DAIMLER

Fehlerbehandlung mittels DML Error Logging

Business Intelligence

Andreas Buckenhofer, 20.03.2014

Daimler TSS
Enabling Excellence

Zur Person
Andreas Buckenhofer

Andreas Buckenhofer
Senior DB Professional
E-Mail: andreas.buckenhofer@daimler.com

Seit 2009 bei Daimler TSS im Fachgebiet Business Intelligence (Cognos/Informatica)

Schwerpunkt DWH/CRM seit 1998 als

- Entwickler
- Administrator
- Berater

Unternehmensüberblick
Unser Selbstverständnis

Daimler TSS – der IT-Spezialist im Daimler-Konzern.

Wir realisieren für unsere Kunden anspruchsvolle Applikationen, stellen effiziente IT-Services bereit und begleiten IT-Projekte jeder Größenordnung in den Bereichen IT-Services, -Solutions und –Consulting. Als 100%ige Daimler-Tochter garantieren wir unseren Kunden im Konzern eine hohe Identifikation mit ihren individuellen Aufgabenstellungen. Unsere innovative Technologie- und Methoden-Kompetenz und ein tiefes Verständnis der internen Geschäfts-Prozesse sind unsere entscheidenden Wettbewerbsvorteile.

Unser Ziel ist es, ein relevanter Spezialist und Partner für unsere Kunden zu sein und ein attraktiver Arbeitgeber in einem innovativen und menschlich geprägten Arbeitsumfeld.

Unternehmensüberblick
Unsere Geschäftsfelder

Solutions

Kundenspezifische Individual- und Standardlösungen für alle Konzernbereiche.

Services

Leistungen auf Basis von Konzern- und Industriestandards: schnell, flexibel und kosteneffizient.

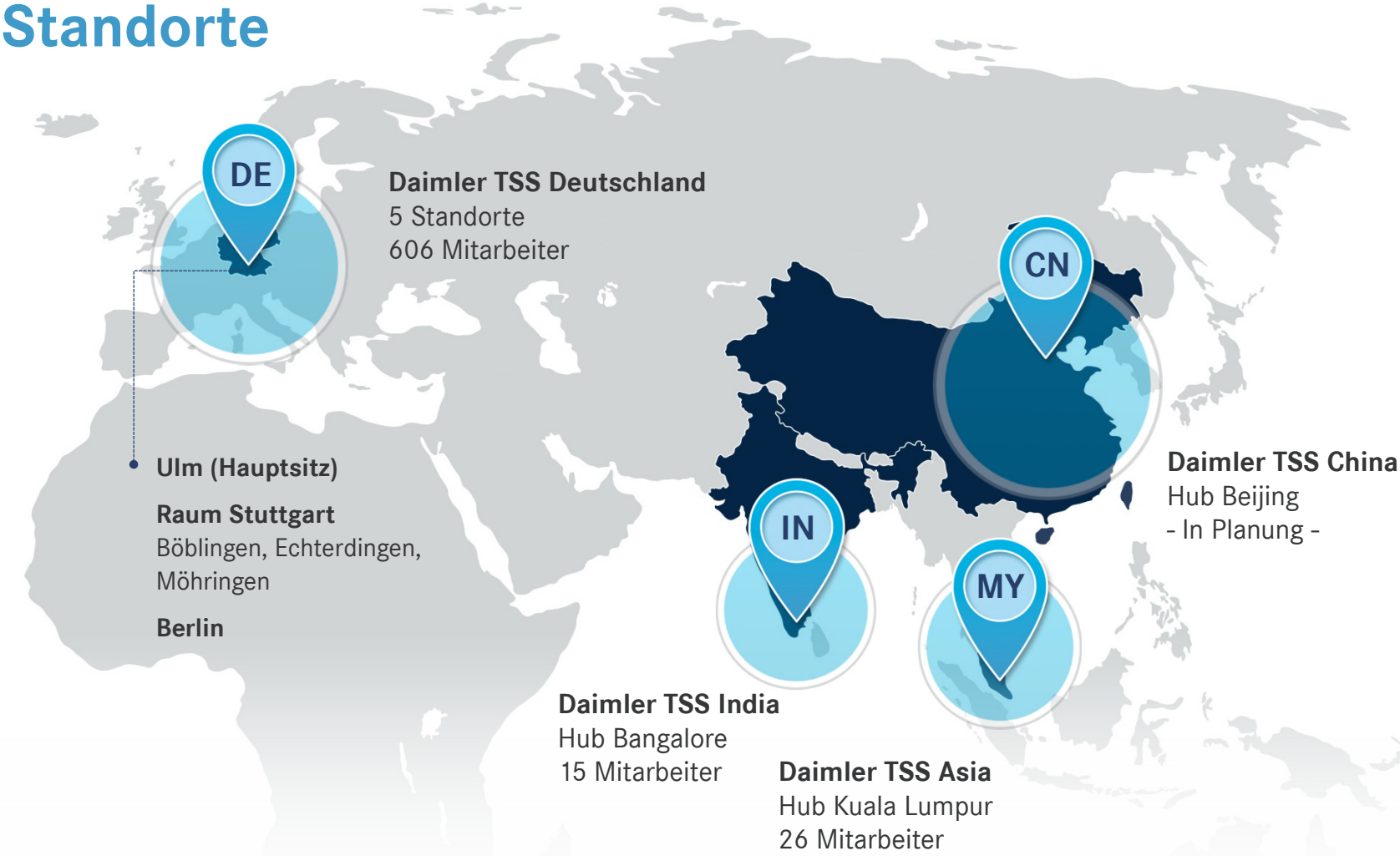
Consulting

Know-how und Kapazitäten in den Disziplinen Technologie, Strategie, Methodik, Prozesssteuerung und -qualität sowie Sicherheit.

IT Retail

Softwarelösungen, Infrastrukturen und Dienstleistungen für Händlerbetriebe werden durch unsere 100% Tochtergesell-schaft **Daimler IT Retail GmbH** erbracht.

Unternehmensüberblick
Standorte



Daimler TSS Deutschland
5 Standorte
606 Mitarbeiter

Ulm (Hauptsitz)

Raum Stuttgart
Böblingen, Echterdingen, Möhringen

Berlin

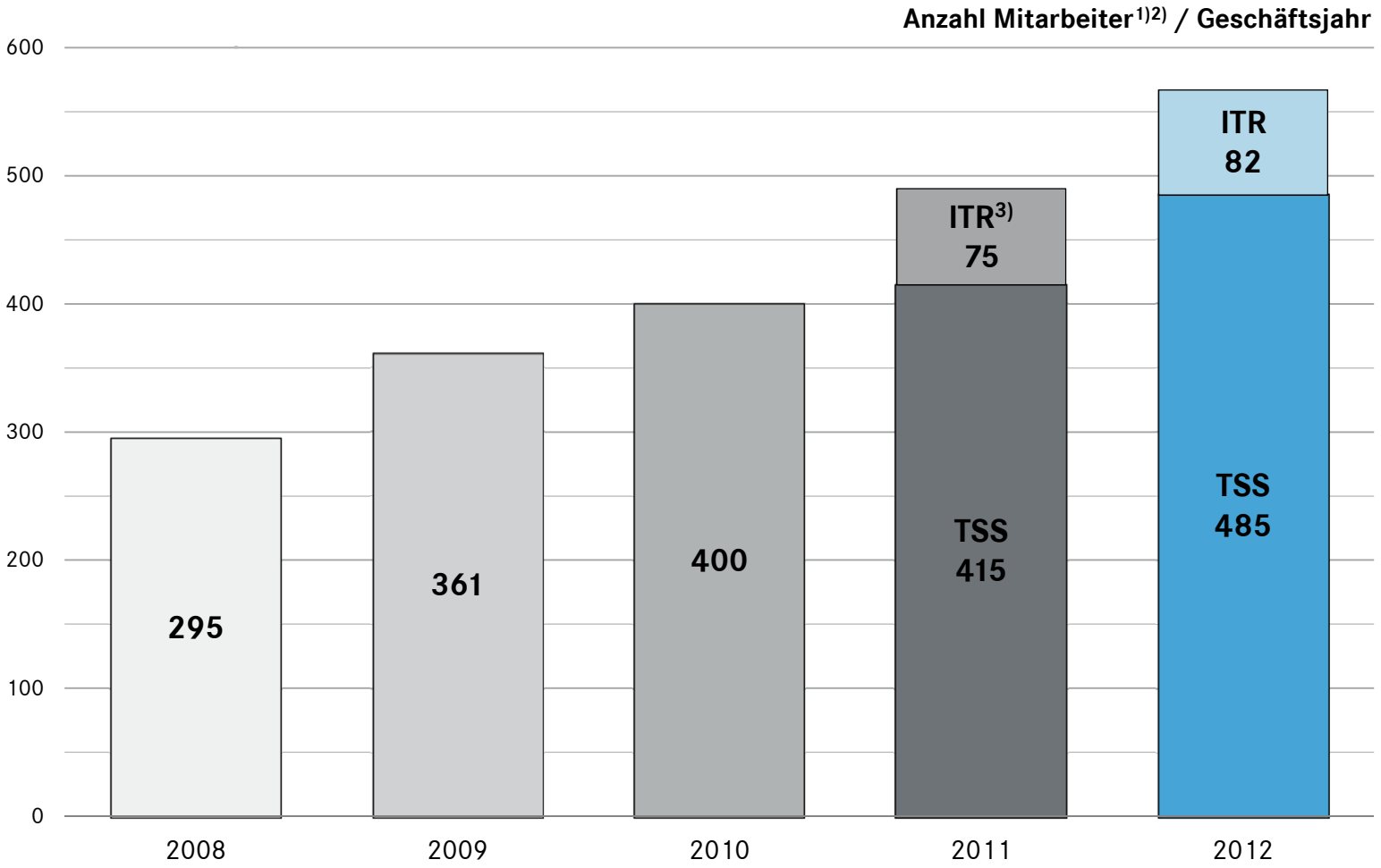
Daimler TSS China
Hub Beijing
- In Planung -

Daimler TSS India
Hub Bangalore
15 Mitarbeiter

Daimler TSS Asia
Hub Kuala Lumpur
26 Mitarbeiter

Ausgehend von den **TSS Standorten** arbeiten weitere TSS Experten in weltweiten Kundenprojekten „onsite“.

Unternehmensüberblick
Mitarbeiterentwicklung 2008 – 2012



[1)] Jahresdurchschnitt
[2)] ohne KL und Studenten
[3)] Gesamtjahr

Fehlerbehandlung mittels DML Error Logging
Überblick

1. ETL Verarbeitung
2. DML Error Logging
3. Performance DML Error Logging
4. Erfahrungen mit DML Error Logging
5. Zusammenfassung

Fehlerbehandlung mittels DML Error Logging
Überblick

1. ETL Verarbeitung
2. DML Error Logging
3. Performance DML Error Logging
4. Erfahrungen mit DML Error Logging
5. Zusammenfassung

ETL Verarbeitung

Codebeispiel mengen- und zeilenbasierte Verarbeitung

Mengenbasierte Verarbeitung

```
INSERT INTO target
SELECT id, name, code, ... FROM source;
COMMIT;
```

Zeilenbasierte Verarbeitung

komplexer,
langer Code

ETL Verarbeitung

Mengenbasierte und zeilenbasierte Verarbeitung

Mengenbasierte Verarbeitung

- Einfache SQL Anweisungen
- Effiziente Verarbeitung
- Rollback bei Fehlern
- Fehlerbereinigung separat

Zeilenbasierte Verarbeitung

- Flexible Fehlerbehandlung
- Datentransport und –bereinigung in einem Schritt
- Komplexer Code
- (Langsamere) Schleifenverarbeitung

ETL Verarbeitung

Alternativen bei einer mengenbasierten Verarbeitung

Subsetting

- Selektion der fehlerfreien Datensätze

Pre-Processing

- Korrektur der fehlerhaften Datensätze vorab

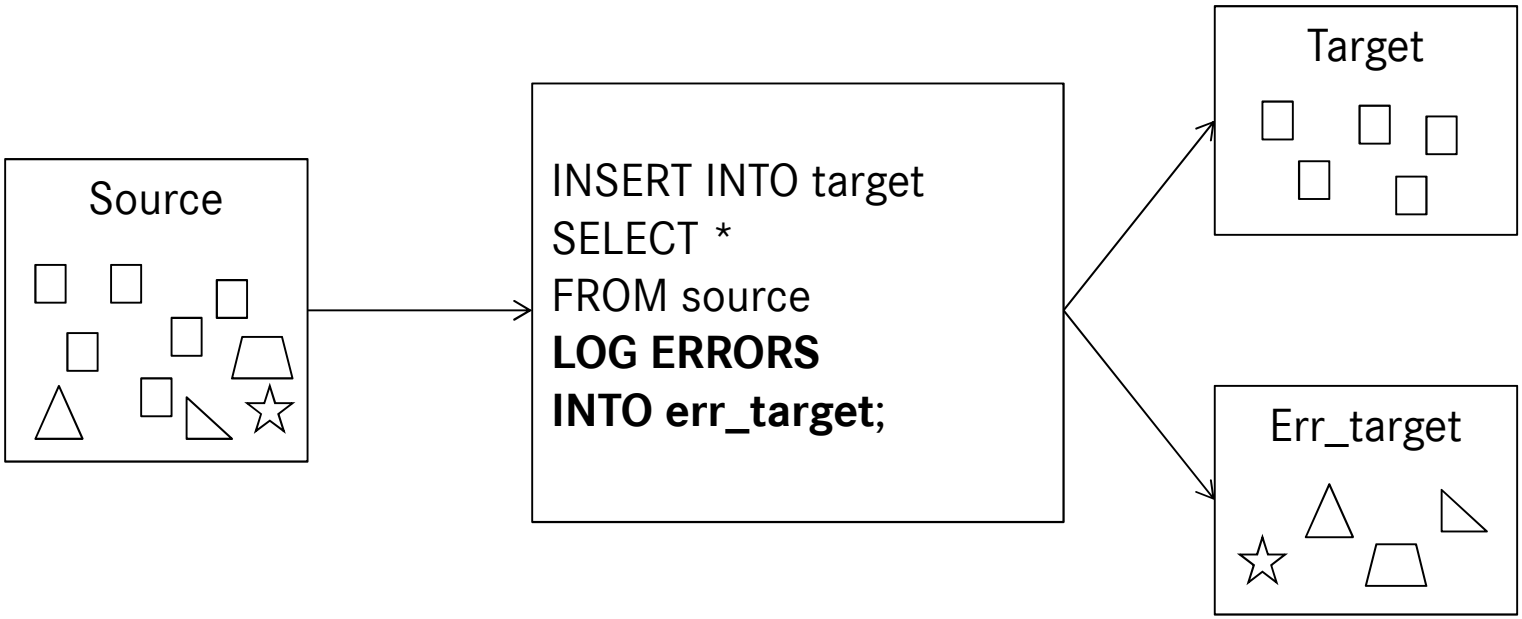
DML Error Logging

- Protokollierung der fehlerhaften Datensätze durch DML Error Log-Klausel

Fehlerbehandlung mittels DML Error Logging
Überblick

1. ETL Verarbeitung
2. DML Error Logging
3. Performance DML Error Logging
4. Erfahrungen mit DML Error Logging
5. Zusammenfassung

DML Error Logging
Überblick



DML Error Logging
Aufbau Fehlertabelle

SQL> DESC target

Name	Null?	Typ
ID	NOT NULL	NUMBER
CODE	NOT NULL	VARCHAR2(10)

SQL> DESC err$_target

Name	Null?	Typ
ORA_ERR_NUMBER$		NUMBER
ORA_ERR_MESG$		VARCHAR2(2000)
ORA_ERR_ROWID$		ROWID
ORA_ERR_OPTYP$		VARCHAR2(2)
ORA_ERR_TAG$		VARCHAR2(2000)
ID		VARCHAR2(4000)
CODE		VARCHAR2(4000)

DML Error Logging
Erzeugung Fehlertabelle 1(2)

```
DBMS_ERRLOG.CREATE_ERROR_LOG
        ( dml_table_name IN VARCHAR2
        , err_log_table_name IN VARCHAR2 := NULL
        , err_log_table_owner IN VARCHAR2 := NULL
        , err_log_table_space IN VARCHAR2 := NULL
        , skip_unsupported IN BOOLEAN := FALSE);
```

Aufruf:

```
exec dbms_errlog.create_error_log ('TARGET', 'ERR_TARGET');
```

DML Error Logging
Erzeugung Fehlertabelle 2(2)

```
CREATE TABLE ERR$_TARGET (
                ORA_ERR_NUMBER$ NUMBER
              , ORA_ERR_MESG$ VARCHAR2(2000)
              , ORA_ERR_ROWID$ ROWID
              , ORA_ERR_OPTYP$ VARCHAR2(2)
              , ORA_ERR_TAG$ VARCHAR2(2000)
              , ID VARCHAR2(4000)
              , CODE VARCHAR2(4000)
);
```

DML Error Logging
Beispiel Verarbeitung fehlerhafter Daten

```
INSERT INTO target
SELECT * FROM source;
```

*SELECT **

* * ERROR at line 2: ORA-01400: cannot insert NULL into ("TEST".“TARGET"."CODE")*

```
INSERT INTO target
SELECT * FROM source
LOG ERRORS INTO err$_target ('INSERT') REJECT LIMIT UNLIMITED;
```

99998 rows created.

Vorsicht: Defaultwert ist 0
Wert gilt für jeden paralellen Prozess

DML Error Logging
Datensicherheit

dbms_errlog.create_error_log übernimmt beim Erstellen der Error Log Tabelle keine sicherheitsspezifischen Konfigurationen, z.B.:

- Virtual Private Database Policies auf Tabellen
- Verschlüsselung von Spalten

Sicherheitsspezifische Konfigurationen müssen jeweils manuell korrigiert werden, z.B.

- VPD Policies auch auf Error Log Tabellen manuell anlegen
- Verschlüsselung von Spalten auch auf Error log Tabellen manuell anlegen

Fehlerbehandlung mittels DML Error Logging
Überblick

1. ETL Verarbeitung
2. DML Error Logging
3. Performance DML Error Logging
4. Erfahrungen mit DML Error Logging
5. Zusammenfassung

Performance DML Error Logging
Oracle Trace

```
insert into target select id  from source log errors into errtarget reject
  limit unlimited
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          1          0           0
Execute      1      0.01       0.01          7         15        379          91
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        2      0.01       0.01          7         16        379          91
```

```
INSERT INTO "ERRTARGET" (ORA_ERR_NUMBER$, ORA_ERR_MESG$, ORA_ERR_ROWID$,
  ORA_ERR_OPTYP$, ORA_ERR_TAG$, "ID")
VALUES
 (:1, :2, :3, :4, :5, :6)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      9      0.00       0.00          2          4         49           9
Fetch        0      0.00       0.00          0          0          0           0
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       10      0.00       0.00          2          4         49           9
```

Performance DML Error Logging
Überblick Testaufbau

```
TRUNCATE TABLE DMLtarget1;          – DML Error Logging
TRUNCATE TABLE ERRtarget1;          – DML Error Logging
TRUNCATE TABLE DMLtarget2;          – PL/SQL Schleifenverarbeitung
TRUNCATE TABLE ERRtarget2;          – PL/SQL Schleifenverarbeitung


exec runstats_pkg.rs_start();
@@run1.sql                          – DML Error Logging
exec runstats_pkg.rs_middle();
@@run2.sql                          – PL/SQL Schleifenverarbeitung
exec runstats_pkg.rs_stop(5000);
```

Performance DML Error Logging

Testskript run1.sql (DML Error Logging)

```
INSERT /*+ NO_GATHER_OPTIMIZER_STATISTICS */ INTO DMLtarget1

SELECT id FROM DMLsource

LOG ERRORS INTO ERRtarget1 REJECT LIMIT UNLIMITED;

COMMIT;
```

Performance DML Error Logging

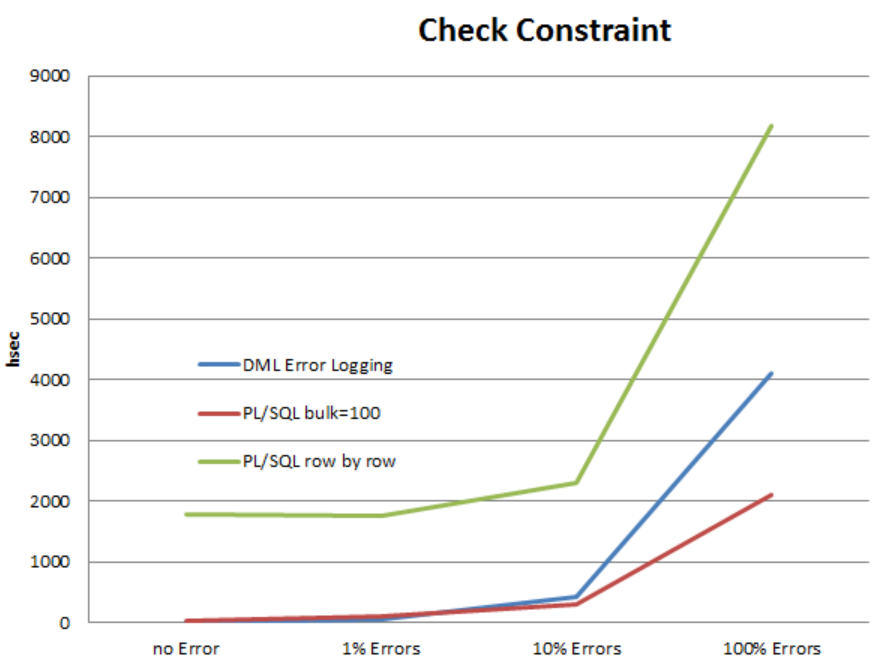
Testskript run2.sql (PL/SQL BULK-Schleifenverarbeitung)

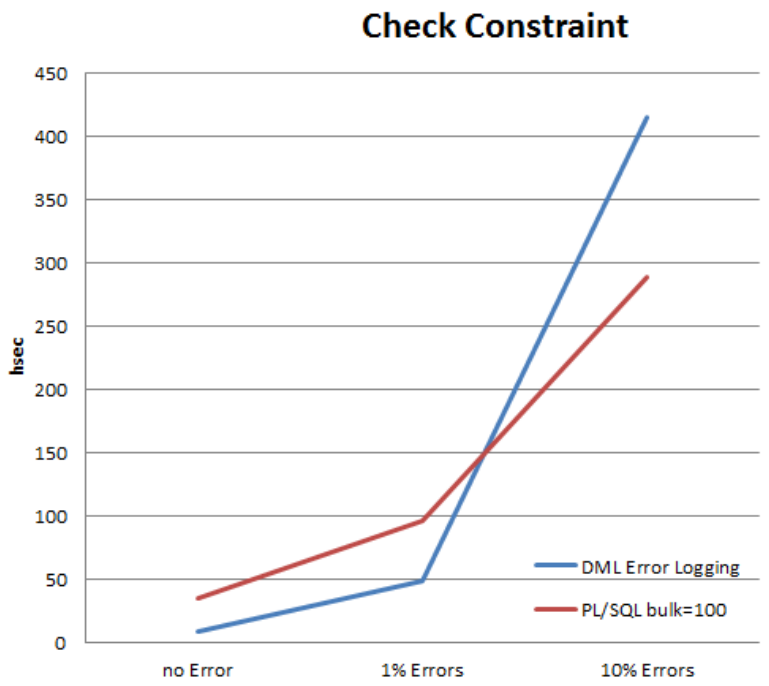
```
DECLARE
CURSOR cur IS
     SELECT *
     FROM   DMLsource;

...
 OPEN cur;
   LOOP
     FETCH cur BULK COLLECT INTO aa LIMIT 100;
     EXIT WHEN aa.COUNT = 0;
     BEGIN
       FORALL i IN INDICES OF aa SAVE EXCEPTIONS
         INSERT INTO DMLtarget2 VALUES aa(i);
     EXCEPTION
       WHEN bulk_exceptions THEN

         n := n + SQL%ROWCOUNT;
         error_logging();
     END;
   END LOOP;
   CLOSE cur;
```

Performance DML Error Logging
Testergebnis





Fehlerbehandlung mit DML Error Logging
Testfall: Daten ohne Fehler; Bug

- INSERT ohne Error Logging und INSERT mit Error Logging-Klausel sind etwa gleich schnell
- Bug 11865420 “Insert as Select with LOG ERRORS INTO slower than expected”

Affects:

Product (*Component*)	Oracle Server (Rdbms)
Range of versions *believed* to be affected	Versions BELOW 12.1
Versions *confirmed* as being affected	• 11.2.0.3 • 11.2.0.2
Platforms affected	Generic (all / most platforms affected)

Fixed:

This issue is fixed in	• 12.1.0.1 (Base Release) • 11.2.0.4 (Future Patch Set) • 11.2.0.3 Patch 2 on Windows Platforms • 11.2.0.2 Patch 12 on Windows Platforms

Fehlerbehandlung mittels DML Error Logging
Überblick

1. ETL Verarbeitung
2. DML Error Logging
3. Performance DML Error Logging
4. Erfahrungen mit DML Error Logging
5. Zusammenfassung

Fehlerbehandlung mit DML Error Logging
Überblick

Codegenerator

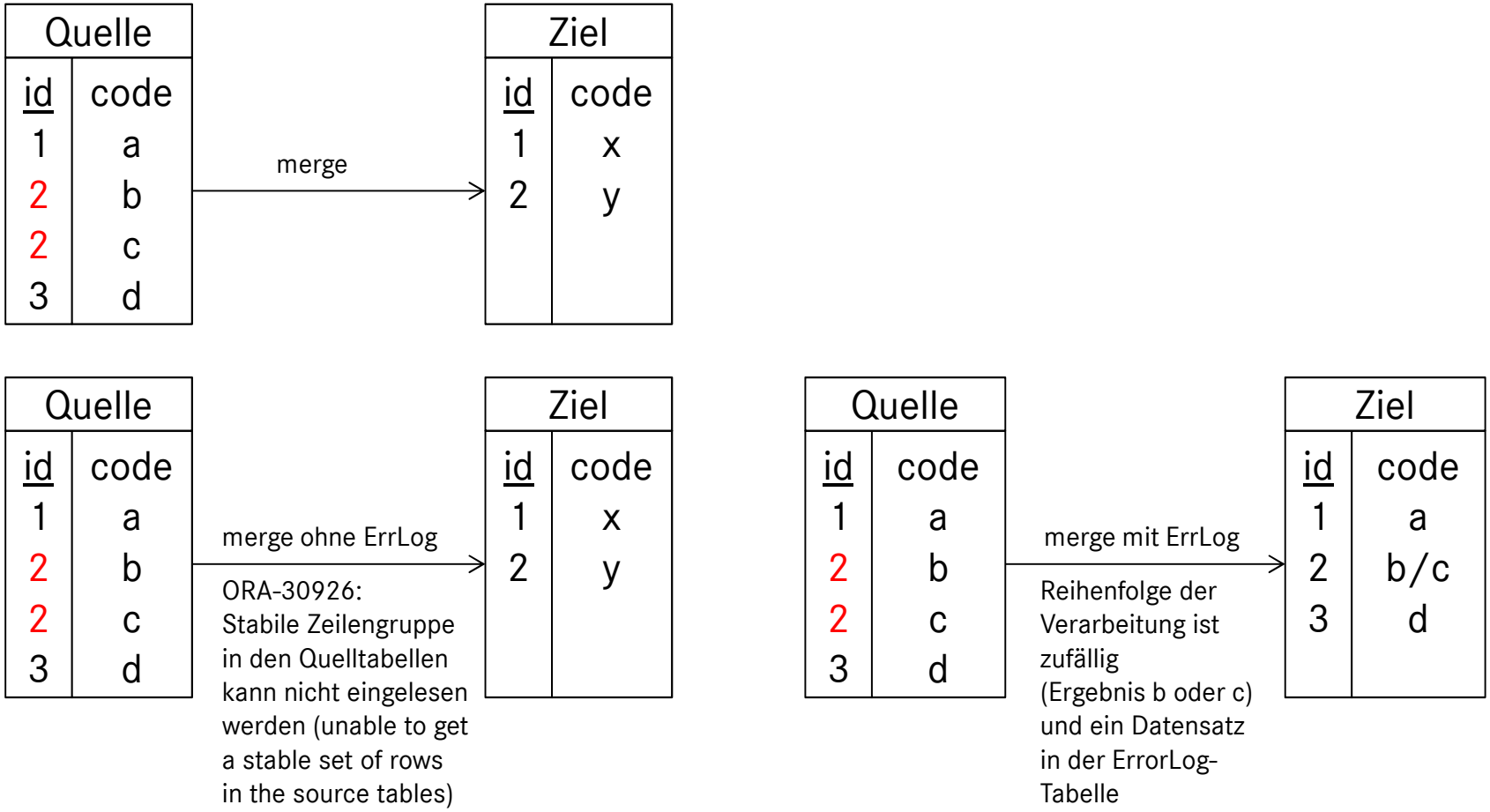
- DB: Oracle RAC 11.2.0.3 Patch 2
- ETL: Informatica
- PL/SQL-Package: 1500 loc
- Typische Einträge in DML Error Log-Tabellen
  - ORA-01427: Unterabfrage für eine Zeile liefert mehr als eine Zeile
  - ORA-01400: Einfügen von NULL in (<Tabelle>) nicht möglich
  - ORA-00001: Unique Constraint (<Constraint>) verletzt
- Einträge in DML Error Log-Tabellen werden nach 30 Tagen geleert

Fehlerbehandlung mit DML Error Logging
Dokumentierte Einschränkungen

- LONG, CLOB, BLOB, BFILE und ADT Datentypen dürfen nicht in der Error Log-Tabellen enthalten sein
- Deferred Constraints dürfen nicht auf der Zieltabelle definiert sein
- Direct-path INSERT oder MERGE Befehle, die einen eindeutigen Constraint (unique constraint) oder eindeutigen Index (unique index) verletzen
- UPDATE Befehle, die einen eindeutigen Constraint (unique constraint) oder eindeutigen Index (unique index) verletzen

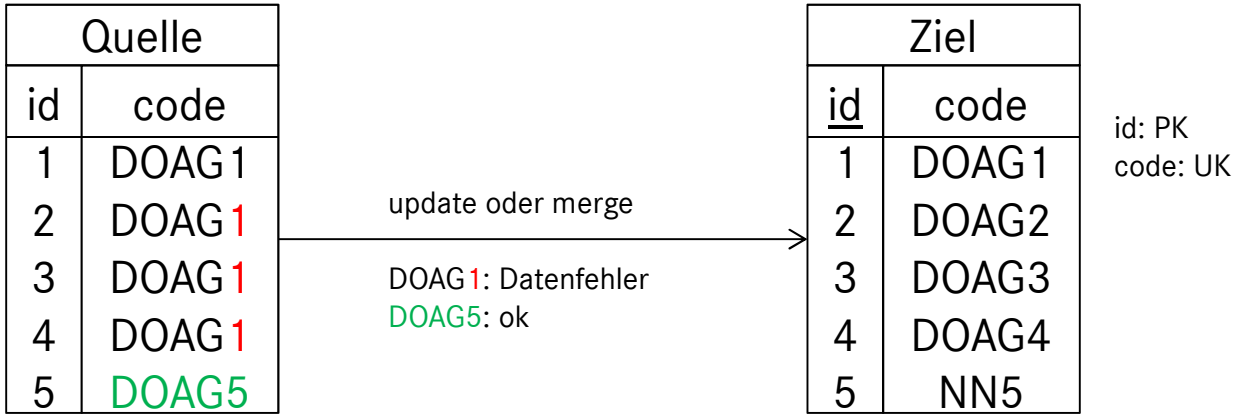
Fehlerbehandlung mit DML Error Logging

Reihenfolge der Verarbeitung bei Dubletten



Fehlerbehandlung mit DML Error Logging

Unique Constraint bei Update und Merge 1(4)

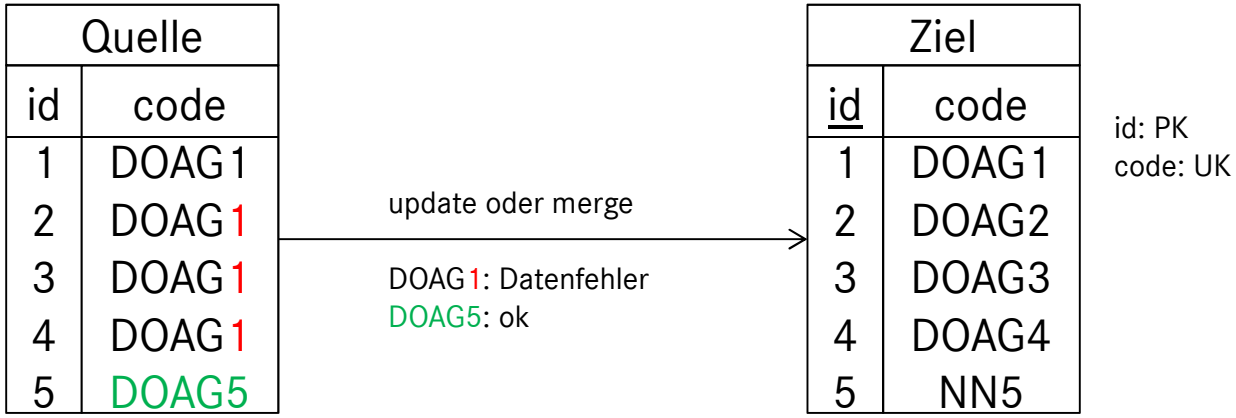



```
UPDATE target
SET code = ( select code from source where target.id = source.id )
LOG ERRORS INTO err_target ('UPDATE') REJECT LIMIT UNLIMITED;
```

```
MERGE INTO target
USING source
ON (target.id = source.id)
WHEN MATCHED THEN
        UPDATE SET target.code = source.code
LOG ERRORS INTO err_target ('MERGE') REJECT LIMIT UNLIMITED;
```

Fehlerbehandlung mit DML Error Logging

Unique Constraint bei Update und Merge 2(4)




```
UPDATE target
SET code = ( select code from source where target.id = source.id )
LOG ERRORS INTO err_target ('UPDATE') REJECT LIMIT UNLIMITED;
```

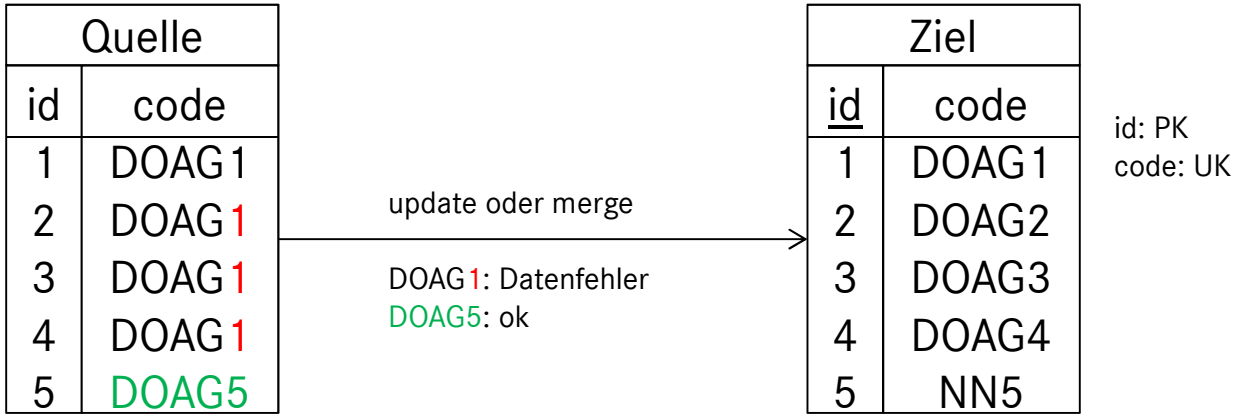
```
*
FEHLER in Zeile 1:
ORA-00001: Unique Constraint (ABU.TARGET_CODE_UIX) verletzt
```

```
MERGE INTO target
USING source
ON (target.id = source.id)
WHEN MATCHED THEN
        UPDATE SET target.code = source.code
LOG ERRORS INTO err_target ('MERGE') REJECT LIMIT UNLIMITED;
```

5 Zeilen integriert.

Fehlerbehandlung mit DML Error Logging

Unique Constraint bei Update und Merge 3(4)



UPDATE Ergebnis

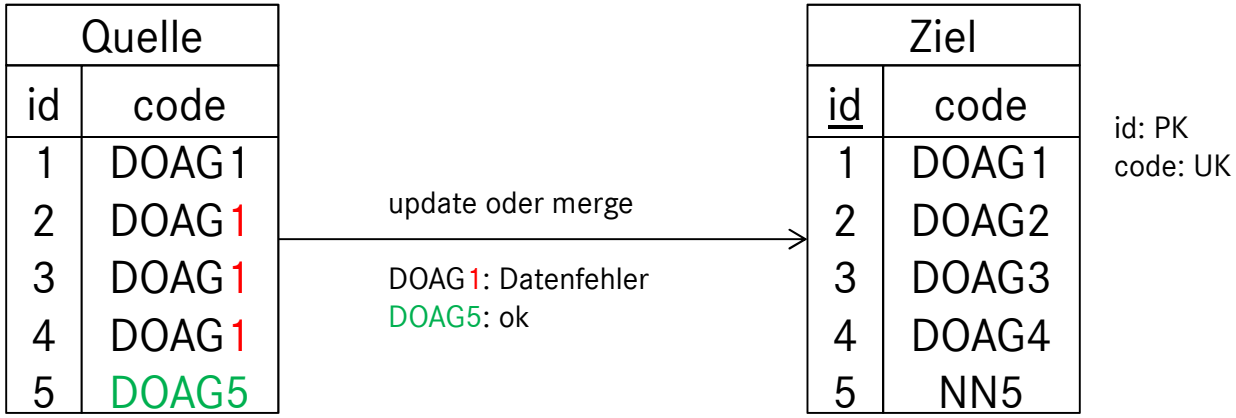
Ziel	
id	code
1	DOAG1
2	DOAG2
3	DOAG3
4	DOAG4
5	NN5

MERGE Ergebnis

Ziel	
id	code
1	DOAG1
2	DOAG2
3	DOAG3
4	DOAG4
5	DOAG5

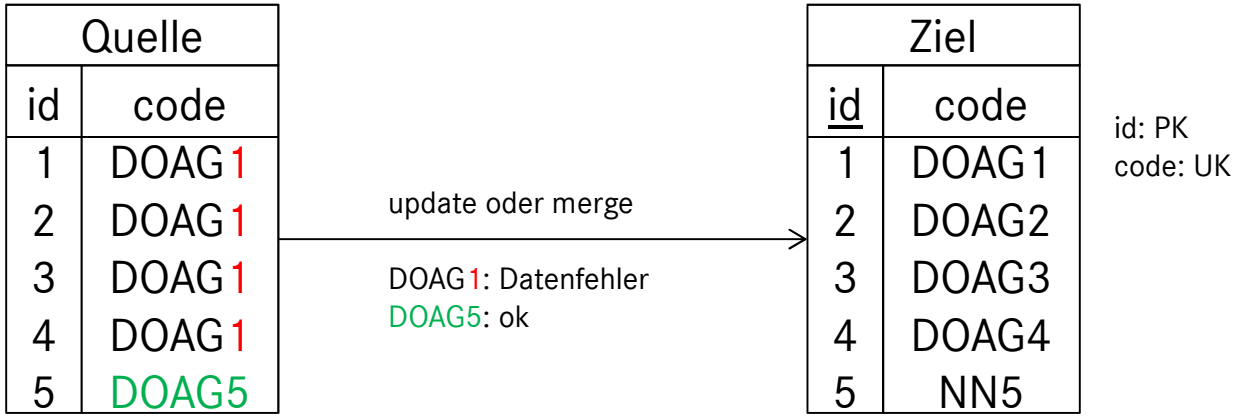
Fehlerbehandlung mit DML Error Logging

Unique Constraint bei Update und Merge 4(4)



ORA_ERR_MESG$	ORA_ERR_OPTYP$	TAG	ID	CODE
ORA-00001: Unique Constraint	U	UPDATE	3	DOAG1
ORA-00001: Unique Constraint	U	UPDATE	4	DOAG1
ORA-00001: Unique Constraint	U	MERGE	2	DOAG1
ORA-00001: Unique Constraint	U	MERGE	3	DOAG1
ORA-00001: Unique Constraint	U	MERGE	4	DOAG1

Fehlerbehandlung mit DML Error Logging
Unklare Meldungen 1(2)




```
MERGE INTO target
USING source
ON (target.id = source.id)
WHEN MATCHED THEN
        UPDATE SET target.code = source.code
LOG ERRORS INTO err_target ('MERGE') REJECT LIMIT
UNLIMITED;
```

5 Zeilen integriert.

5 Zeilen integriert. Alles ok?? Keine Fehler????

Wie viele Datensätze wurden tatsächlich erfolgreich verarbeitet (2) und wie viele Datensätze wurden in die Error Log Tabelle geschrieben (3)?

Fehlerbehandlung mit DML Error Logging
Unklare Meldungen 2(2)

```
truncate table target;
truncate table err_target;
insert into target values(1, '1');
insert into target
select *
from
(
  select rownum as id, to_char(rownum) as code
  from dual
  connect by level <= 10
)
LOG ERRORS INTO err_target (INSERT') REJECT LIMIT
UNLIMITED;
```

9 Zeilen erstellt.

```
select count(*) from err_target;

  COUNT(*)
_______
         1
```

9 Zeilen erstellt bedeutet hier, dass 9 Datensätze erfolgreich verarbeitet wurden und ein Datensatz in die Error Log Tabelle geschrieben wurde.

Rückmeldung/Rowcount nicht verlässlich!

Fehlerbehandlung mit DML Error Logging
Daten in der Error Log Tabelle

- Daten der Quelltabelle (ID, CODE) werden in die Error Log Tabelle geschrieben
- Daten der Zieltabelle (ROWID) werden in die Error Log Tabelle geschrieben bei update oder delete

ORA_ERR_MESG$	ORA_ERR_ROWID$	TAG	ID	CODE
ORA-00001: Unique Constraint	AAAzN2AAFAAAB1fAAC	UPDATE	3	DOAG1
ORA-00001: Unique Constraint	AAAzN2AAFAAAB1fAAD	UPDATE	4	DOAG1
ORA-00001: Unique Constraint	AAAzN2AAFAAAB1fAAB	MERGE	2	DOAG1
ORA-00001: Unique Constraint	AAAzN2AAFAAAB1fAAC	MERGE	3	DOAG1
ORA-00001: Unique Constraint	AAAzN2AAFAAAB1fAAD	MERGE	4	DOAG1

Fehlerbehandlung mit DML Error Logging
Rollback während Verarbeitung

Autonome Transaktionen bei der Protokollierung von Fehlern:

Rollback des DML Error Logging-Befehls (z.B. Serverausfall, Out-of-Space-Fehler, usw.) führt zu einem unvollständigen Rollback

- Rollback der Datentabellen
- Daten in der Error Log Tabelle bleiben erhalten

Fehlerbehandlung mit DML Error Logging
Maintenance der Error Log-Tabelle

Error Log Tabellen können sehr groß werden

- Größe/Wachstum überwachen
- Regelmäßig leeren
- Bei schlechter Datenqualität am besten Tabellen partitionieren (Tabellen manuell partitioniert anlegen und nicht über dbms_errlog-Package) und in eigenen Tablespace
- Wird die Zieltabelle gelöscht (drop table target), so bleibt die Error Log-Tabelle erhalten (Error Log Tabelle muss manuell gelöscht werden)

Fehlerbehandlung mittels DML Error Logging
Überblick

1. ETL Verarbeitung
2. DML Error Logging
3. Performance DML Error Logging
4. Erfahrungen mit DML Error Logging
5. Zusammenfassung

Zusammenfassung
Ausblick

- **Größter Vorteil:** einfacher, wartbarer Code im Vergleich zu PL/SQL-Schleifenverarbeitung
- Verfügbar seit 10gR2, jedoch bisher eher wenig genutzt
- DML Error Logging geeignet, wenn Daten wenig Fehler aufweisen (< 5-10%), ansonsten größerer Performanceverlust
- DML Error Logging hat sich bewährt in einem Projekt für einen Codegenerator
- Man muss mit Oracle Bugs rechnen bei nicht-alltäglichen Fehlern
- Einschränkungen bei eindeutigen Constraints (unique constraint) bzw. eindeutigen Indexen (unique index) bei UPDATE
- Organistorische Regelung nötig, wie mit den Daten in der Fehlertabelle umgegangen wird

Vielen Dank!

Daimler TSS GmbH

Wilhelm-Runge-Straße 11
89081 Ulm
Telefon +49 731 505-06
Fax +49 731 505-65 99
tss@daimler.com

Internet: www.daimler-tss.com
Intranet: intra.corpintra.net/intra-itc/tss
Intranet-Portal-Code: @TSS

Daimler TSS GmbH
Sitz und Registergericht: Ulm, HRB-Nr.: 3844
Geschäftsführung: Dr. Stefan Eberhardt (Vorsitzender), Steffen Bäuerle