbms_random.string('X', 5) as rand_str
  , decode( mod(level, 3)+1
          , 1, 'eins'
          , 2, 'zwei'
          , 3, 'drei'
           , 'viele' ) as zahl_str
from dual connect by level <= 10;
```

ZAHL	ZAHL_1_3	RAND_1_9	DATUM	STRING	RAND_STR	ZAHL_STR
1	2	2	01. Jan 16	X	YM6LN	zwei
2	3	6	02. Feb 16	YX	9EZ2M	drei
3	1	6	03. Mrz 16	YYX	OPZYJ	eins
4	2	5	04. Apr 16	YYYX	MM1SL	zwei
5	3	7	05. Mai 16	YYYYX	3BKDU	drei
6	1	1	06. Jun 16	YYYYYX	HNC2Q	eins
7	2	8	07. Jul 16	YYYYYYX	2FWHZ	zwei
8	3	7	08. Aug 16	YYYYYYYX	RFE2T	drei
9	1	7	09. Sep 16	YYYYYYYYX	OBN0B	eins
10	2	1	10. Okt 16	YYYYYYYYYX	W1YQY	zwei

CGI

Generierung von Anonymisierten Echtdaten

- Wenn möglich: Echte Testdaten aus Produktivsystem
- Häufig aus Datenschutzgründen Anonymisierung erforderlich
- Häufig Mengeneinschränkung erforderlich, da Test < Prod
- Selektion der Produktiv-Daten via Anonymisierungsview
- Datentransfer von Prod→Test:
  - DB-Link
  - ETL-Tool
  - IMP/EXP
  - ...

CGI

Anonymisierungsview: Beispiel

```
CREATE OR REPLACE VIEW CUSTOMER
(
   ID,
   LAST_NAME,
   FIRST_NAME,
   DATE_OF_BIRTH,
   STREET,
   ZIP_CODE,
   TOWN,
   COUNTRY,
   EMAIL,
   CHANGE_DT
)
AS
   SELECT
              ID,
              AHASH(LAST_NAME) LAST_NAME,
              AHASH(FIRST_NAME) FIRST_NAME,
              DATE_OF_BIRTH,
              RAND(STREET) STREET,
              RAND(ZIP_CODE) ZIP_CODE,
              RAND(TOWN) TOWN,
              RAND(COUNTRY) COUNTRY,
              RAND(EMAIL) EMAIL,
              CHANGE_DT
     FROM  customer@dwhdblink
     WHERE mod(ID, 10) = 0;
```

CGI

Metadaten zur View-Generierung

anon_rules		
Table_Name	**Col_Name**	**Anon**
customer	id	NULL
customer	last_name	AHASH
customer	first_name	AHASH
customer	Date_of_birth	NULL
customer	street	RAND
customer	zip_code	RAND
customer	town	RAND
customer	country	RAND
customer	email	RAND
customer	change_dt	NULL

CGI

Script zur View-Generierung

```
declare
  str_sql varchar(2000);
  -- Cursor gibt Tabellenname, Attributliste, und Attributliste mit Anon-Funktion zurück
  cursor  cur_anon is
  select
      table_name
    , listagg(col_name, ',') within group(order by col_name) as col_list
    , listagg( anon || '(' || col_name || ') ' || col_name , ',')
        within group(order by col_name) as anon_list
  from anon_rules
  group by table_name;
begin
  for t in cur_anon
  loop
    begin
      str_sql :=
        'CREATE OR REPLACE VIEW '  || t.table_name
        || ' ( '  || t.col_list  || ' ) AS SELECT '
        || t.anon_list
        || ' FROM '  || t.table_name  || '@dwh_link';
      execute immediate str_sql;
      --dbms_output.put_line(str_sql);
    end;
  end loop;
end;
```

```
CREATE OR REPLACE VIEW CUSTOMER
(
   ID,
   LAST_NAME,
   FIRST_NAME,
)
AS
   SELECT
      ID,
      AHASH(LAST_NAME) LAST_NAME,
      AHASH(FIRST_NAME) FIRST_NAME,
FROM  customer@dwhdblink;
```


CGI

Manuelles Testen

- Bei kleinen Testmengen: Manueller Abgleich des Ergebnisses
- Vergleich mit erwartetem Ergebnis (Regressionstest, Performance-Test, Refactoring). Gleiche Daten sollen das gleiche Ergebnis liefern.
- Höhere Aggregationsebene: Summen, Anzahl Datensätze
- Komplett-Abgleich: Nachbau ETL mit SQL
- Testen von Namenskonventionen mit SQL:
  - Dictionary views (all_tables, all_tab_columns, …)
  - ETL Tool Repository
  - Runtime Repository (Laufzeiten, Fehlermeldungen, …)

CGI

Automatisiertes Regressionstesten

- Tägliche Regressionstests:
  - Transfer Testdaten via Anon-Views Prod→Test
  - 1 kompletter ETL-Beladungslauf
  - 1 kompletter Testfall-Lauf
- Mappings in Informatica, getriggert aus Informatica Workflows
- Testfälle in SQL
  - Aufgerufen aus Batch-File
  - Gescheduled per Cron
  - Synchronisation mit ETL-Lauf über Warte-Jobs/Files /Status-Flags
  - 1 SQL-Datei pro Testfall
    - Eintrag in Test-Protokoll-Tabelle am Anfang und Ende jedes Scriptes

CGI

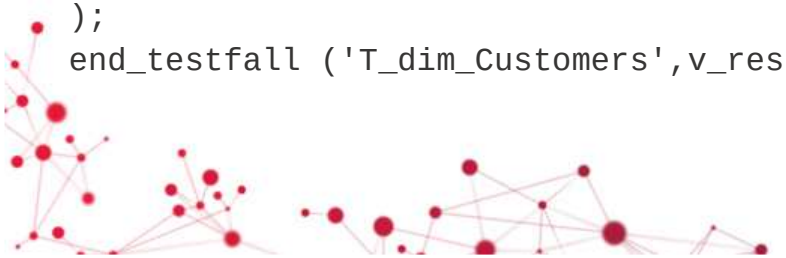
Beispiel Testfall

- Mappinglogik nachgebaut in SQL
- Danach Abgleich mit Zieltabelle per Minus
- Registrierung / Protokollierung der Ergebnisse am Anfang und Ende

Testresult					
Run_Id	**Testfall**	**Timestamp**	**Runtime**	**Result**	**Notes**
1	T_dim_Customers	14.10.2016 00:23	300	S	
1	T_revenue_sum	14.10.2016 00:40	120	E	300 Abweichungen
1	T_sales_sum	14.10.2016 00:23	60	E	20 Abweichungen
2	T_dim_Customers	15.10.2016 00:23	300	S	
2	T_revenue_sum	15.10.2016 00:23	120	S	
2	T_sales_sum	15.10.2016 00:23	60	E	20 Abweichungen
3	T_dim_Customers	16.10.2016 00:23	300	S	
3	T_revenue_sum	16.10.2016 00:23	120	S	
3	T_sales_sum	16.10.2016 00:23	60	S	

Beispiel Testfall

```
begin_testfall ('T_dim_Customers');
select
    case when count(*)=0 then 'S' else 'E' end, count(*) || ' Abweichungen'
into v_result, v_notes
from (
  with
  tf as (
    Select * from a inner join b on (x=y) where 1=1
  ),
  ipc as (
    Select * from dm.dim_customers
  )
  (
     Select 'tf' x, t.* from tf t
     minus
     Select 'tf' x, t.* from ipc t
   )
   union all
   (
     Select 'ipc' x, t.* from ipc t
               minus
               Select 'ipc' x, t.* from tf t
             )
);
end_testfall ('T_dim_Customers',v_result,v_notes);
```



CGI

Fazit

- Generierung von Testdaten ist automatisierbar
- Auch Namenskonventionen und Coding Guidelines lassen sich überprüfen
- Kompletter Regressionstest ist möglich, aber aufwändig

CGI

Christian König

Senior Consultant BI

Mobile: +49 170569 7832
E-Mail: christian.koenig@cgi.com

Vielen Dank für Ihre Aufmerksamkeit

CGI

Experience the commitment®

Our commitment to you

We approach every engagement with one objective in mind: to help clients succeed

CGI

Experience the commitment®