Möglichkeiten zur Prozesskontrolle in Datenbank-Batchverarbeitungen

Dr. Kurt Franke

Cellent Finance Solutions GmbH

Kurt.Franke@cellent-fs.de, Kurt-Franke@web.de

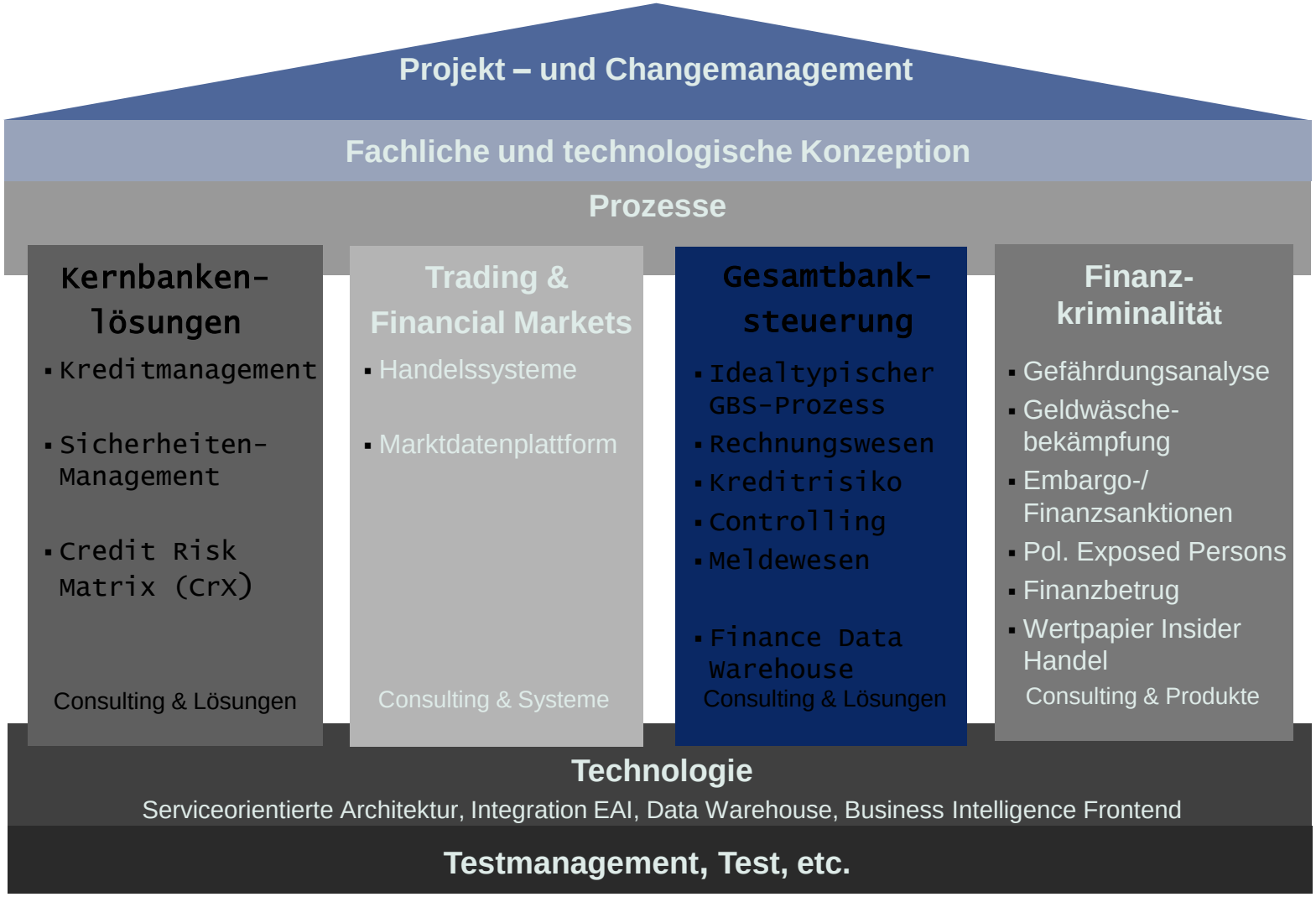
Agenda

- cfs Kurzvorstellung
- Batchbetrieb in einer DB-Applikation
- Prozesskontrolle von DB-Sessions/Prozessen
- Methode 1: Verwendung von V$SESSION etc.
- Methode 1b: Views nur mit Applikations-Prozessen
- Methode 2: Session-Duration-Locks + eindeutige Namen
- Vergleich Methode 1 / 1b: ⇔ Methode 2
- Zusammenfassung

Zahlen und Fakten auf einen Blick

Firmensitz:	Stuttgart
Geschäftsstellen:	Frankfurt am Main, München
Branchenerfahrung:	seit 1988
Umsatz:	€ 28 Mio.
Anzahl Mitarbeiter:	200
Rechtsform:	Gesellschaft mit beschränkter Haftung
Geschäftsführer:	Thomas Wild
Gesellschafter:	100 % Landesbank Baden-Württemberg (LBBW)
Aufsichtsratsvorsitz:	Dr. Martin Setzer

Das Angebot der Cellent Finance Solutions



Bekämpfung von Finanzkriminalität

Kundenstamm von mehr als 800 Instituten in 50 Ländern

- **Geldwäschebekämpfung**
- **Gefährdungsanalyse**
- **Bekämpfung der Terrorismus-finanzierung**
- **Embargoüberwachung**
- **Kundenscreening**
- **PEP-Identifikation**
- **Bekämpfung von Finanzbetrug**
- **Wertpapier-Insiderhandel**

Referenzen SMARAGD (Auszug)

Batchbetrieb in einer DB-Applikation

- Verarbeitung durch Hintergrundjobs
  - kein spezieller Dispatcher erforderlich
    - ➢ jeder darf Job-Tasks einstellen
      - – in eine simple Tabelle für Anforderungen
    - ➢ Anstarten durch DBMS_SCHEDULER Job
      - – nach Bedarf oder zeitgesteuert (verschiedene Muster möglich)
      - – unter Berücksichtigung von Abhängigkeiten
      - – ggf. als separater DBMS_SCHEDULER Job
- Hintergrundjobs setzen Status selbst
  - durch Schreiben in eine Status-Tabelle für die Job-Tasks
  - beim Start
  - beim Beenden (eventuell auch Fehlerstatus)
  - Setzen Ende-Status eventuell nicht mehr möglich
    - ➢ bei Oracle Fatal Error (ORA-00600 etc.)
      - – Session existiert nicht mehr

Batchbetrieb in einer DB-Applikation II

- folgende Situation nach Fatal Error Hintergrundjob
  - Verarbeitung hat sich als „aktiv“ eingetragen
  - die verarbeitende Session existiert nicht mehr
  - und der Status der Verarbeitung ist nicht „beendet“
- abgebrochene Session nicht einfach erkennbar
  - Welche Session hat die Verarbeitung gestartet ?
  - Existiert diese Session zum aktuellen Zeitpunkt noch ?
  - nicht erkennbar, dass ein Problem existieren könnte !
    - ➔ Benachrichtigung DBA keine Option
  - Methodik notwendig, die diese Situation erkennt
    - und in geeigneter Form handelt
    - Überprüfung auf solche Situation
      - vor Anstarten der nächsten Job-Task
      - für besseren Fehlerzeitpunkt auch zeitgesteuert (z. B. 1 min)

Prozesskontrolle von DB-Sessions/Prozessen

- Prüfung Existenz von Session / Process
  - Möglichkeit 1: directe Verwendung von Oracle V$-Views
    - V$SESSION, V$PROCESS etc.
    - nur mit Zugriffsrechten für DB-Applikations-User
    - Verarbeitungsstart: Identifizierungsdaten in Statustabelle
      - SID, SPID, LOGON_TIME, USER_NAME (bei verschiedenen), …
    - zusätzlich Verwendung von dbms_application_info sinnvoll
    - Join erlaubt dann umfassendes Listing der aktiven Sessions
  - Möglichkeit 1b: Views basierend auf Oracle V$-Views
    - mit Einschränkung auf Applikations-Sessions / Processe
    - z. B. über Username(s) der Verarbeitungen
    - erhöhte Sicherheit, weil Applikation nur eigene Sessions sieht
    - mehrere Applikationen in einer DB komplett trennbar
    - ansonsten gilt das gleiche wie bei Möglichkeit 1

Prozesskontrolle von DB-Sessions/Prozessen II

- Möglichkeit 2: Verwendung Locks mit Session-Duration
  - diese Locks verschwinden automatisch bei Abbruch der Session
  - 1 : 1 Zuordnung zu Verarbeitung über berechneten Lock-Namen
  - ermöglicht eindeutige Rückschlüsse:
    - Lock existiert ➔ Verarbeitung ist aktiv
    - Lock existert nicht ➔ Verarbeitung nicht aktiv
      - wenn zuvor gestartet, dann abgebrochen
  - Session-Listing für DB-Applikation nicht möglich
    - nur Einzelprüfung ausgehend von Verarbeitung
  - benötigt Zugriffsrechte auf dbms_lock für DB-Applikations-User
  - zusätzlich Verwendung von dbms_application_info sinnvoll
    - für DBA-checks
  - Listing mit DBA-Rechten möglich

Methode 1: Verwendung von V$SESSION etc.

- umfassende Überwachung Sessions / Processe möglich
- alle Sessions einer DB-Applikation identifizierbar
- Voraussetzung: Zugriffsrechte auf V$-Views
- eindeutige Zuordnung Verarbeitung zu Session
  - wegen Wiederverwendung SID weitere Attribute notwendig
  - SPID aus V$PROCESS
  - LOGON_TIME
  - eigene Session über V$MYSTAT
  - dbms_application_info.set_module():
    - verarbeitungs-spezifischer Name
    - Zurücksetzen am Verarbeitungsende ermöglicht Session Re-use
    - z.B. für Tests

Methode 1: Verwendung von V$SESSION etc.

- Setzen / Zurücksetzen dbms_application_info - Module

```
BEGIN
  dbms_application_info.read_module(saved_module_name,saved_action_name);
  appl_info_read := TRUE;
  dbms_application_info.set_module('BATCH: ' || jobnam, '');
  -- hier findet die Verarbeitung statt
  IF appl_info_read THEN
    dbms_application_info.set_module(saved_module_name);
  END IF;
EXCEPTION WHEN OTHERS THEN
  IF appl_info_read THEN
    dbms_application_info.set_module(saved_module_name);
  END IF;
END;
/
```

Methode 1: Verwendung von V$SESSION etc.

- Bereitstellung von Applikations-Views
  - basierend auf Job-Status-Tabelle, V$SESSION, V$PROCESS, …
    - ➢ einschließlich Module aus V$SESSION
    - ➢ Join trifft nicht mehr zu, wenn Module zurückgesetzt wurde
  - JOB_RUNNING, JOB_COMPLETED, JOB_BROKEN, …
    - ➢ nur die letzte Verarbeitung eines Jobs erscheint in einem der Views
    - ➢ ermöchlichen Administration der Applikation
      - – ohne dass DBA-Rechte benötigt werden
      - – allein durch Zugriff auf dies Views alle Job-Stati erkennbar
      - – korrigierendes Handling über Aufrufe von Start-Package

Methode 1: Verwendung von V$SESSION etc.

- keine Verarbeitungslocks für Processkontrolle notwendig
  - In den V$-Views ist aktueller Stand abgebildet
  - keine sepziellen Aufrufe zu Überprüfung notwendig
- beliebige andere Absicherung gegen Mehrfachstart
  - z. B. zuerst Exclusivlock auf Job-Statustabelle
  - Prüfung, ob gewünschter Job bereits läuft
  - Erzeugen, wenn nicht
  - Exclusivlock wieder freigeben

Methode 1b: Views nur mit Applikations-Prozessen

- Einschränkung der Applikatonsrechte
  - z. B. bei mehreren DB-Applikationen in einer DB
  - Rechte nur für Sessions der eigenen Applikation
  - durch Erstellung eigene Views für jede DB-Applikation
    - basierend auf V$-Views aus Methode 1
  - in einem hochprivilegiertem Schema
    - z. B. System
      - benötigt Zugriffsrechte mit GRANT-Option
    - Einschränkung möglich über Sessions des Applikations-Users
  - Methodik ist identisch zu Variante 1
  - nur die sichtbaren Sessions sind eingeschränkt

Methode 2: Session-Duration-Locks + eindeutige Namen

- mögliches Verfahren, wenn kein Zugriff auf V$SESSION
- Object notwendig, das mit Session-Ende verschwindet
  - PL/SQL-User-Lock mit Duration = 'SESSION'
  - Zuordnung zu Verarbeitung über berechneten Lock-Namen
    - ➢ aus Art der Verarbeitung, und klassifizierenden Parametern
  - ausgehend von Verarbeitung wird der Lockname berechnet
  - und nachfolgend die Existenz des Locks geprüft
    - ➢ Lock existiert ➔ Verarbeitung aktiv
    - ➢ Lock existiert nicht ➔ Verarbeitung nicht aktiv
      - – zuvor gestartet ➔ Verarbeitung ist abgebrochen
  - Lock dient auch der Absicherung gegen gleichzeitige Mehrfachläufe
    - ➢ keine zusätzlich Methode notwendig
- nur Rechte auf Package DBMS_LOCK notwendig

Methode 2: Session-Duration-Locks + eindeutige Namen

- kein Listing der aktiven Sessions in der Applikation selbst
- auch kein Listing für Applikations-Admin möglich
- nur, wenn DBA-Rechte vorhanden sind

```
SELECT la.name,
       l.session_id AS hold_by_session_id,
       l.blocking_others,
       la.expiration
 FROM dbms_lock_allocated la,
      dba_lock l
 WHERE la.expiration > sysdate
   AND la.name LIKE 'PROCESS?_CONTROL.%' ESCAPE '?'
   AND l.lock_type = 'PL/SQL User Lock'
   AND l.mode_held = 'Exclusive'
   AND ltrim(rtrim(to_char(la.lockid,'999999999999'))) = l.lock_id1
/
```

Methode 2: Session-Duration-Locks + eindeutige Namen

- SELECT für Lock-Existence-Prüfung
  - spezifisch für Lock-Implementierung
  - Einschränkung auf Lockname-Prefix
  - vorliegender Fall benötigt nur Exclusiv-Locks
    - Statement prüft nur solche
  - bei Absicherung Batch + GUI-Pflege
    - zusätzlich Shared-Locks für GUI (unspezifisch nach Fachlichkeit)
    - für gleichzeitige Pflege durch mehrere Anwender
- vorliegendes SELECT – join mit V$SESSION
  - wenn SELECT aus Lock eine SID liefert, dann auch VSESSION
- Setzen / Zurücksetzen dbms_application_info - Module
  - besserer Überblick für DBA
  - Zurücksetzen bei Wiederverwendug der Session (wie Methode 1)
    - z.B. bei Tests

Methode 2: Session-Duration-Locks + eindeutige Namen

- als RUNNING gekennzeichnete Jobs in Statustabelle
  - müssen diesen Zustand nicht wirklich haben
  - falls ein FATAL error (ORA-00600 etc.) die Session gecrashed hat
  - damit ist die betreffende Verarbeitung blockiert
- Zustand muss explicit detectiert werden
  - möglichst zeitnah wegen Fehlerzeitpunkt
    - ➢ durch einen kurzperiodischen Schedulerjob (z. B. 1 Minute)
  - zur einfacheren Zuordnung des Tracefiles
  - und Kennzeichnung des korrekten Verarbeitungs-Zustandes
  - wird die zuvor nicht erkannte Blockierung aufgehoben
    - ➢ ein Handling kann erfolgen – automatisch oder via Benachrichtigung

Vergleich Methode 1 / 1b: ⇔ Methode 2

	V$-Views / darauf aufbauende Views	Locks mit Session-Duration
notwendige App-Privilegien	wenige V$-Views	dbms_lock Package
Admin-Privilegien	Applikations-Admin benötigt keine DBA-rechte	Session-Listing nur mit DBA-Rechten
Entwicklungs-Aufwand	vergleichbar	vergleichbar
Administratives Handling	einfach wegen Views	Package / Procedure für manuelle Ckecks
sichere Detektierung abgebrochener Session	automatisch durch V$-Views	kurzperiodischer Job (1 min)

Zusammenfassung

- Methode 1 mit V$-Views bzw. darauf basierenden Views
  - Applikations-Admin benötigt keine DBA-Rechte
  - zwischengeschaltete Views: Einschränkung auf Applikationssessions
  - für mehrere Applikationen möglich
    - ➢ diese sind voneinander getrennt - durch die Trennung erhöhte Sicherheit
  - Implementierung von Applikationsviews
    - ➢ übersichtliches vereinfachtes Handling
    - ➢ besser bei großen Applikation mit vielen Verabeitungen
- Methode 2 mit Session-Duration-Locks – keine V$-Views

  + berechnete Locknamen aus Verarbeitung
  - kein Sessionlisting in Applikation / für App-Admin ohne DBA-Rechte
  - Session-Existenz-Prüfung mit Utility
    - ➢ per kurzperiodischer Job, bei Bedarf auch manuell
    - ➢ unhandlich bei vielen Verarbeitungen

Fragen & Antworten

Vielen Dank für Ihre Aufmerksamkeit

Kontakt: **Dr. Kurt Franke**
Cellent Finance Solutions GmbH, Calwer Str. 33, D-70173 Stuttgart
Email: Kurt.Franke@cellent-fs.de , Kurt-Franke@web.de
Phone: +49-(0)711-222992676 , Mobil: +49-(0)171-7963089