Parallelisierung in Datenbank-Batchverarbeitungen

Dr. Kurt Franke

Cellent Finance Solutions AG

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

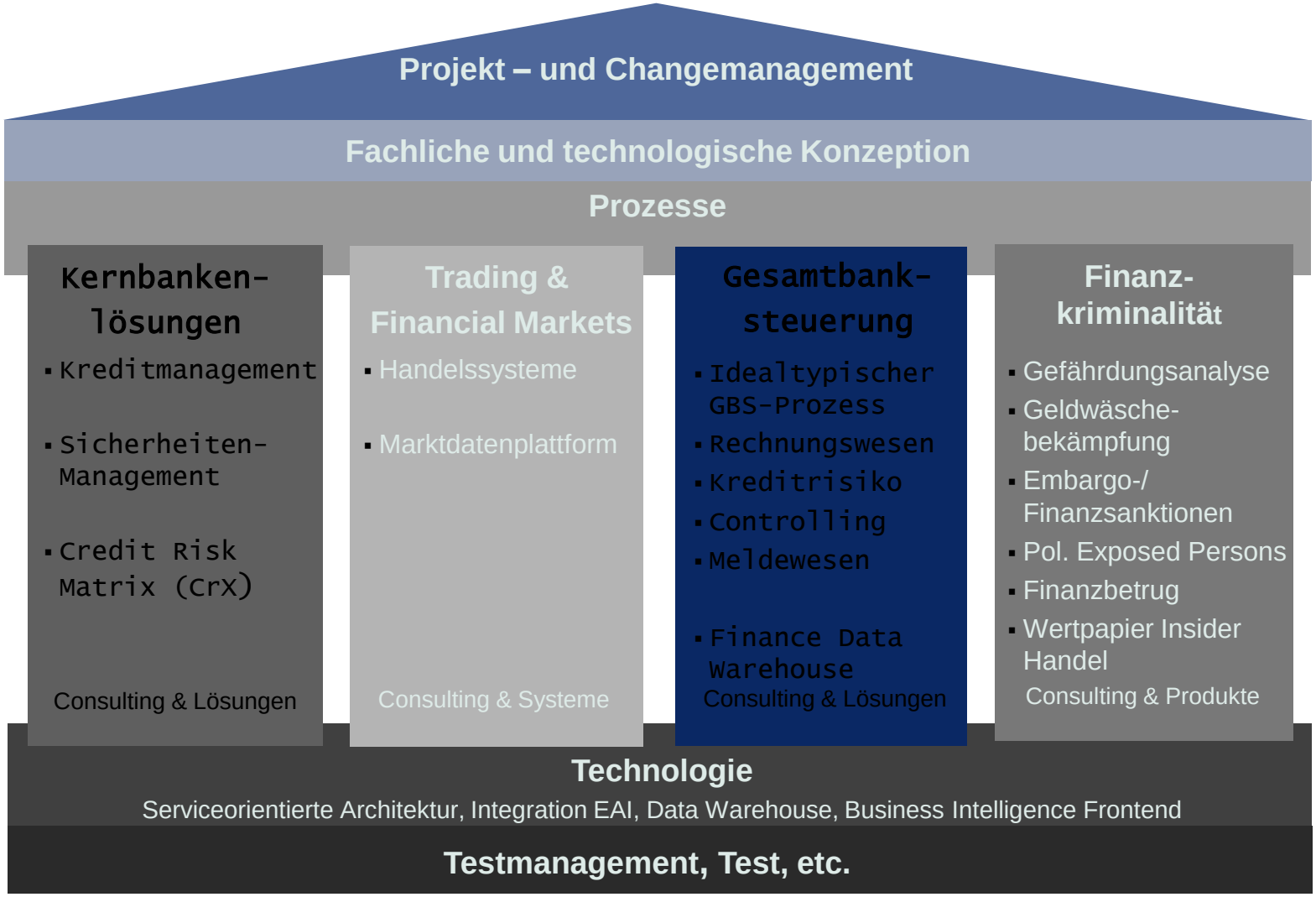
Agenda

- cfs Kurzvorstellung
- Batchbetrieb in einer DB-Applikation
- Möglichkeiten zur Parallelisierung
- SQL Parallelisierung
- Parallel laufende Verarbeitungen
- Parallelisierungsarten: Vergleich Abläufe
- Vergleich Parallelisierung: SQL ⇔ Verarbeitungen
- Kombination der Parallelisierungsmöglichkeiten
- Zusammenfassung

Zahlen und Fakten zum Unternehmen

Firmensitz:	Stuttgart
Marktpräsenz:	seit 1985
Aktionärsstruktur:	100% Landesbank Baden-Württemberg (LBBW)
Aufsichtsrat:	Dr. Martin Setzer
Vorstand:	Thomas Wild
Anzahl Mitarbeiter:	200
Standorte:	Stuttgart, Frankfurt am Main, München
Umsatz:	€ 28 Mio.

Kompetenzfelder



SMARAGD Compliance Suite

- Mehr als 1.600 Institute in über 50 Ländern vertrauen auf SMARAGD Produkte
- Unter den 50 Top Banken weltweit nutzen 9 Banken SMARAGD Lösungen
- Von den Top 10 Banken in Deutschland setzen die ersten 7 auf die SMARAGD Compliance Suite
- Seit 1999 stehen wir mit unseren Lösungen der Finanzwirtschaft und seit 2011 der Industrie zur Verfügung

Referenzen SMARAGD Anti-Finance Crime Suite (Auszug)

Batchbetrieb in einer DB-Applikation

- Vorgaben für Batchjobs
  - begrenztes Zeitfenster für volle Resourcen-Nutzung
  - während Online-Betrieb nur sehr eingeschränkt
    - ➢ Performance für Online-Betrieb darf nicht vermindert werden
    - ➢ deshalb meist nur für umgehende benötigte Ergebnisse eingesetzt
  - Hardware ist immer begrenzt
    - ➢ fast nie genügend schnell für rein serielle Komplett-Verarbeitung
      - in der Regel wirken Schreib/Lese-Zugriffe limitierend
      - bei rechen-intensiven Verarbeitungen auch die CPU
    - ➢ aber:
      - in der Regel mehrere CPU‘s
      - bei größeren Systemen auch mehrere IO-Pfade
      - ➔ das ist nur parallel nutzbar

Möglichkeiten zur Parallelisierung

- Parallelisierung in SQL-Statements
  - komplettes Handling durch Oracle-DB
  - nur mit Enterprise-Edition möglich
  - sinnvoll nur bei großen Datenmengen je paralleles Statement
- gleichzeitig mehrere Batch-Verarbeitungen ausführen
  - applikationsspezifische Steuerung
    - ➢ Berücksichtigung von Abhängigkeiten
    - ➢ nicht nur einzelne Statements, sondern ganze Abläufe sind parallel
  - mit jeder Oracle-Edition möglich
  - parallele Prozesse über dbms_scheduler
    - ➢ Features nach Bedarf einsetzbar
    - ➢ früher eigene Implementierung notwendig
      - – wegen Limitierung dbms_job auf 36 Prozesse

Parallele Ausführung von SQL-Statements

- unterstützt nur von Enterprise Edition
- wird komplett vom Oracle-Server gehandelt
  - bisheriger Session-Prozess plus zusätzliche
  - vom Server bereitgestellte Parallel-Query-Prozesse
  - führen gemeinsam je einen Teil der Aufgaben aus
  - 1 Query-Koordinator für Kommunikation mit anderen Teilnehmern
    - ➢ verteilt Aufgaben, führt Ergebnisse zusammen
  - Session-Verlauf
    - ➢ 1 Session vor Beginn der parallelen Ausführung
    - ➢ Aufsplittung in mehrere Sessions
    - ➢ Zusammenführung in ursprünglicher Session am Ende des Statements
- wenig Aufwand für Entwickler

Parallele Ausführung von SQL-Statements II

- SQL-Parallelisierungsgrad wird vom Optimizer bestimmt
  - bei der Erstellung des Ausführungsplans
  - unter Berücksichtigung von:
    - an den Objecten hinterlegten Parallel-Degree-Werten
    - Parallel-Hints für Haupt- und Sub-Queries
    - Verfügbarkeit von Parallel-Queury-Prozessen
      - DB-Parameter parallel_max_servers
      - sind noch genügend davon frei verfügbar
- Steuerungsmöglichkeit
  - auf Object-Ebene
    - ALTER TABLE mytable PARALLEL 8;
  - auf Statement-Ebene
    - SELECT /*+ parallel(t,16) */ FROM mytable t ...

Parallele Ausführung von SQL-Statements III

- eine DB-Application bei verschiedenen Kunden für:
  - um Größenordnungen unterschiedliche Datenmengen
  - auf verschiedener Hardware
  - erfordert Anpassungsmöglichkeiten:
    - idealerweise auf Statement-Ebene
    - unter Berücksichtigung der parallel laufenden Verarbeitungen
    - über einen oder mehrere Werte in einer Parametertabelle

```
DECLARE
  stmnt VARCHAR2(32767);
  p_degree BINARY_INTEGER;
BEGIN
  SELECT p_value INTO p_degree FROM param_table WHERE p_name = 'DEGREE';
  stmnt := 'CREATE TABLE mytable2 AS' || chr(10) ||
     ' SELECT /*+ parallel(t,' || to_char(p_degree) || ') */' || chr(10) || ' FROM mytable t1';
  EXECUTE IMMEDIATE stmnt;
END;
```

Parallel laufende Verarbeitungen

- komplette Läufe mit Logik zwischen SQL sind parallel
- keine Abhängigkeiten untereinander
  - kein Output der anderen Verarbeitung als Input
  - bei DDL-Methotik wie Exchange Partition
    - ➢ keine Bearbeitung der gleichen Daten
      - z. B. verschiedene Attribute
- kein Komplettservice durch die Datenbank
- notwendige Session entwicklerseitig bereitstellen
  - seit Oracle 10 mit dbms_scheduler, bis zu 999 Prozesse
  - vorher mit dbms_job, bis zu 36 Prozesse
    - ➢ meist nicht ausreichend
    - ➢ ➔ Eigenimlementierung mit separaten Connects von DB-Server
      - via Aufruf einer C-Library

Parallel laufende Verarbeitungen II

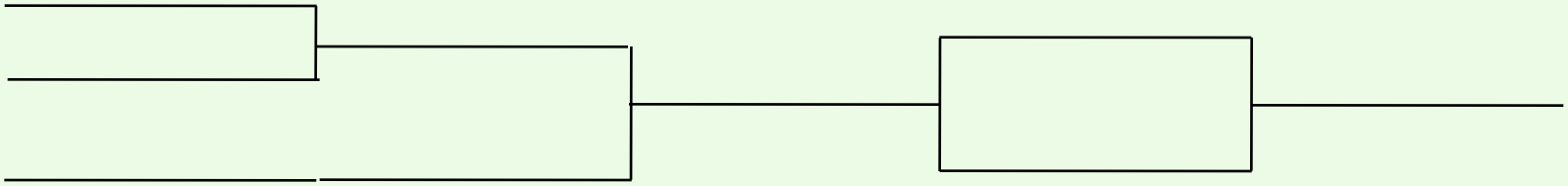
- Kommunikation zwischen
  - ➢ aufrufenden Sessions
  - ➢ ausführenden Sessions
  - muss entwicklerseitig implementiert werden
- Statustabelle als einfachste Möglichkeit
  - mögliche Werte: angefordert, aktiv, abgeschlossen, abgebrochen, ..
  - ggf. mit Oracle-Fehlernummern und Oracle-Fehlermeldungen
- sichere Erkennung laufender Verarbeitungen
  - a) Steuerung läuft als hochprivilegierter User mit v$session-Zugriff
    - ➢ Hinterlegung von sid, spid, logon-date etc. in Statustabelle
    - ➢ dbms_application_info: setzen von module und action
  - b) Verwendung von Locks mit berechneten Namen
    - ➢ Zugriff auf dbms_lock Package erforderlich
    - ➢ Duration = 'SESSION' ➔ Existenz Lock bedingt Existenz Session

Parallel laufende Verarbeitungen III

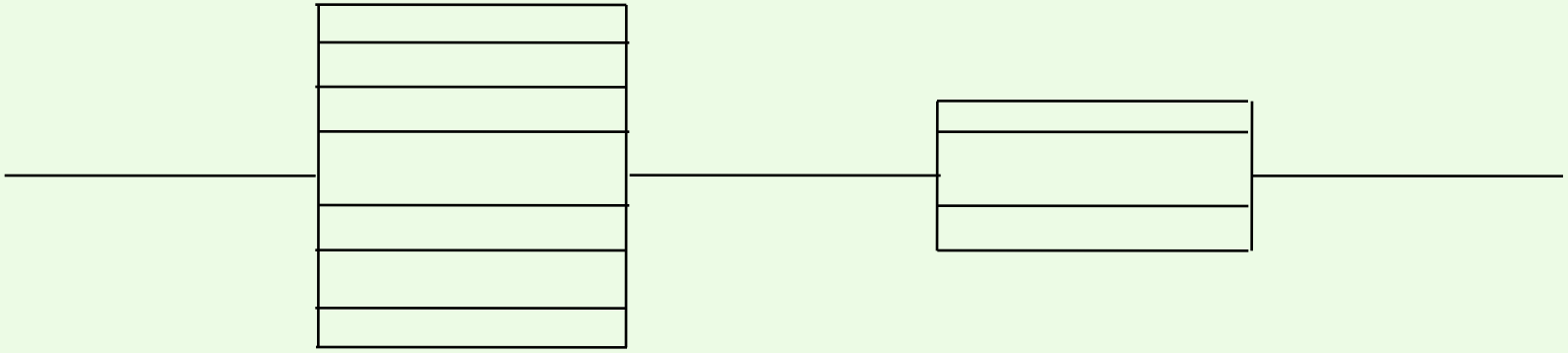
- Parallelisierung unabhängig von Verarbeitungsmethodik
  - also auch Einzelsatz-Verarbeitungen mit Cursor-Loops
    - ➢ die vom parallelen SQL nicht profitieren
  - auch hierbei bessere Hardware-Auslastung möglich
    - ➢ falls genügend parallel ausführbare Verarbeitungen existieren
      - – abhängig vom Design der Verarbeitungen
  - Verarbeitungen bei notwendigen Serialisierungspunkten unterteilen
    - ➢ alle Serialisierungspunkte im Steuerungsbereich verfügbar
    - ➢ UND-verknüpfte Abhängigkeiten für Start
    - ➢ optimale Nutzung der Parallelisierungsmöglichkeiten
  - ansonsten:
    - ➢ alle benötigten Daten bis zum letzen Serialisierungspunkt vor Start
    - ➢ alternativ Ablauf-Steuerungslogik in Verarbeitungen einbauen

Parallelisierungsarten: Vergleich Abläufe

- Parallelisierung / Abfolge von Verarbeitungen
  - separate Verarbeitungsjobs zwischen Knoten
  - jeder Zusammenführungsknoten ist ein Serialisierungspunkt



- SQL-Parallelisierungen in einer Verarbeitung
  - automatische Zusammenführung am Ende eines SQL-Statements



Vergleich Parallelisierung: SQL ⇔ Verarbeitungen

	SQL Statements	komplette Verarbeitungen
Verfügbarkeit	Enterprise Edition	generell
Design-Aufwand	keiner	geeignete Aufteilung in separate Verarbeitungen
Entwicklungs-Aufwand	minimal	hoch
Parallelisierungsgrad	SQL-Hint, Object Parallel Degree	Anzahl Parallele Sessions
sichere Detektierung laufender Parallel-Session	wird von Oracle-DB durchgeführt	muss implementiert werden

Kombination beider Parallelisierungsmöglichkeiten

- System mit großen Daten-Mengen und -Durchsatz
- vielen verschiedenen Verarbeitungs-Schritten
- komplexen Zusammenhängen der Verarbeitungen
- ➔Parallele Verarbeitungsläufe sinnvoll
  - wahrscheinlich sogar notwendig wegen Batch-Zeitfenster
- ➔SQL-Parallelisierung sinnvoll
  - innnerhalb von Verarbeitungsläufen
  - bei SQL-Statements mit großem Datendurchsatz
  - verkürzt die Laufzeit des betreffenden Verarbeitungslaufes
  - ermöglicht früheres Anstarten abhängiger Verarbeitungen
- SQL-Parallelisierung und Anzahl paralleler Verarbeitungen
  - müssen aufeinander abgestimmt sein
  - damit die Hardware ausgelastet, aber nicht überlastet wird

Kombination beider Parallelisierungsmöglichkeiten II

- Abstimmung Parallelisierung SQL und Verarbeitungen
  - abhängig von:
  - Struktur der gesamten Abfolge paralleler Verarbeitungen
  - und deren Parallelisierungs-Konfiguration
    - ➢ zur Anpassung an die eingesetzte Hardware
  - SQL-Parallelisierung muss dazu passen
    - ➢ damit die Hardware ausgelastet, aber nicht überlastet wird
  - sollte sicherstellen:
    - ➢ dass für nachfolgende abhängige Parallele Verarbeitungen
    - ➢ und andere Parallele SQL-Statements
    - ➢ hinreichend Resourcen (Parallel-Prozesse) verfügbar sind
  - das ist möglich, weil:
  - bei Design/Entwicklung/Ausführung abhängiger Verarbeitungen
    - ➢ in die Zukunft geplant wird
    - ➢ und damit bekannt ist, welche Resourcen nachfolgend benötigt werden

Kombination beider Parallelisierungsmöglichkeiten III

- Query-Optimizer kennt nur die aktuelle Situation der DB
  - kann prinzipiell nichts über nachfolgend benötigte Resourcen wissen
  - weil es keine Möglichkeit der Mitteilung gibt
- Informationsvorsprung kann benutzt werden
  - für optimalere Hardwareauslastung
  - durch parametergesteuerte SQL-Parallelisierung
    - ➢ mit geeigneter Anzahl Parallelisierungs-Parameter
    - ➢ Vewendung von Faktoren auf diese Parameter
      - – für bekannten geänderten Resourcenbedarf in Folgestatements
    - ➢ ggf. zur Laufzeit berechnete Faktoren
      - – zum Handling unvorhersehbarer Laständerungen
        - – z. B. verspätete Daten-Lieferung von anderem System
        - – oder zusätzliche Anwender-verursachte Verarbeitungen
  - sollte nur für Beinflussung der SQL-Parallelisierung verwendet werden
    - ➢ nicht für andere dynamische SQL-Hints (nicht Optimizer nachbauen)

Zusammenfassung

- System mit großen Datenmengen, aber nur einfachen Batchverarbeitungen ohne spezielle Abhängigkeiten
  - ➢SQL-Parallelisierung ausreichend
- System mit großen Datenmengen und komplexeren Batchverarbeitungen, auch mit Abhängigkeiten
  - ➢Kombination von Verarbeitungs-Parallelisierung mit SQL-Parallelisierung empfehlenswert
- System mit kleineren Datenmengen und komplexeren Batchverarbeitungen, auch mit Abhängigkeiten
  - ➢Verarbeitungs-Parallelisierung empfehlenswert
  - ➢ + SQL-Parallelisierung, wenn Enterprise Edition und wenn Performance-Verbesserung erwartet wird

Fragen & Antworten

Vielen Dank für Ihre Aufmerksamkeit

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