Regressionstesten und Testdatenanonymisierung von DWH-Ladestrecken

Christian König
CGI (Germany) GmbH & Co. KG
Düsseldorf

Schlüsselworte
ETL, DWH, Testen, Testdaten, Regressionstesten, Anonymisierung

Einleitung

Neben dem Testen mit künstlich erzeugten Daten hat im ETL-Umfeld das Testen mit anonymisierten Echt-Daten einen sehr hohen Stellenwert. Damit können nicht nur mögliche Fehler in der Implementierung, sondern auch mögliche Probleme in der Spezifikation und bei der Datenanalyse identifiziert werden. Aus Datenschutzgründen ist das Testen mit Original-Produktionsdaten häufig nicht möglich, so dass diese Daten anonymisiert bereitgestellt werden müssen. Um eine hohe Qualität der Entwicklung auch bei kleinen Änderungen zu gewährleisten, ist es außerdem wünschenswert, Testfälle automatisiert ausführen zu können. Dieser Vortrag stellt Methoden zur Generierung von künstlichen Testdaten, zur Anonymisierung von Produktionsdaten, und zum automatisierten oder manuellen Regressionstesten vor. Alle im Vortrag vorgestellten Methoden lassen sich mit Oracle DB Boardmitteln (SQL und PL/SQL) und Unix-Tools (batch, cron) umsetzen.

Testen im Data Warehouse vs. Testen in Operativen Systemen

Wie in jedem Software-Projekt kommt auch im Data Warehouse der Qualitätssicherung eine besondere Bedeutung zu. Im Unterschied zu operativen Systemen gibt es allerdings einige Punkte, die die Qualitätssicherung im Data Warehouse zu einer besonderen Aufgabe machen:

1. Anforderungen und Spezifikationen sind oft unklar, oder nicht korrekt. Dies gilt nicht nur für die Anforderungen an Kennzahlen und Dimensionen, sondern in einem noch viel größeren Maße für die Spezifikation und Dokumentation von Quellsystemen. Ein reines Testen gegen Spezifikationen ist häufig nicht ausreichend.

2. ETL-Jobs sind datengetrieben. Im Gegensatz zum Testen von operativen Systemen werden für das Testen von ETL-Jobs Quelldaten benötigt. Diese Quelldaten müssen soweit wie möglich den echten Daten entsprechen, um aussagekräftige Tests zu erstellen. Da man sich in der Praxis auf Spezifikationen der Quellsysteme nur eingeschränkt verlassen kann, kann man mit künstlich erzeugten Testdaten nur eingeschränkt testen.

Dem gegenüber haben wir im Data Warehouse auch einzige Vorteile gegenüber operativen Systemen:

1. Wir verwenden relativ hohe Programmiersprachen und Tools, wie ODI, Informatica, oder SQL. Mit diesen Tools werden ganze Fehlerklassen komplett vermieden. Fehleranfällige Themen wie Memory Management, Pointer Arithmetik, etc. haben für uns eine geringere Bedeutung.

2. Das gleiche gilt für das Thema Security. Auch wenn Software-Sicherheit im Data Warehouse an Bedeutung eher zunehmen wird, und sicherlich nicht komplett zu vernachlässigen ist, so hat das Thema im DWH nicht den gleichen Stellenwert wie z.B. bei

öffentlich zugänglichen Web-Front-Ends. Häufig werden die Zugriffe und Berechtigungen im DWH eher vom Reporting-Tool oder der Datenbank verwaltet.

Generierung von künstlichen Testdaten

Um künstliche Testdaten zu generieren, gibt es mehrere Möglichkeiten:

1. Manuelles Erstellen von künstlichen Testdaten. Häufig braucht man keine großen Datenmengen zum Testen, sondern will einen ETL-Job anhand einiger weniger Beispiel-Datensätze testen. Diese Datensätze lassen sich gut manuell erstellen, entweder mit einem Text-Editor, oder z.B. in Excel. Gerade Excel bietet sich an, wenn man die Funktionsweise der ETL-Strecke an einem Beispiel darstellen und dokumentieren will. In diesem Fall kann man das Excel Dokument (Quelldaten und erwartetes Ergebnis) auch als Dokumentation der ETL-Logik verwenden. Manche Datenbank-Tools, wie z.B. TOAD, können Excel oder csv-Daten direkt in Datenbank-Tabellen importieren. Alternativ kann man auch mit Excel INSERT-Statements generieren (mit „Verketten“).

2. Alternativ kann man Testdaten auch direkt mit SQL generieren. Das folgende Statement generiert z.B. eine Tabelle mit 10 Zeilen:

```
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;
```

Das Attribut

- Zahl ist eine fortlaufende Zahl (Zeilennummer)
- Zahl_1_3 ist eine Zahl zwischen 1 und 3
- Rand_1_9 ist eine Zufallszahl zwischen 1 und 9
- Das Datum zählt die Monate und Tage jeweils um 1 hoch
- String ist eine Zeichenkette mit definierter Länge
- Rand_Str ist eine zufällige Zeichenfolge
- Und Zahl_str enthält eine von 3 definierten Zeichenketten.

Damit und mit anderen Funktionen (und z.B: Joins) lassen sich relativ einfach und performant beliebige Tabelleninhalte erzeugen, auch für größere Datenmengen.

.

Diese Tabelle sieht folgendermaßen aus:

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-2016	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-2016	YYYYYYYYYX	W1YQY	zwei

Diese Art der Testdatengenerierung eignet sich besonders für Performance-Tests. Hierbei kann man die Daten ziemlich detailliert spezifizieren und gleichzeitig große Datenmengen generieren.

Echte Testdaten

Aufgrund der o.g. Anforderungen eignen sich künstlich generierte Testdaten nur bedingt, um ETL-Strecken vollumfänglich zu testen. Viele Probleme werden erst ersichtlich, wenn man mit möglichst echten Daten testet. Am besten wäre es, wenn man die Daten aus produktiven Quellsystemen unverändert benutzen könnte. Leider gibt es in der Praxis häufig zwei Einschränkungen:

1. Aufgrund von Datenschutzbestimmungen müssen viele Attribute der Testdaten anonymisiert werden.

2. Da die Testsysteme häufig kleiner dimensioniert sind als Produktivsysteme, ist es erforderlich, die Datenmenge zu reduzieren.

Eine Möglichkeit, diese beiden Bedingungen zu erfüllen, ist es, die Daten über Anonymisierungs-Views abzugreifen:

```
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;
```

Mit einer solchen View lassen sich einzelne Attribute anonymiseren, und die Datenmenge reduzieren. Wenn möglich, kann man durch diese View direkt auf das Produktiv-System zugreifen, anderenfalls kann man die Daten auf das Test-System kopieren, importieren/exportieren, oder via ETL-Tool auf die Testumgebung verschieben.

Solche Views kann natürlich manuell anlegen, man kann sie aber auch aus Metadaten generieren. Dazu braucht man nur eine Tabelle, in der der für jedes Attribut eine entsprechende Anonymisierungsregel definiert ist:

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

Aus einer solchen Tabelle kann man z.B. o.g. View per Script generieren. So ein Script sieh z.B. wie folgt aus:

```
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 column_name) as col_ist
```

```
    , listagg( anon || '(' col_name || ') ' || col_name ) within
group(order by column_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_blink';
      execute immediate str_sql;
    end;
  end loop;
end;
```

Testdurchführung

Nachdem die Testdaten erstellt wurden, kann der ETL-Ladelauf getestet werden. Anschließend muss man nur noch die Ergebnisse des ETL-Ladelaufes mit den Erwartungen abgleichen. Bei manuellen Tests mit kleinen Testdaten ist es möglich, die Ergebnismenge manuell abzugleichen. Bei großen Datenmengen kann man das Ergebnis häufig nicht komplett überblicken. Dann ergeben sich beispielsweise folgende Möglichkeiten:

- Das Ergebnis der Tabelle wird mit einem vorher erstellten Ergebnis verglichen. Damit kann man vor allem Migrationen, Refactoring- oder Performance-Maßnahmen testen, also alle Maßnahmen, bei denen das Ergebnis nicht verändert wird.
- Vergleiche mit den Quelldaten auf höherer Aggregationsebene: Ist die Summe gleich, stimmt die Anzahl der Datensätze? Hat die Zieltabelle genau x Monatspartitionen?
- Wenn man die Zieldaten komplett mit den Quelldaten vergleichen will, muss man die komplette Logik der ETL-Strecke mit SQL oder anderen Tools nachbauen. Das kann sicherlich in Einzelfällen sinnvoll sein; eine 100%ige Test-Abdeckung ist so nur mit sehr viel Aufwand zu erreichen.
- Auch Namenskonventionen und andere Architekturvorgaben lassen sich mit SQL sehr gut überprüfen. Entsprechen alle Tabellen und Spalten den Namenskonventionen? Sind alle erforderlichen Indizes und Constraints definiert? Sind entsprechende Kommentare vorhanden? Auch das Repository des ETL-Tools kann man auf diese Weise abfragen. Wenn man automatisierte Testläufe hat, dann kann man auch zusätzlich das Runtime-Repository abfragen (Sind alle Jobs technisch ohne Fehler in der erwarteten Zeit durchgelaufen ?

Automatisiertes Testen

Für einen automatisierten Testlauf muss man die o.g. Lösungen zusammenfügen. Dann könnte beispielsweise ein nächtlicher Testlauf auf der Testumgebung wie folgt aussehen:

- Die Daten der produktiven Quellsysteme werden via Anonymisierungs-Views anonymisiert und auf das Testsystem transferiert.
- Danach werden die ETL-Jobs gestartet.

- Nachdem die ETL-Beladung gelaufen ist, werden die Testfälle ausgeführt. Die Testfälle können z.B. aus SQL-Scripten bestehen, die nacheinander aufgerufen werden (z.B. von einem Batch-Job) und die das Testergebnis in einer Testprotokoll-Tabelle protokollieren.

Kontaktadresse:
Christian König
CGI (Germany) GmbH & Co. KG
Heerdter Lohweg 35
D-40549 Düsseldorf

Telefon: +49 170569 7832
E-Mail christian.koenig@cgi.com
Internet: www.de.cgi.com