# Fehlertolerante Ladeprozesse in Oracle gegen schlaflose Nächte

**Dani Schnider**
**Trivadis AG**
**Zürich/Glattbrugg, Schweiz**

**Schlüsselworte:**

Data Warehouse, ETL, Fehlerbehandlung, Singletons, Embryo-Einträge, SQL, Analytische Funktionen, DML Error Logging, Oracle Warehouse Builder, Oracle Data Integrator

## Einleitung

Mitten in der Nacht bricht die ETL-Verarbeitung ab, weil ein falscher oder unvollständiger Datensatz geliefert wurde. Das Data Warehouse kann nicht geladen werden, 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 Fehlersituationenen zu erkennen und automatisch zu behandeln.

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 gelieferten 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 bzw. ein DWH-Entwickler in der Nacht manuelle Datenkorrekturen vornehmen und den ETL-Job wieder starten muss. Im ersten Fall führt dies dazu, dass bei wiederkehrenden Fehlerfällen die Akzeptanz des Data Warehouses 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.

Weder unzufriedene Benutzer noch übermüdete DWH-Entwickler sind dem Erfolg eines Data Warehouses 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 es, Fehler einfach zu ignorieren. So ist es beispielsweise in Oracle Warehouse Builder möglich, die Konfiguration eines Mappings so einzustellen, dass eine maximale Anzahl Fehler pro Lauf erlaubt ist. Das tönt zwar verlockend einfach, hat aber zwei Nachteile. Erstens werden die Daten unvollständig geladen. Zweitens dauert die Verarbeitung bei grossen Datenmengen 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 Datenqualität und Ladezeiten sind solche Lösungen nicht realistisch. Zum Glück gibt es andere, ebenfalls einfache Lösungen.

Auf den folgenden Seiten wird anhand von typischen Fehlersituationen 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. Im folgenden wird jeweils ein einfaches Beispiel verwendet, das die Produktdimension 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 entsprechenden Fehlermeldung:

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

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

Die einfachste Lösung, um einen Abbruch zu verhindern, besteht darin, die fehlerhaften Datensätze zu filtern und somit zu vermeiden, dass sie in die Zieltabelle 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:

```
INSERT INTO cls_products (product_code, product_desc)
SELECT product_code, product_desc
  FROM stg_products
 WHERE product_desc IS NOT NULL;
```

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, welcher die Anzahl Datensätze in Staging Area und Cleansing Area vergleicht und bei Überschreitung eines Schwellwertes die Verarbeitung abzubrechen.

Diese Anforderung lässt sich – sofern der Schwellwert als absolute Zahl und nicht als prozentualer Anteil definiert werden soll – in Oracle mit Hilfe von DML Error Logging implementieren:

```
INSERT INTO cls_products (product_code, product_desc)
SELECT product_code, product_desc
  FROM stg_products
LOG ERRORS REJECT LIMIT 100;
```

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

Da aber die fehlerhaften Datensätze nicht in die Zieltabelle 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 Fehlertabelle 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

existieren. Was passiert in unserem Beispiel, wenn in einem späteren Schritt der ETL-Verarbeitung Fakten in eine Faktentabelle geladen werden, welche auf das fehlende Produkt verweisen? Das Filtern der fehlerhaften Datensatzes kann zu Folgefehlern in weiteren ETL-Schritten führen.

Ein anderer einfacher, aber nicht zu empfehlender Ansatz ist es, den NOT NULL-Constraint auf der Zieltabelle wegzulassen und somit fehlende Attributwerte zu erlauben. Das tönt zwar verlockend simpel, 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, dass 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 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:

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

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 Fakteinträgen mit Verweisen auf den Dimensionseintrag. 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 Codewerte**

Typisch in ETL-Prozessen sind Lookups auf Code- oder Referenztabellen. Anhand eines Codewerts oder eines fachlichen Schlüssels wird ein künstlicher Schlüssel (Surrogate Key) sowie eventuell weitere Attribute wie eine Bezeichnung ermittelt. Was passiert nun während der ETL-Verarbeitung, wenn der entsprechende Codewert 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, in dem in SQL ein normaler Inner-Join zwischen Quelltabelle und Lookup-Tabelle gemacht wird. Die Folge ist, dass Datensätze mit fehlenden oder unbekannten Codewerten nicht in die Zieltabelle 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 Warehouses wird in jede Lookup-Tabelle ein Singleton-Eintrag geschrieben, der durch einen speziellen Schlüssel, z.B. eine negative ID gekennzeichnet wird:

| ID | CODE | DESCRIPTION |
|---|---|---|
| -1 | n/a | Unknown |

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

```
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);
```

Das Prinzip der Singletons erlaubt auch hier ein vollständiges Laden aller Daten, hat aber den Nachteil, dass eine nachträgliche Zuordnung zum korrekten Codewert 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 Originalwert des Quellsystems wird zusätzlich im DWH gespeichert.

Eine flexiblere, aber auch etwas aufwendigere Möglichkeit besteht darin, fehlende Codewerte 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 Codetabelle geschrieben, der im ersten Moment aussieht wie ein Singletonwert, jedoch einen neuen Surrogate Key (hier den Wert 5432) zugewiesen bekommt.

| ID | CODE | DESCRIPTION |
|---|---|---|
| 5432 | ABC | Unknown |

Wird die Bezeichung 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 vorgängig 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 (d.h. keine Bezeichnung) haben, nennen wir sie „Embryo“-Einträge. Die „Geburt“, also die Umwandlung eines Embryo-Eintrags in einen „echten“ Lookup-Eintrag erfolgt zum Zeitpunkt, wenn der Codewert vom Quellsystem geliefert und ins Data Warehouse geladen wird.

**Fehlende Dimensionseinträge**

Was hier anhand von Codetabellen beschrieben wurde, lässt sich für beliebige Lookup-Tabellen und somit auch für Dimensionstabellen anwenden. Dies kann dann interessant sein, wenn die Situation auftreten kann, dass Fakten bereits geliefert werden, bevor die zugehörigen Dimensionseinträge im Data Warehouse vorhanden sind. Auch hier können die oben beschriebenen Varianten Filterung, Singleton-Einträge und Embryo-Einträge angewendet werden. Eine ausführliche Beschreibung dieser Problemstellung sowie der drei Lösungsvarianten ist im Artikel «Wenn die Fakten zu früh eintreffen»[1] dokumentiert.

**Doppelte Datensätze**

Eine ebenfalls häufig anzutreffende Fehlerursache sind doppelt oder mehrfach vorhandene Datensätze innerhalb einer Lieferung. Grund dafür können Mehrfachlieferungen oder nicht eindeutige Joinkriterien 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 naheliegendste 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 einem beschreibenden Attribut unterscheiden, aber trotzdem 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 in folgendem Beispiel gezeigt:

```
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;
```

Diese Variante funktioniert in jedem Fall, und durch Verwendung der Analytischen Funktion erst noch effizient. Doch die entscheidende Frage lautet: Welches 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.

[1] http://www.trivadis.com/uploads/tx_cabagdownloadarea/Wenn_die_Fakten_zu_frueh_eintreffen.pdf

Eine elegantere Lösung besteht darin, mit Hilfe von DML Error Logging die doppelten Datensätze in die zuvor erstellte Fehlertabelle zu schreiben:

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

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 Zieltabelle geladen, der zweite in die Fehlertabelle, wie die folgende Abfrage auf die Fehlertabelle zeigt:

```
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
```

Welches 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 genau auf wie bei der Variante mit ROW_NUMBER.

**Zusammenfassung**

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 Datenqualität, Folgefehler 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.

Kontaktadresse:

Dani Schnider
Trivadis AG
Europa-Strasse 5
CH-8152 Glattbrugg

Telefon: +41(0)44-808 70 20
Fax: +41(0)44-808 70 21
E-Mail dani.schnider@trivadis.com
Internet: www.trivadis.com