Mitten in der Nacht bricht die ETL-Verarbeitung ab, weil ein falscher oder unvollständiger Datensatz geliefert wurde. Das Laden des Data Warehouse ist nicht möglich, und erst nach einem manuellen Eingriff können die Ladeprozesse fortgesetzt oder neu gestartet werden. Das muss nicht sein! Zwischen ETL-Abbruch und Ignorieren der fehlerhaften Datensätze gibt es zahlreiche weitere Möglichkeiten, um häufig auftretende Fehlersituationen zu erkennen und automatisch zu behandeln.

Fehlertolerante Ladeprozesse in Oracle gegen schlaflose Nächte

Dani Schnider, Trivadis AG

Oft sind es Kleinigkeiten, die zum Abbruch der ETL-Verarbeitung führen. Ein fehlendes Attribut, ein unbekannter Codewert, eine ungültige Referenz auf eine Dimension oder eine Schlüsselverletzung aufgrund doppelt oder mehrfach gelieferter Datensätze kann dazu führen, dass der ETL-Job abgebrochen wird. Die Folge davon ist entweder, dass die aktuellen Daten am anderen Morgen nicht zur Verfügung stehen oder dass ein Mitarbeiter des Betriebsteams beziehungsweise ein DWH-Entwickler in der Nacht manuelle Daten-Korrekturen vornehmen und den ETL-Job wieder starten muss. Im ersten Fall führt dies dazu, dass bei wiederkehrenden Fehlerfällen die Akzeptanz des Data Warehouse bei den Anwendern stark leidet. Der zweite Fall führt zu häufigen manuellen Eingriffen in der Nacht und somit zu schlaflosen Nächten bei den betroffenen DWH-Mitarbeitern.

```
INSERT INTO cls_products (pro-
duct_code, product_desc)
SELECT product_code, product_
desc
  FROM stg_products;

ORA-01400: cannot insert
NULL into("CLEANSE"."CLS_
PRODUCTS"."PRODUCT_DESC")
```

Listing 1

Weder unzufriedene Benutzer noch übermüdete DWH-Entwickler sind dem Erfolg eines Data Warehouse förderlich. Deshalb brauchen wir Alternativen, mit denen die ETL-Verarbeitung auch dann fortgesetzt werden kann, wenn einzelne Datensätze unvollständig oder falsch sind. Ein einfacher Ansatz ist, Fehler einfach zu ignorieren. So ist es beispielsweise in Oracle Warehouse Builder möglich, die Konfiguration eines Mappings so einzustellen, dass eine maximale Anzahl von Fehlern pro Lauf erlaubt ist. Das klingt zwar verlockend einfach, hat aber zwei Nachteile. Erstens werden die Daten unvollständig geladen. Zweitens dauert die Verarbeitung bei großen Daten-Mengen länger, weil das Mapping in der Betriebsart „row-based" ausgeführt werden muss. In vielen DWH-Systemen mag dieser Ansatz genügen, aber je nach Anforderungen an Daten-Qualität und Ladezeiten sind solche Lösungen nicht realistisch. Zum Glück gibt es andere, ebenfalls einfache Lösungen.

Auf den folgenden Seiten wird anhand von typischen Fehler-Situationen aufgezeigt, wie fehlerhafte Datensätze erkannt und so behandelt werden können, dass die ETL-Verarbeitung trotzdem fortgesetzt werden kann. Die hier vorgestellten Verfahren lassen sich in Oracle sowohl mit SQL als auch mit ETL-Tools wie Oracle Warehouse Builder (OWB) oder Oracle Data Integrator (ODI) realisieren. In den nachfolgenden Beispielen wird jeweils der Lösungsansatz mit SQL gezeigt, der aber sinngemäß auch mit OWB oder ODI implementiert werden kann.

Fehlende Attribute

Ein typischer Fehlerfall ist ein leeres Attribut, das in der Zieltabelle als „NOT NULL" definiert ist. Dies führt normalerweise zu einem Abbruch der Verarbeitung, sobald ein unvollständiger Datensatz, beispielsweise ein Produkt ohne Beschreibung, auftritt. Das folgende einfache Beispiel zeigt, dass die Produkt-Dimension aus der Stage-Tabelle „STG_PRODUCTS" in die bereinigte Tabelle „CLS_PRODUCTS" der Cleansing-Area lädt. Ohne explizite Fehlerbehandlung in SQL führt dies im Falle eines NULL-Werts zu einem Abbruch mit entsprechender Fehlermeldung (siehe Listing 1).

Die einfachste Lösung, um einen Abbruch zu verhindern, besteht darin, die fehlerhaften Datensätze zu filtern und somit zu vermeiden, dass sie in die Ziel-Tabelle geschrieben werden. Dies kann entweder mit einem Cursor-Loop und entsprechendem Exception Handler in PL/SQL implementiert werden (also row-based) oder ganz einfach mit einer WHERE-Bedingung im entsprechenden SQL-Statement (siehe Listing 2).

Bei dieser Lösung nehmen wir in Kauf, dass unter Umständen unvollständige Daten ins Data Warehouse geladen werden. In einigen Fällen kann

dies akzeptabel sein, solange die Anzahl der Fehler nicht zu groß wird. Es empfiehlt sich deshalb, nach dem Laden in die Cleansing-Area einen Check einzubauen, der die Anzahl der Datensätze in Staging- und Cleansing-Area vergleicht und bei Überschreitung eines Schwellwerts die Verarbeitung abbricht. Diese Anforderung lässt sich – sofern der Schwellwert als absolute Zahl und nicht als prozentualer Anteil definiert werden soll – in Oracle mithilfe von DML-Error-Logging implementieren (siehe Listing 3).

Dabei werden Datensätze, die zu Constraint-Verletzungen führen, nicht in die Ziel-Tabelle geladen, sondern in eine zuvor erstellte Fehler-Tabelle geschrieben. Bei Bedarf kann ein Schwellwert – hier maximal 100 Fehler – definiert werden. Ein wesentlicher Vorteil von DML-Error-Logging besteht darin, dass die Verarbeitung weiterhin „set-based“ ausgeführt werden kann.

Da aber die fehlerhaften Datensätze nicht in die Ziel-Tabelle geladen werden, handelt es sich hier nur um eine erweiterte Implementation der Filter-Variante – allerdings mit dem Vorteil, dass die fehlerhaften Datensätze nicht einfach ignoriert werden, sondern in der Fehler-Tabelle verfügbar sind.

Das Filtern von fehlerhaften Datensätzen – ob mit „WHERE“-Bedingung oder DML-Error-Logging implementiert – führt spätestens dann zu Problemen, wenn Referenzen auf die fehlenden Daten auftreten. Was passiert in unserem Beispiel, wenn in einem späteren Schritt der ETL-Verarbeitung Fakten in eine Fakten-Tabelle geladen werden, die auf das fehlende Produkt verweisen? Das Filtern des fehlerhaften Datensatzes kann zu Folgefehlern in weiteren ETL-Schritten führen.

Ein anderer einfacher, aber nicht zu empfehlender Ansatz ist, den „NOT NULL“-Constraint auf der Ziel-Tabelle wegzulassen und somit fehlende Attributwerte zu erlauben. Das hört sich zwar verlockend simpel an, führt aber ebenfalls zu Folgefehlern und zu Problemen bei den Auswertungen. Wie sollen die leeren Felder in einem Report oder einem OLAP-Tool angezeigt werden, sodass sie für den Anwender erkennbar sind? Zusätzliche Fallunterscheidungen in den Abfragen werden notwendig, um die leeren Felder entsprechend zu markieren. Damit wir dies nicht für jede Auswertung tun müssen, ist es naheliegender und einfacher, diesen Schritt bereits im ETL-Prozess durchzuführen. Dies führt zum empfohlenen und bewährten Lösungsansatz, mit Singletons zu arbeiten.

Ein Singleton ist ein Platzhalter oder Defaultwert, der in bestimmten Fehlersituationen, beispielsweise bei einem leeren Attribut, eingesetzt wird. Für unser Beispiel möchten wir anstelle einer leeren Produktbezeichnung den Wert „unknown“ anzeigen. Das lässt sich durch eine einfache Erweiterung im ETL-Prozess erreichen (siehe Listing 4).

```
INSERT INTO cls_products (pro-
duct_code, product_desc)
SELECT product_code, product_
desc
  FROM stg_products
 WHERE product_desc IS NOT
NULL;
```

Listing 2

```
INSERT INTO cls_products (pro-
duct_code, product_desc)
SELECT product_code, product_
desc
  FROM stg_products
LOG ERRORS REJECT LIMIT 100;
```

Listing 3

```
INSERT INTO cls_products (product_code, product_desc)
SELECT product_code
     , NVL(product_desc, 'Unknown')
  FROM stg_products;
```

Listing 4

Dieses Verfahren, das wir in erweiterter Form auch für Code-Lookups und Referenzen auf Dimensionen verwenden können, wird in vielen Data Warehouses eingesetzt und hat den Vorteil, dass der Datensatz geladen und somit später referenziert werden kann, beispielsweise von Fakt-Einträgen mit Verweisen auf den Dimensions-Eintrag. In Reports und OLAP-Tools werden die Einträge auch angezeigt, allerdings mit der Bezeichnung „unknown“ statt der korrekten (fehlenden) Bezeichnung.

Wie verhalten sich Singletons in Kombination mit Slowly Changing Dimensions (SCD)? Nehmen wir an, das fehlende Attribut wird im Quellsystem nachträglich ergänzt und in einem späteren ETL-Lauf ins DWH geladen. Bei „SCD Typ 1“ wird der bisherige Datensatz überschrieben, und ab diesem Zeitpunkt wird in allen Auswertungen der korrekte Wert angezeigt. Bei „SCD Typ 2“ wird eine neue Version in die Dimensionstabelle eingefügt. Das hat zur Folge, dass neue Fakten auf den vollständigen Eintrag verweisen. Bereits geladene Fakten zeigen aber weiterhin auf den Eintrag mit der Bezeichnung „unknown“.

Unbekannte Code-Werte

Typisch in ETL-Prozessen sind Lookups auf Code- oder Referenz-Tabellen. Anhand eines Code-Werts oder eines fachlichen Schlüssels werden ein künstlicher Schlüssel (Surrogate Key) sowie eventuell weitere Attribute wie eine Bezeichnung ermittelt. Was passiert nun während der ETL-Verarbeitung, wenn der entsprechende Code-Wert in der Lookup-Tabelle nicht vorhanden ist? Der einfachste Fall besteht auch hier wieder darin, die fehlerhaften Datensätze zu ignorieren.

Dieses Verfahren wird häufig – oft ungewollt – verwendet, indem in SQL ein normaler Inner-Join zwischen Quell-Tabelle und Lookup-Tabelle gemacht wird. Die Folge ist, dass Datensätze mit fehlenden oder unbekannten

```
INSERT INTO cls_products (product_code, product_desc, category_id)
SELECT stg.product_code
     , NVL(product_desc, ‚Unknown‘)
     , NVL(lkp.category_id, -1)
FROM stg_products stg
  LEFT OUTER JOIN co_categories lkp
    ON (lkp.category_code = stg.category_code);
```

Listing 5

```
INSERT INTO cls_products (product_code, product_desc)
SELECT product_code, NVL(product_desc, ‚Unknown‘)
  FROM (SELECT product_code, product_desc
             , ROW_NUMBER() OVER(PARTITION BY product_code
                                 ORDER BY product_desc) rnum
          FROM stg_products)
 WHERE rnum = 1;
```

Listing 6

ID	CODE	DESCRIPTION
-1	n/a	Unknown

Tabelle 1

ID	CODE	DESCRIPTION
5432	ABC	Unknown

Tabelle 2

Code-Werten nicht in die Ziel-Tabelle geschrieben werden. Weil durch diese Lösungsvariante fehlerhafte Datensätze gefiltert werden, haben wir wiederum das Problem, dass die Daten unvollständig geladen werden. Um dies zu vermeiden, können auch hier Singletons eingesetzt werden, wenn auch in etwas erweiterter Form.

Beim initialen Laden des Data Warehouse wird in jede Lookup-Tabelle ein Singleton-Eintrag geschrieben, der durch einen speziellen Schlüssel, zum Beispiel eine negative ID, gekennzeichnet wird (siehe Tabelle 1).

Beim Lookup wird ein Outer-Join auf die Lookup-Tabelle gemacht. Damit ist gewährleistet, dass auch Datensätze mit fehlenden oder unbekannten Code-Werten in die Ziel-Tabelle geschrieben werden. Um zu vermeiden, dass ein leerer Schlüssel übergeben wird, wird der leere Eintrag durch den Schlüssel des Singleton-Eintrags, in unserem Fall durch den Wert „-1“, ersetzt. In Oracle SQL lässt sich dies einfach mittels der NVL-Funktion realisieren (siehe Listing 5).

Das Prinzip der Singletons erlaubt auch hier ein vollständiges Laden aller Daten, hat aber den Nachteil, dass eine nachträgliche Zuordnung zum korrekten Code-Wert nicht mehr möglich ist. Auch wenn später der fehlende Code nachgeliefert wird, kann er in den bereits geladenen Daten nicht mehr aktualisiert werden – es sei denn, der Original-Wert des Quellsystems wird zusätzlich im DWH gespeichert.

Eine flexiblere, aber auch etwas aufwändigere Möglichkeit besteht darin, fehlende Code-Werte vorgängig in die Lookup-Tabelle zu laden. Angenommen, das Quellsystem liefert einen Datensatz mit dem Code „ABC“, der im Data Warehouse noch nicht vorhanden ist. Deshalb wird nun ein neuer Eintrag in die Code-Tabelle geschrieben, der im ersten Moment aussieht wie ein Singleton-Wert, jedoch einen neuen Surrogate Key (hier den Wert „5432“) zugewiesen bekommt (siehe Tabelle 2).

Wird die Bezeichnung für den Code „ABC“ später nachgeliefert, kann der Datensatz überschrieben werden. Da die bereits geladenen Daten auf die ID „5432“ verweisen, wird ab diesem Zeitpunkt der korrekte Wert angezeigt.

Die zuvor eingefügten Datensätze in der Lookup-Tabelle entsprechen somit den echten Datensätzen, die später vom Quellsystem geliefert werden. Da sie bereits vorhanden sind, aber noch keinen Namen (also keine Bezeichnung) haben, nennen wir sie „Embryo“-Einträge. Die Geburt, das heißt die Umwandlung eines Embryo-Eintrags in einen echten Lookup-Eintrag, erfolgt, wenn der Code-Wert vom Quellsystem geliefert und ins Data Warehouse geladen wird.

Fehlende Dimensions-Einträge

Was hier anhand von Code-Tabellen beschrieben wurde, lässt sich auf beliebige Lookup-Tabellen und somit auch auf Dimensions-Tabellen anwenden. Dies ist dann interessant, wenn die Situation auftreten kann, dass Fakten bereits geliefert werden, bevor die zugehörigen Dimensions-Einträge im Data Warehouse vorhanden sind. Auch hier können die oben beschriebenen Varianten-Filterung, Singleton- und Embryo-Einträge angewendet werden. Eine ausführliche Beschreibung dieser Problemstellung sowie der drei Lösungsvarianten ist im Artikel „Wenn die Fakten zu früh eintreffen“ (siehe http://www.trivadis.com/uploads/tx_cabagdownloadarea/Wenn_die_Fakten_zu_frueh_eintreffen.pdf) dokumentiert.

Doppelte Datensätze

Eine ebenfalls häufig anzutreffende Fehlerursache sind doppelt oder mehrfach vorhandene Datensätze innerhalb einer Lieferung. Grund dafür können Mehrfach-Lieferungen oder nicht eindeutige Join-Kriterien in den Extraktionsprozessen sein. Doppelte Datensätze führen typischerweise zu Schlüsselverletzungen (Primary Key Violation oder Unique Key Violation) beim Laden ins DWH und somit zu einem Abbruch der ETL-Verarbeitung.

Der nächstliegende und einfachste Ansatz, um doppelte Datensätze zu eliminieren, besteht darin, ein „DISTINCT“ in der Abfrage auf die Stage-Tabelle zu verwenden. Solange alle Attribute der Datensätze identisch sind, funktioniert dies tadellos. Doch sobald sich die Datensätze in mindestens ei-

nem beschreibenden Attribut unterscheiden, aber dennoch den gleichen Schlüssel besitzen, wird die ETL-Verarbeitung trotzdem abgebrochen. Dieser Fall dürfte zwar theoretisch nicht auftreten, kommt aber in der Praxis aufgrund von ungenauen oder fehlerhaften Extraktionsprozessen leider vor. Um sicherzustellen, dass auch in diesem Fall nur ein Datensatz übernommen wird, können beispielsweise mit der analytischen Funktion „ROW_NUMBER“ alle Datensätze mit einem Schlüssel durchnummeriert und nur der jeweils erste Datensatz übernommen werden, wie Listing 6 zeigt.

Diese Variante funktioniert in jedem Fall und durch Verwendung der analytischen Funktion erst noch effizient. Doch die entscheidende Frage lautet: Welcher ist der erste Datensatz? Im hier aufgeführten Beispiel wird der Eintrag übernommen, dessen Produktbezeichnung alphabetisch zuoberst erscheint. Die Regel ist völlig willkürlich und dient nur dazu, einen zufälligen Datensatz zu übernehmen. Falls eine solche pragmatische Lösung nicht genügen sollte, ist die Variante mit „DISTINCT“ besser geeignet, da sie zwar bei identischen Datensätzen funktioniert, aber bei unterschiedlichen Datensätzen mit gleichem Schlüssel zu einem Abbruch führt. Eine elegantere Lösung besteht darin, mithilfe von DML-Error-Logging die doppelten Datensätze in die zuvor erstellte Fehler-Tabelle zu schreiben (siehe Listing 7).

```
INSERT INTO cls_products (product_code, product_desc)
SELECT product_code, product_desc
  FROM stg_products
LOG ERRORS REJECT LIMIT UNLIMITED;
```

Listing 7

```
SELECT product_code, ora_err_mesg$ FROM err$_cls_products;

PRODUCT_CODE   ORA_ERR_MESG$
-------------- ------------------------------------------------
12345-67890-76 ORA-00001: unique constraint
               (CLEANSE.CLS_PRODUCTS_UK) violated
```

Listing 8

Nehmen wir an, in der Stage-Tabelle „STG_PRODUCTS“ kommen zwei Datensätze mit dem Produktcode „12345-67890-76“ vor. Der erste Datensatz wird dann in die Ziel-Tabelle geladen, der zweite in die Fehler-Tabelle, wie die Abfrage auf die Fehler-Tabelle zeigt (siehe Listing 8).

Welcher der erste (und damit korrekte) Datensatz ist, hängt von der physischen Speicherung in der Stage-Tabelle ab und ist somit auch zufällig. Das oben beschriebene Problem im Falle von unterschiedlichen Attributen tritt hier also genauso auf wie bei der Variante mit „ROW_NUMBER“.

Fazit

Eine allgemeine Patentlösung für die Behandlung von Fehlern in ETL-Prozessen gibt es zwar nicht und es ist auch kaum möglich, alle auftretenden Fehlerfälle abzufangen und automatisch zu behandeln. Trotzdem sollte versucht werden, durch geeignete Verfahren die häufig auftretenden Fehlerfälle so zu behandeln, dass sie nicht zu einem Abbruch der ETL-Verarbeitung führen.

Bei allen hier beschriebenen Verfahren kann die Verarbeitung auch beim Auftreten von fehlerhaften Datensätzen fortgesetzt werden, und alle Lösungen lassen sich set-based ausführen. Jede Variante hat aber Vor- und Nachteile bezüglich Daten-Qualität, Folgefehlern und Komplexität. Deshalb sollten für jedes Data Warehouse die geeigneten Verfahren zur Fehlerbehandlung definiert werden. Die in diesem Artikel beschriebenen Methoden können dabei helfen, stabilere ETL-Abläufe zu implementieren und somit den Betriebsverantwortlichen und DWH-Entwicklern eine ruhige und erholsame Nacht zu ermöglichen.

Dani Schnider
dani.schnider@trivadis.com