Partition Exchange of dependent Entities - Consistency in error case

Dr. Kurt Franke

Cellent Finance Solutions GmbH

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

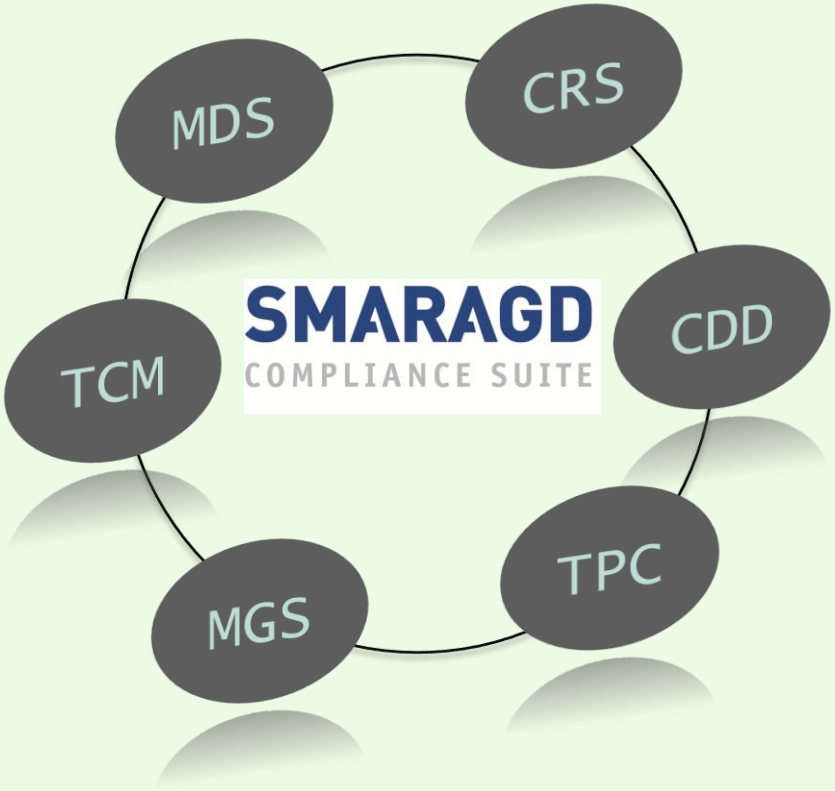
Agenda

- brief description of cfs
- motivation to use partition exchange for data integration
- data integration into targettable by partition exchange
- simple case: just one entity
- complex data models
- occurence of a new type of problem
- a possible problem handling
- handling of session-breaks by fatal error
  - (ORA-00600 etc.)
- additional benefit
  - of method for consistency restoration
- conclusion

Company facts & figures

Headquarter:	Stuttgart
Branches:	Frankfurt/Main, Munich
Industry experience:	since 1988
Turnover:	€ 30 Millions
Employees:	210
Legal form:	Limited liability company
CEO:	Thomas Wild
Parent company:	100 % Landesbank Baden-Württemberg (LBBW)
Deputy chairman:	Dr. Martin Setzer

SMARAGD Compliance Suite



- More than 1.600 corporates and financial institutes in over 50 countries rely on the SMARAGD Compliance suite
- Nine out of the 50 top worldwide banks use SMARAGD
- The first seven of the top 10 German banks rely on SMARAGD
- The German and the Austrian Central Banks use SMARAGD
- Since 1999 Cellent Finance Solutions provides software solutions for the financial market

Customer overview SMARAGD (Extract)

Motivation to use Partition Exchange for Data Integration

- high data volume in delivery(i. e. in a full sync)
  - multi-stage preparation using work tables
    - ➢ utlitize massdata SQL-Statements
      - – high performance
- high number of changes after re-calculation of risk
  - when risk model has changed
- target tables are:
    - ➢ historized (per entity characteristic)
      - – new data rows always at end (no history changes)
      - – new rows has open validity range (direct load insert)
      - – predecessor rows have to be closed (update)
    - ➢ partitioned (list) by independent units
    - ➢ subpartitioned (range) by end value of validity range
  - ➔ primary key disabling not possible during processing
    - ➢ sometimes runtime duration up to 12 hours for last integration steps

Data Integration into Targettable by Partition Exchange I

- preparation of exchange table
  - inserting of prepared new rows from delivery
    - these are rows with current validity
    - using INSERT /*+ APPEND */
  - inserting rows for closing the predecessor rows for historizing
    - using select from target table actual partition
    - including the setting of close date
      - i. e. a mikrosecond before validity start of the new rows
    - using INSERT /*+ APPEND */
  - inserting all other rows which must be preserved in actual partition
    - using INSERT /*+ APPEND */
  - creating:
    - all indexes (including PK) and check constraints like on target table
    - all outgoing and incoming FK constraints like on target table
  - disable of all incoming and outgoing FK constraints
    - on exchange table and target table

Data Integration into Targettable by Partition Exchange II

- do exchange subpartition for current unit
  - INCLUDING INDEXES WITHOUT VALIDATION
    - ➢ important for performance
  - avoid global partitioned or non-partitioned indexes
    - ➢ otherwise time consuming for update or recreation
- enable the disabled FK constraints
  - ideally after exchange partition operations for all entities
- precondition for this approach
  - adequate temporally partition boundaries
    - ➢ during this processing
    - ➢ for cause the closed rows belonging to the actual partition
  - where required adapt partitioning utility
    - ➢ split partition for previous period not before 2nd day of actual period
    - ➢ ➔ exchange of actual partition is sufficient
      - – changed data exists only for actual partition

just a simle Case: only one Entity

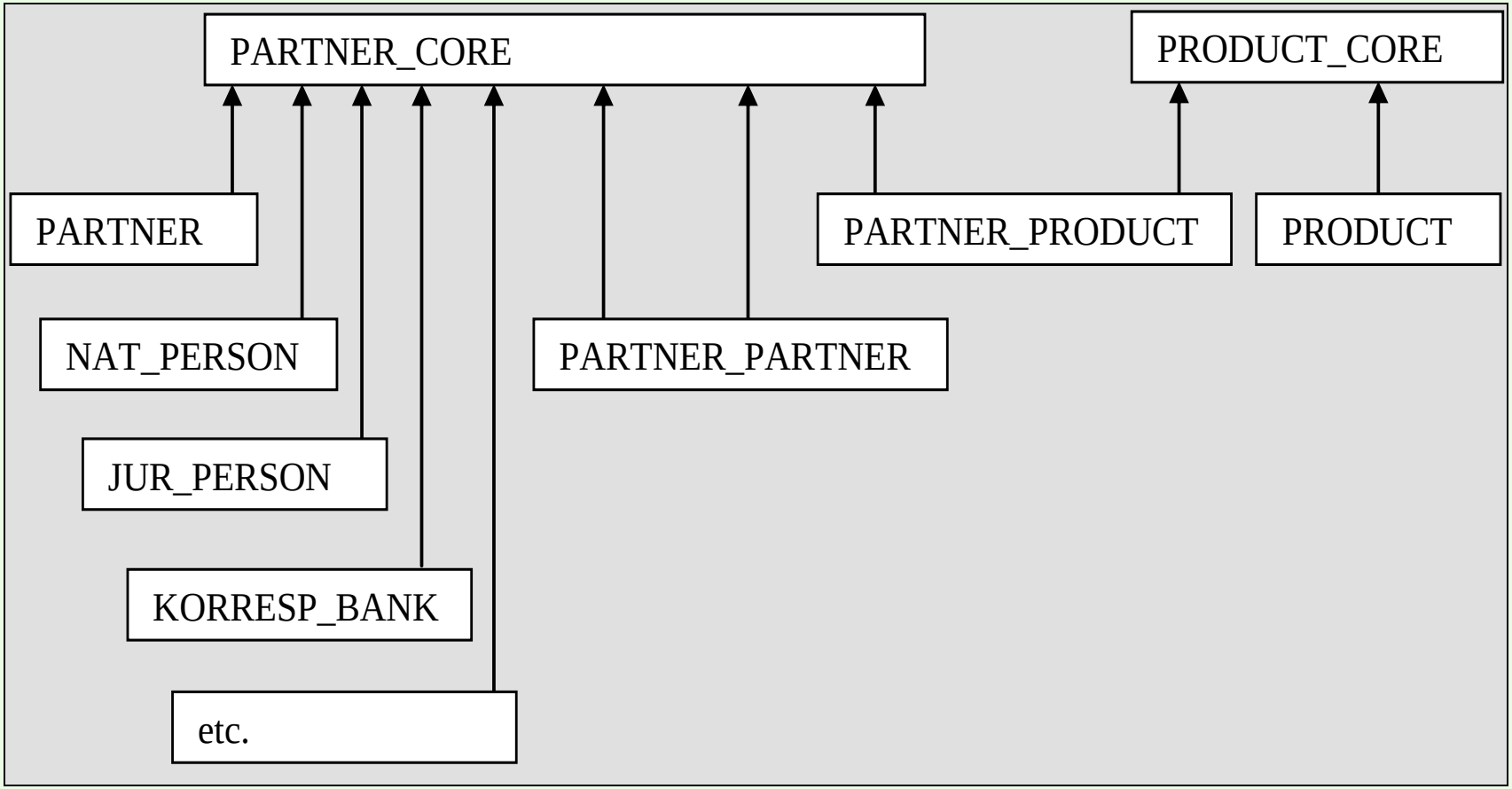
- multiple independent entities
  - always separate handling in separate processing possible
- handling of one entity
  - partition exchange in „AUTONOMOUS_TRANSACTION“

```
PROCEDURE part_exchange(…) AS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN … /* Exchange Code */ END;
```

  - ➔ outside DML is protected from DDL COMMIT
  - a possible error during partition exchange
    - ➔ no partition exchange is done
    - exception raised
    - DML is rolled back
    - state is unchanged like at start of action
  - error state in standard transaction handling resolved

Complex Data Models

- extract from a more complex data model
  - ✓ the data of the entities have to fit together



Scenario: exchange error with dependent entities

- for each single entity
  - partition exchange in „AUTONOMOUS_TRANSACTION“

```
PROCEDURE part_exchange(…) AS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN … /* Exchange Code */ END;
```

  - ➔ outside DML is protected from DDL COMMIT
  - possbible error during partition exchange 4
    - ➔ no partition exchange for entity 4 is done
    - exception raised
    - DML is rolled back
    - **but:** partition exchanges for entities 1 - 3 was succesful
    - they could not rolled back because done using DDL
    - could not restore state like at start of action
      - unless implementing a special handling for this
- error situation !!!
  - could not be resolved using standard transaction handling

Occurence of a new type of problem

- dependent entities: data state for same time ?
  - Y: data consistency of the entity set
  - N: no data consistency of the entity set
- partition exchanges of some dependent entities – entity set
  - error during 2nd or later partition exchange
  - ➔ no data state for same time over all entities
  - data of entities in its entirety – the entity set – are inconsistent
  - no resolving using standard transaction handling is possible
  - ➔ need to implement a special handling
    - ➢ for resolving this error situation

possible Solution of the new problem

- problem caused by use of partition exchange DDL
  - ➔ reasonably should be a solution using partition exchanges
  - by reverse unwanted exchange operations
  - and avoid reverse exchange
    - for entities without exchange operation until the error occured
- requirements for this solution
  - exchange tables with the exchanged data must still exist
    - ➔ cleanup is forbidden until succesful completion of all exchanges
  - need a methode for recognition of already exchanged partitions
    - with less data examination as possible
    - to get sufficient performance

reverse a Partition exchange Operation

- exchange operation:
  - low level swap of the data objects behind
    - a target table partition
    - a special exchange table with identically structure
  - second apply onto the both objects
    - restores the original state
      - without a special reverse operation
      - like a 2-fold symmetry operation
- utility for exchange operation + additions
    - code for serialization of multiple calls for same target table
    - logging
    - but not any data changes (DML)
  - behaves in same way as bare exchange operation
  - ➔ it is usable for both data integration exchange and reversing it
    - no special reversing utility needed

Detection of already exchanged Partitions

- before a partition exchange:
  - Exchange table will be filled completly new
    - after all changes in the target table
  - ➔ all exchange table rows are newer than all target table rows
  - ➔ SCN (exchange table) **>** SCN (target table partition)
    - for arbitrary rows of both exchange table and target table partition
- after a succesful partition exchange:
  - the both objects are exchanged
    - exchange table became target table partition
    - and vice versa
  - ➔ SCN (target table partition) **>** SCN (exchange table)
    - for arbitrary rows of both exchange table and target table partition
- after an error during partition exchange:
  - all is identically like before the partition exchange

Detection of already exchanged Partitions II

- **pseudo-code:**

```
FOR ALL (target_table_part / exchange_table) pairs
LOOP
 SELECT ora_rowscn FROM exch_tab
   INTO exch_tab_scn
   WHERE rownum = 1;
 SELECT ora_rowscn FROM target_tab_part
   INTO target_tab_scn
   WHERE rownum = 1;
 IF target_tab_scn > exch_tab_scn
 THEN
  util_exchange_part(target_tab_part, exch_tab);
 END IF;
END LOOP;
```

Restore original State (before Processing started state)

- when a partition exchange error occures
  - stop regular processing
  - raise a special exception
    - to jump into restore mode
  - LOOP over all involved entities
    - reverse partition exchange for all entities with succesful exchange
    - INCLUDING INDEXES WITHOUT VALIDATION
    - same procedure as done during exchange for data integration
  - if exists, disable and re-enable FK constraints around the exchange
    - same procedure as done during exchange for data integration
  - cleanup (set state, release locks, …)
  - completion of processing with error state
- problem is completely resolved
  - continuation with other standard processing possible

Session-Break by Fatal Error (ORA-00600 etc.)

- fatal error while restoring original state
  - processing session does not exist any longer
  - state of the processed unit is inconsistent
- the following have to take effect:
  - further data changes of any kind have to be prohibited
    - because of the data inconsistency
    - for the previous processed unit
  - the only allowed action is a call to restore consistency
    - for the previous processed unit
  - a database application update have to be prohibited
    - due to possible changes in the table structure
      - of the processed tables
    - otherwise a partition exchange potentially becomes impossible
      - and thus an automatic restore of consistency

Session-Break by Fatal Error II

- use a consistency marker per unit (values: Y/N, NOT NULL)
  - set to N + COMMIT before starting with partition exchanges
  - this will be preserved after a session break
  - set to Y + COMMIT:
    - after succesful completion of all involved partition exchanges
    - after succesful restoration of the state before processing started
      - just after a partition exchange error occured
    - after succesful restoration of the state before processing started
      - in a separate call
- value of N for a unit
  - update of the DB application is prohibited
  - any type of processing for this unit is prohibited
    - through integration of the consistency marker into lock handling
  - excluding: the call to restore consistency

additional Benefit

- proceeding for consistency restoration
  - option for reset to before last processing
    - ➢ also after succesful completion of processing
    - ➢ same precondition as for consistency restoration
  - only a suitable steering is required
- proceeding for reset a processing
  - much faster as processing
    - ➢ only partition exchange portion of processing utility is used
      - – the original partitions will be integrated again
  - without enhancement merely a reset to before last processing
  - none partitioned data are ruled out

additional Benefit II

- enhanced proceeding for reset processings
  - methode for allow reset of multiple processings
    - suppress partition split completely
      - eligible for a test mode only
    - postpone partition split at n days to future
      - 1 day postponement is an assumption for the proceeding
      - 7 days postponement allow 7 days switch back
        - split previous period first at 8th day of actual period
        - always at least 7 days in partition for actual period
        - should be reasonably configurable
    - save exchange tables directly after processing
      - shield through rename from normal processing
      - partition exchange utility: exchange table as a parameter
    - implement switch back of unpartitioned data
      - using DELETE and UPDATE for backtracking to target time
    - control table for backtracking timestamps
      - (unit, numero, timestamps, type of processing)

additional Benefit III

- enhanced proceeding for reset processings
  - enables direct reset of multiple processings
    - ➢ without backtracking step by step
      - – in special when processing all tables (type of processing)
    - ➢ when only a subset of tables is handled in reset point processing
      - – first an intermediate reset is necessary
        - – to before the next following complete processing
        - – ➔ correct contents for the other tables also
        - – which did not handled in reset point processing
      - – if necessary switch back in stages
      - – till desired reset point is reached
- also if multiple stages needed
  - reset processings is independend from number of processings
  - ➔ runtime nearly independend from number of processings
  - flashback ☺ is a reasonable term for this proceeding

Comparison: one entity ⇔ dependend entities

	one entity	dependend entities
partition exchange	AUTONOMOUS TRANSACTION	AUTONOMOUS TRANSACTION
outside DML protected from DDL COMMIT	yes	yes
partition exchanges after exchange error	not any	previous exchanges before current
after ROLLBACK	state before begin of handling restored	intermediate state around begin
consistency after exchange error	YES	**NO**

Conclusion

- fast data integration into historized target tables
  - for some entities with inter-dependencies
- exchange partition with new filled exchange tables
  - in an Autonomous Transaction for commit-protecting outside DML
- error in exchange operation after one successful done
  - reverse exchange partition for all successful exchanges
  - determined by any greater SCN in target table partition
    - than in exchange table
  - when session break occures during this
    - consistency flag stored before beginning of exchanges remains N
    - this will prevent another changes
    - ➔ prohibit garbage creation by usual processing
      - because of assuming consistency
    - unless consistency is restored

Q & A

Thank You !

Contact: **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