Der perfekte 12c Error Handler

Sven-Uwe Weller
syntegris information solutions GmbH
Neu Isenburg

Schlüsselworte
PL/SQL, Errorstack, Callstack, Error Backtrace, Optimization levels, utl_callstack,

Einleitung
Neue Datenbankversionen bringen neue Features mit. In Falle von Oracle DB 12c ist es unter anderem das Package `UTL_CALLSTACK`. Der Vortrag stellt Überlegungen an, was einen guten Error Handler ausmacht und wie sich dies erreichen lässt. Dabei kann `UTL_CALLSTACK` Informationen bereitstellen, die es so bisher nicht gab. Die kann die Art und Weise verändern, wie wir einen vernünftigen Error Handler schreiben. Ob dabei schon der „perfekte" Error Handler entsteht, muss jeder am Ende für sich selbst entscheiden.

Was erwarten wir von einem Error Handler?
Bei Software versucht man meist mehrere konkurrierende Ziele unter einen Hut zu bringen. Dies sind unter anderem Korrektheit, Performanz, Wartbarkeit. Für einen Error Handler ergeben sich aber ein paar Besonderheiten.

Korrektheit

Im Fehlerfall möchten wir möglichst genau informiert werden. Welcher Fehler ist aufgetreten. An welcher Code-Stelle ist der Fehler passiert. Welche Daten (Runtime Variablen, Konfig Parameter, etc.) waren zu dem Zeitpunkt wie gesetzt.

Der Fehler sollte auch mit all diesen Informationen vollständig gespeichert (error log) werden.

Performanz

Im Fehlerfall spielt die Performanz meist eine extrem untergeordnete Rolle. Allerdings sollte der Code den wir zum Instrumentieren verwenden, möglichst keine zusätzliche Laufzeit kosten, solange alles glatt läuft. Hier unterscheiden sich die Anforderungen extrem z.b. im Vergleich zu Trace Logs.

Wartbarkeit

Minimaler Programmieraufwand für möglichst viel Resultat.

Wartbarkeit ist eine häufig unterschätzte Eigenschaft. Im Bereich Fehlerhandling bedeudet Wartbarkeit, das auch wenn kein expliziter Exception Handler vorhanden ist, trotzdem alle Informationen zugänglich sein sollten. Es bedeudet auch,d as möglichst keine redundanten Infromationen weggeschrieben werden sollen. Ansonsten verwirrt das den Entwickler, der versucht einen bestimmten Fehler zu finden.

Wie sieht ein typischer 11g Error Handler aus?

```
when others then
  logger.log_error(module=>'myPackage.myProcedure', parms=>parms);
  raise;
```

Die verschiedenen Stacks erklärt.

Es gibt in der Oracle Datenbank mehrere Stacks, die beim Programmablauf gefüllt werden. Diese Stacks waren bisher nur über `DBMS_UTILITY` zugänglich. Mit 12c und dem Package `UTL_CALLSTACK` hat sich das deutlich geändert.

- Call Stack – Welche Module wurden bisher aufgerufen.
- Error Stack – Welche Fehler sind aufgetaucht.
- Error Backtrace Stack – Wo ist der Fehler ursprünglich aufgetaucht, bis er behandelt wurde.

Es wird simuliert bzw. demonstriert, wie beim Programmablauf die verschiedenen Stacks befüllt werden. Und zusätzlich wie sich die Angaben zur `Dynamic_Depth`, `Lexical_Depth`, `Error_Depth` und `Backtrace_Depth` verhalten.

Optimization Levels

Vielen Programmierern ist nicht bewusst, das das Optimization Level Einfluß auf den Callstack (und konsequenterweise auch auf den Errorstack und das Error Backtrace) hat. Ursache ist das der Optimizer, den Plsql code umbauen kann. Dabei sind die Änderungen bei Optimization Level 2 meist relativ gering. Bei Optimization Level = 3 werden Module aber „inlined“ und ggf. sogar komplett aus dem Code entfernt. Der damit erzeugte Code läuft teilweise dramatisch schneller. Allerdings ist das Nachvollziehen von Fehlern deutlich schwieriger geworden.

Ein Trick hier ist die Verwendung von Edition Based Redefinition. Damit kann die Package in einem geringeren Optimization Level (=1) nochmal auf den gleichen Daten gestartet werden. Und der Entwickler kann nachvollziehen, wo genau der Fehler passiert ist. Dies ist aber nur in speziellen Umgebungen machbar.

Worst practices

Eine der häuftigst verbreitesten Praktiken, ist es in jedes plsql Modul einen `when others exception` handler zu schreiben. Dies ist absolut überflüssig. Entweder wird der gleiche Fehler einfach nur wiederholt. Dies macht die Fehlersuche extrem schwierig.

In der Vergangenheit wurde das oft gemacht, damit man den Namen des Moduls in den Logger Eintrag übertragen kann. Mit 12c und utl_callstack ist das überholt. Diese Information steht nun auch zur Auswertung zur Verfügung und muss nicht mühsam programmiert und eingefügt werden.

Es gibt noch viele andere Worst Practices. Diese sind aber oft bekannt. Mit 12c sollte man sich jedoch die aktuelle Fehlerstrategie anschauen und neu überdenken.

Best practices

Alle öffentlich zugänglichen Module sollten einen allgemeinen Errorhandler bekommen.
Alle anderen brauchen nur dann einen extra Errorhandler, wenn zusätzliche Informationen beim Fehler weggeschrieben werden MÜSSEN. Hier bieten sich zwei unterschiedliche Strategien an.

```
when others then
  raise_application_error(-20001, <Message>, true);
```

oder

```
when others then
  logger.log_error(parms=>parms);
  raise;
```

Beide Strategien haben Vor- und Nachteile. Im ersten Fall wird nur der Errorstack ergänzt. Wenn die gewünschten Zusatzinformationen in die Error Message aufgenommen werden können, dann sollte dies immer Vorrang bekommen.

Hier ergeben sich auch Querverbindungen zum Tracing.

Verwendung von Logger

Logger ist eines der beliebtesten Logging Frameworks. Vor allem weil es sehr schnell ist und eine ausreichende Komfortabilität bietet. Alelrdings ist Logger noch nicht auf 12c eingestellt. U.a. wird im Error Fall der Callstack nicht mehr weggeschrieben. Es wird der Errorstack und das Error Backtrace kombiniert und in die Callstack Spalte geschrieben. Dies ist in Fehlerfall durchaus sinnvoll. Jedoch könnten damit wertvolle Informationen verlorengegangen sein.

Kontaktadresse:
Sven-Uwe Weller
Syntegris information solutions GmbH
Hermanstraße 54-56
D-63263 Neu-Isenburg

Telefon:	+49 (0) 160-7465015
E-Mail	sven.weller@syntegris.de
Internet:	www.syntegris.de
Blog:	svenweller.wordpress.com