OPITZ CONSULTING

Menschen. Innovationen. Lösungen.

Wenn das Projekt eben doch kein Standard-Fall ist …

Bessere Datawarehouses durch Table Functions

Gero Knapstein
OPITZ CONSULTING Gummersbach GmbH

Gummersbach, 6.11.2011

Wenn das Projekt eben doch kein Standard-Fall ist …

Massendatenverarbeitung?

Reine SQL-Statements sind da immer das Beste …

Wirklich immer?

Nein

Ein kleiner Teil der Anforderungen weigert sich hartnäckig gegen eine performante, wartbare SQL-Implementierung …

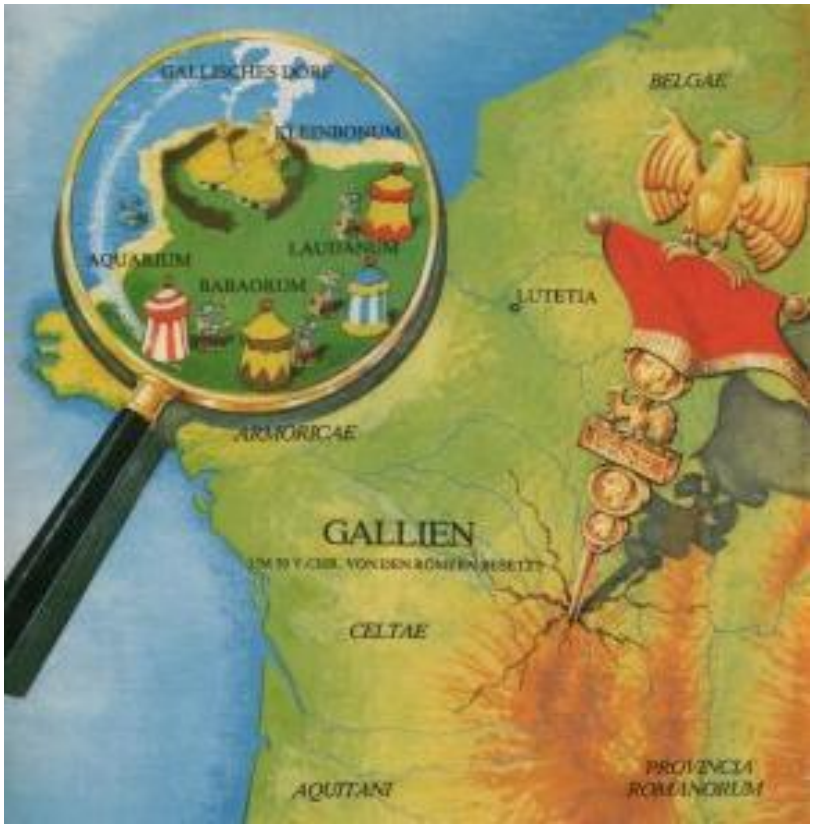


Table Functions in Datawarehouses Agenda

1. Wie erstelle ich eine Table-Function?

1. Table Function allgemein
2. Non-Pipelined versus Pipelined Table Functions
3. Pipelined Table Function im Datenstrom
4. Parallelisierung / Partitionierung

2. Einsatzbeispiele in Datawarehouses

1. Near-Realtime-Datawarehouse mit PLSQL
2. Kontrolle / Manipulation historisierter Stages
3. Zeitdimension

3. Einsatzvorschläge in Datawarehouses

1. Qualitätsprüfung auf Package-Collections und –Cursor
2. SCRUM-Sprints / Agile Entwicklung von ETL-Strecken

4. Fazit

Ihr ORACLE Center of Excellence

Leistungs-angebot

- Java
- SOA/BPM
- ORACLE
- BI/DWH
- Outtasking
- Exadata
- Exalogic

ORACLE APPROVED EDUCATION CENTER

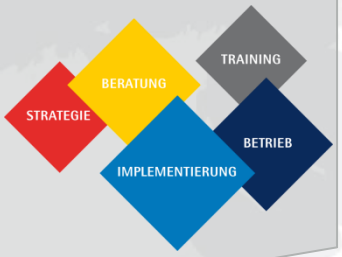
Kunden / Kollegen

- Über 600 Kunden
- Branchen-übergreifend

- Über 450 Kollegen
- An 8 Standorten

Tätigkeits-felder

- IT-Strategie
- Beratung
- Implementierung
- Betrieb
- Training



ORACLE Platinum Partner
Specialized
Oracle Database

ORACLE Platinum Partner
Specialized
Service-Oriented Architecture

ORACLE Platinum Partner
Specialized
Oracle Enterprise Linux

ORACLE Platinum Partner
Specialized
Oracle Real Application Clusters

ORACLE Platinum Partner
Specialized
Oracle Business Intelligence Foundation

1.1 Table Function allgemein

Table Function allgemein

- **Returns Collection**
  - Nested Table
  - Varray
  - Anydataset

- **Rückgabe als Ergebnismenge in SQL verwenden:**

```
select * from TABLE (my_package.my_table_function);
```

Table Function Vorbereitung

- **Return-Collectiontyp muss im SQL-Kontext bekannt sein**
  - Mit DDL Object Type und Collection Type anlegen

```
create or replace type my_date_objtype
is object (jahr number, tag number, datum date);

create or replace type my_date_coltype
is table of my_date_objtype;
```

  - In Package-Spezfikation Record-Type und Collection definieren
    - werden implizit als Object-Types angelegt, wenn als Table-Function genutzt
    - Name: „SYS_PLSQL_64703_141_1“

Table Functions

Arten

- **Zwei Arten mit unterschiedlicher Wirkung**
  - Pipelined Table Function
  - (nonpipelined) Table Function
  - in Zukunft (12.x): Bulk Pipelined Table Function

1.2 Non-Pipelined versus Pipelined Table Functions

(nonpipelined) Table Function

```
create or replace function get_date_of_year
(p_year in number) return my_date_coltype is
  v_retcol my_date_coltype := my_date_coltype();
  v_date date v_date := to_year(p_year);  i number := 1;
begin
  loop
    exit when v_date >= to_year(p_year+1);
    v_retcol.extend;
    v_retcol(i) := my_date_objtype(p_year, i, v_date);
    v_date := v_date + 1;  i := i + 1;
  end loop;
  return v_retcol;
end;
```

- **Erst wird komplette Collection aufbereitet**
  Dann wird Collection zurückgegeben
- **Konsequenzen**
  - Kompletter Collection-Speicherbedarf muss allokiert werden
  - Wenn Rückgabe nach SQL-Kontext, dann nur 1 Kontextswitch (!)

Pipelined Table Function

```
create or replace function get_date_of_year_pipelined
(p_year in number) return my_date_coltype pipelined is
  v_date date := to_year(p_year);
  i number := 1;
begin
  loop
    exit when v_date >= to_year(p_year+1);
    pipe row (my_date_objtype(p_year, i, v_date));
    v_date := v_date + 1;  i := i + 1;
  end loop;
  return;
end;
```

- **Erst wird KEINE Collection aufgebaut**
  Separate Rückgabe potentieller Collectionzeilen
- **Konsequenzen**
  - KEIN Collection-Speicherbedarf
  - Für jede Zeile nach SQL 1 Kontextswitch (!)

Table Functions — Vergleich

- **Non-Pipelined Table Function**
  - Sinnvoll bei kleinen Collections
    - Speicherbedarf proportional zu Collectiongröße
    - Performante Bereitstellung als SQL-Datenquelle
- **Pipelined Table Function**
  - Sinnvoll bei großen Collections
  - Performance-Einbuße (Kontextswitches)
- **Ausstehend (Oracle 12.x?)**
  Bulk Pipelined Table Function

1.3 Pipelined Table Function im Datenstrom

Pipelined Table Function mit Cursor-Parameter

- **IN-Parameter vom Typ REF CURSOR**

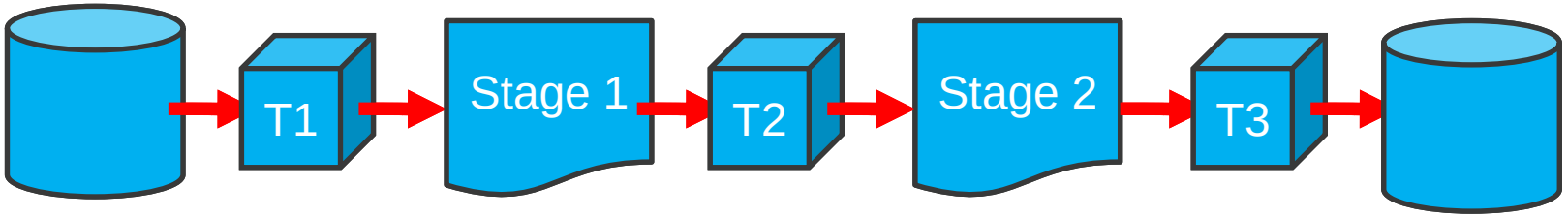
```
create or replace function get_date_of_year
(p_refcurs in t_ref_cursor) return my_date_coltype pipelined
Is …
```

- **Aufruf der Funktion mit**

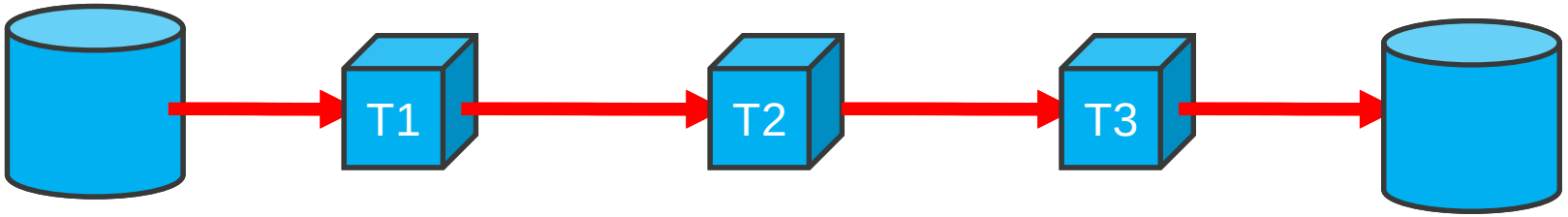
```
Select * from table(get_date_of_year (CURSOR (select * from …));
```

- **Funktion CURSOR ()**
  wandelt Query in einen geöffneten REF CURSOR um

Pipelined Table Function im Datenstrom



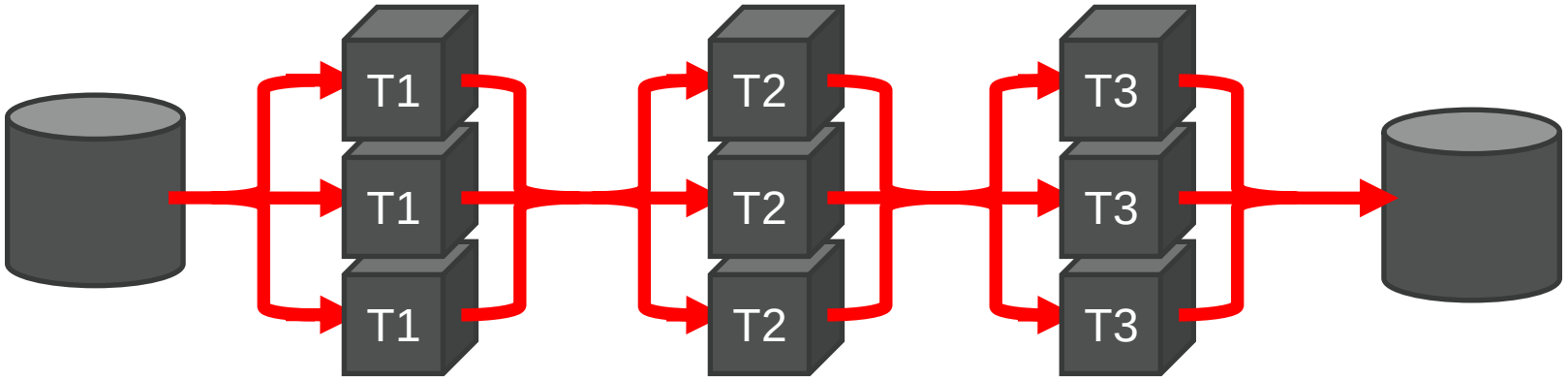
- **T2 wartet mit Start, bis Stage1 komplett von T1 befüllt wurde**
  - Schnell zwischen Stages, wenn T1, T2 und T3 reine SQL-Statements sind



- **Table Function T2 nimmt Datenstrom aus Table Function T1 unmittelbar auf**
  - Optimaler „First-Rows“-Mechanismus
  - Gebremst durch Kontextswitches zwischen SQL- und PLSQL-Kontext
- **Beschleunigung durch Parallelisierung / Partitionierung**

1.4 Parallelisierung / Partitionierung

Pipelined Table Function parallelisiert




```
create or replace function get_date_of_year_pipelined
(p_refcurs in t_ref_cursor) return my_date_coltype pipelined
PARALLEL_ENABLE (PARTITION refcurs BY HASH (artikel_nr)
Is …
```

Parallel_enable

Parallel_enable (partition <curs> by any)

Parallel_enable (partition <curs> by range <columnlist>)

Parallel_enable (partition <curs> by hash <columnlist>)

2 Einsatzbeispiele in Datawarehouses

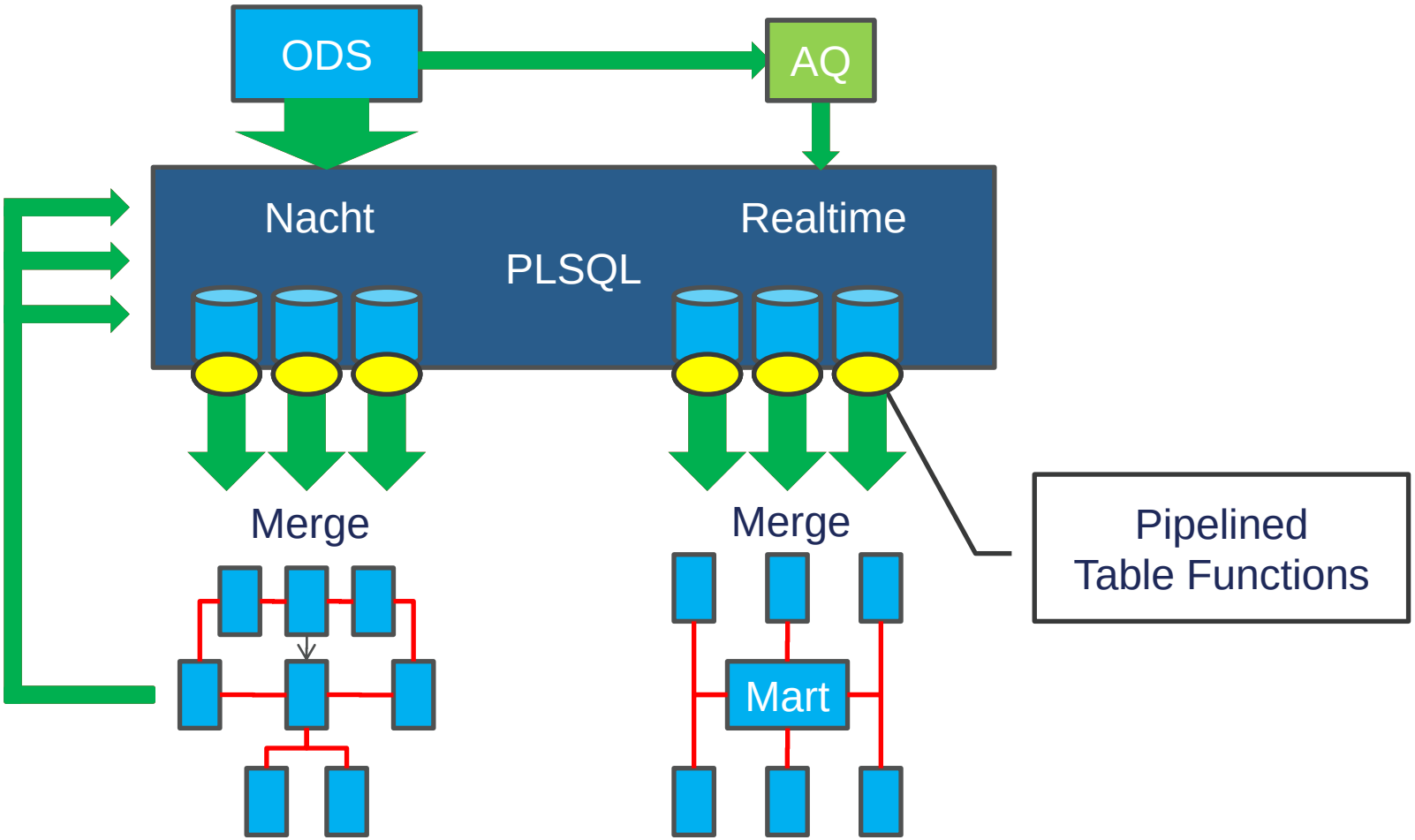
2.1 Near-Realtime Datawarehouse

Near-Realtime-DWH Motivation

- Eine breite, denormalisierte Quelltabelle (ODS)
- Komplexe Aufbereitung
- Kunde wollte/will PL/SQL (will nicht OWB)
- Near-Realtime-Verarbeitung < 10 Min.

Near-Realtime-DWH

Architektur



2.2 1:1-Übernahme der Historisierung ?

Übernahme Historisierung — Motivation

- **1:1-Übernahme der Historisierung**
  - Für DWH-Anforderungen unbrauchbar (ERP hatte geringe Ansprüche)
  - Nachbesserungen im ERP waren nicht ausreichend.
  - Fehleridentifikation mit einfachem Select notwendig
  - -> selbstgeschriebene Analytische Funktion
- **Modifikation in Stunden-genaue Historisierung**
  - Fehlerbereinigung
    - Lückenlose, nicht überlappende Terminierung
    - Korrekte Statusabfolge
  - Reduzierung auf ältesten Satz in Stunde
- **Metadaten-gesteuerte Historisierungsaufbereitung**
  - Expressions für Modifikation (Stunden-genau, Minuten-genau, …)
  - Historisierung (SCD1 oder SCD2)?
  - Welcher Status wird durchgelassen?

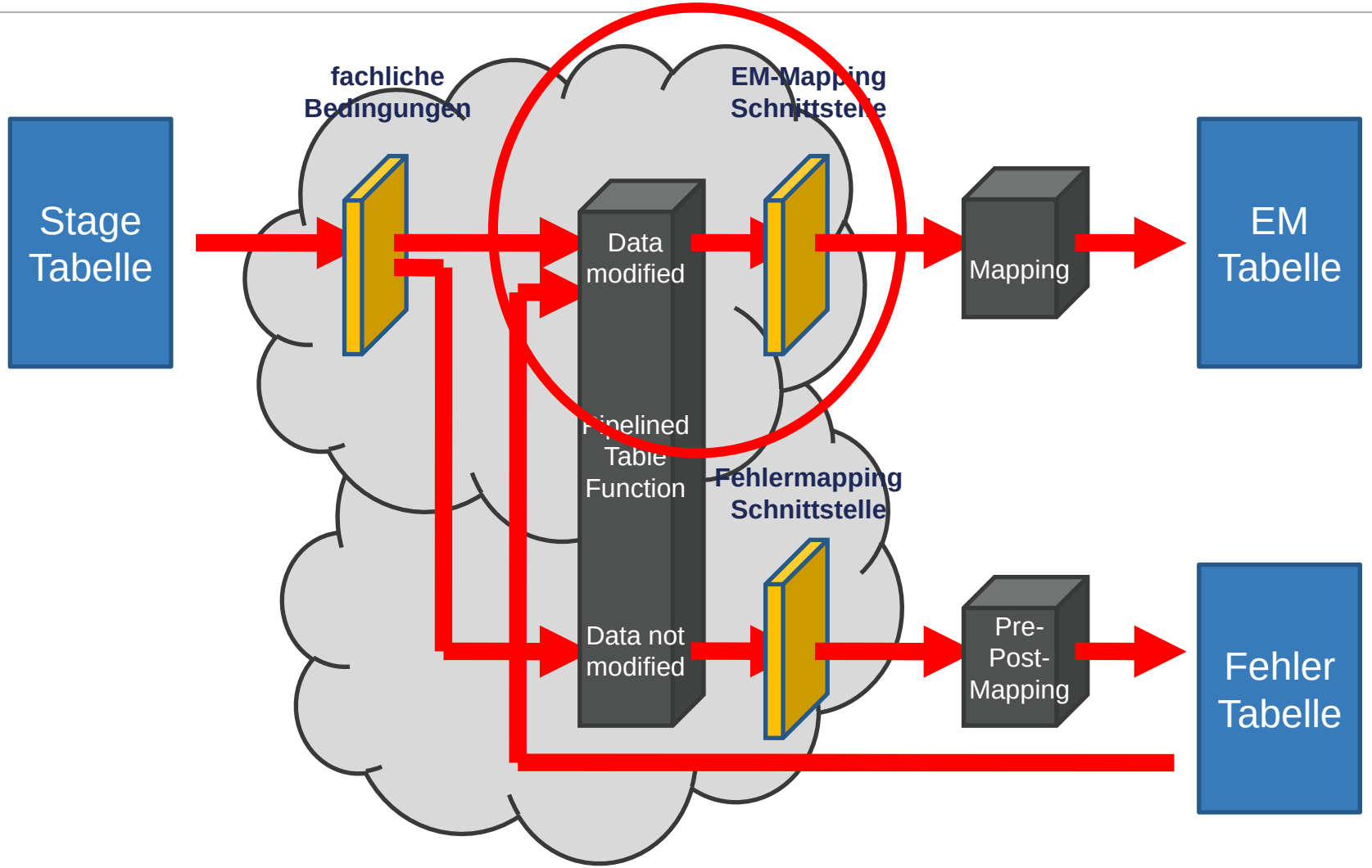
Übernahme Historisierung Lösungsansatz

- **Metadaten-gesteuerte Generatoren**
- **Table Function Generator**
- **View Generator**

- **Views und ggf. Table Functions werden mit Pre-Mapping aktualisiert**

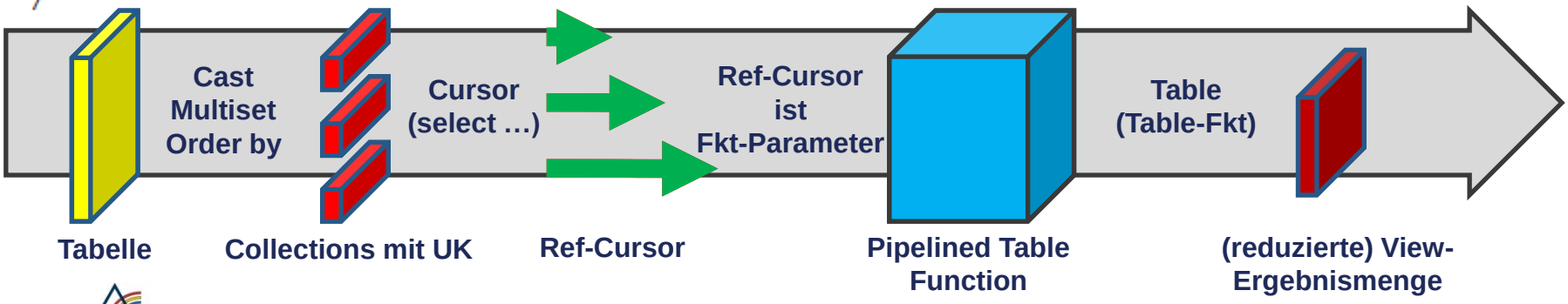
- **Pipelined Table Function**
  - Übernimmt Sätze in sortierten Collections pro Schlüssel
  - Korrigiert / eliminiert Sätze (separate Fehlerprotokollierung möglich)
  - Schreibt korrigierte Sätze in indizierte Collection -> jüngster Satz bleibt übrig

Übernahme Historisierung Lösungsansatz



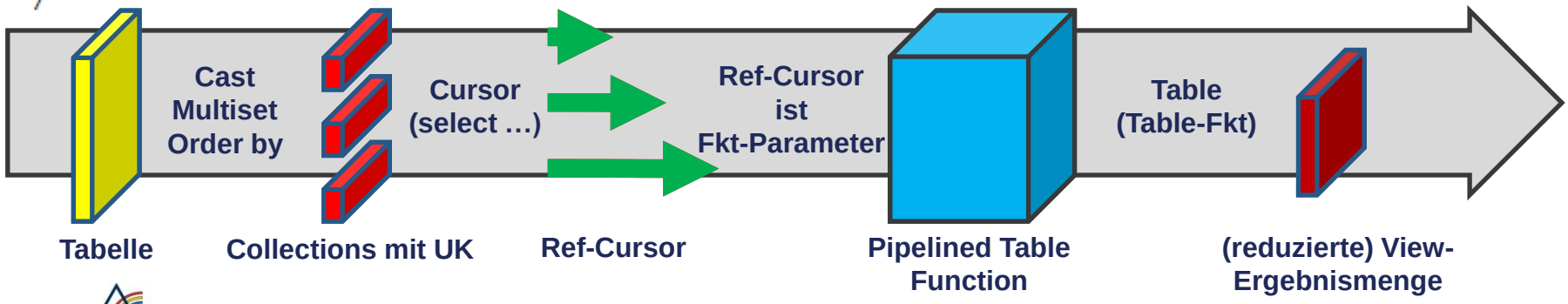
Übernahme Historisierung EM-Mapping-View

```
CREATE VIEW SE_ARTIKEL_BASIS_V
AS
   SELECT *
    FROM TABLE (SE_ARTIKEL_BASIS_P.SE_ARTIKEL_BASIS
       (CURSOR (with my_source AS (SELECT ID,... FROM SE_ARTIKEL_BASIS_B WHERE <uk-cols not null>
                                   UNION ALL SELECT ID,... FROM FE_ARTIKEL_BASIS   WHERE <uk-cols not null>
                                  )
SELECT MAX(ART_NR) AS ART_NR ,MAX(MANDANT) AS MANDANT ,MAX(WG) AS WG
     , CAST
       ( MULTISET
         ( SELECT SE_ARTIKEL_BASIS_T(ID,...)
             FROM my_source t2
            WHERE t1.ART_NR=t2.ART_NR AND t1.MANDANT=t2.MANDANT AND t1.WG=t2.WG
            ORDER BY t2.gueltig_von, t2.gueltig_bis
         ) AS SE_ARTIKEL_BASIS_N
       ) AS my_collection
  FROM my_source t1
 GROUP BY ART_NR,MANDANT,WG
), p_set_trunc_history_on => 1))
/
```

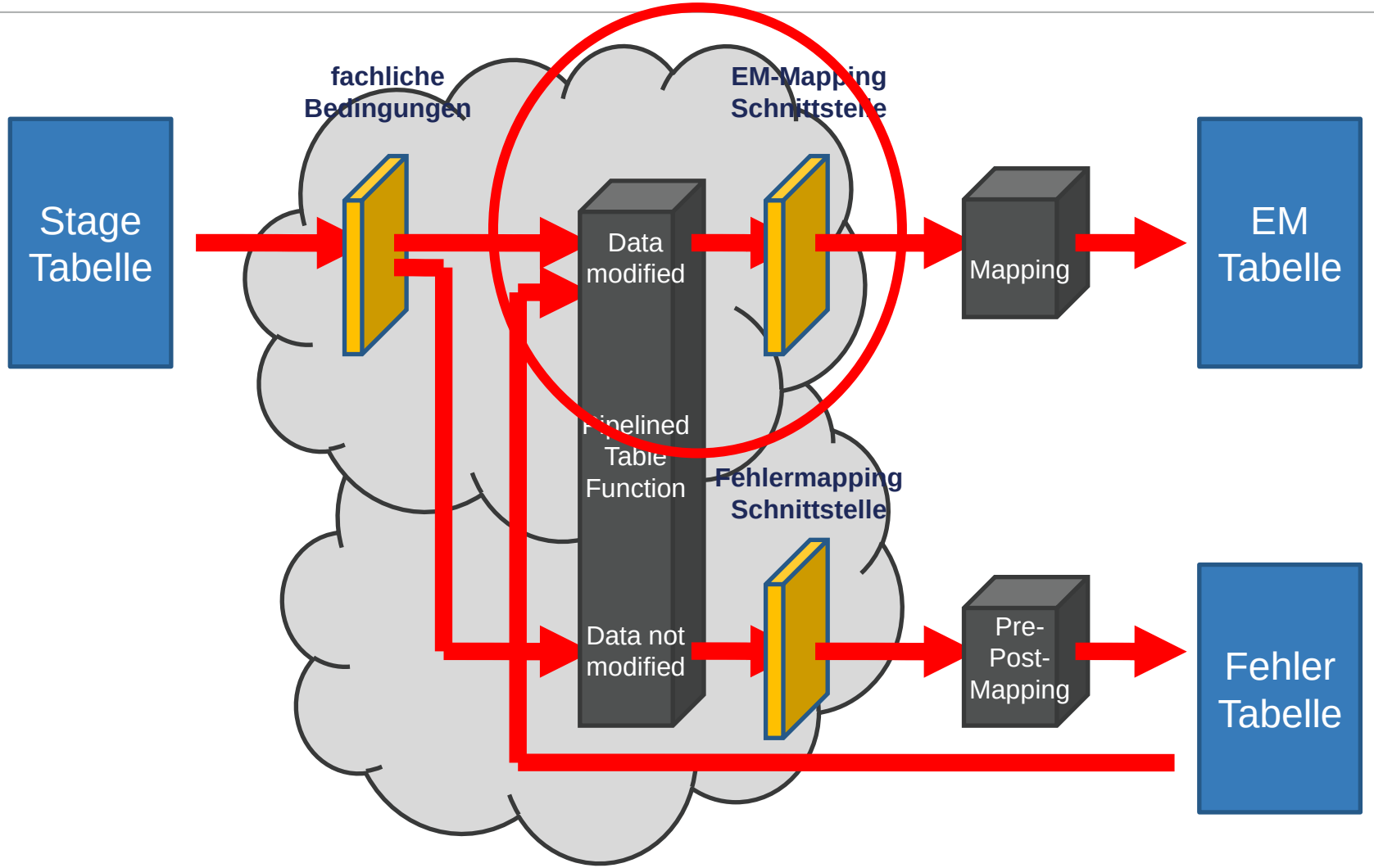


Übernahme Historisierung EM-Mapping-View

```
CREATE VIEW SE_ARTIKEL_BASIS_V
AS
   SELECT *
    FROM TABLE (SE_ARTIKEL_BASIS_P.SE_ARTIKEL_BASIS
       (CURSOR (with my_source AS (SELECT ID,... FROM SE_ARTIKEL_BASIS_B WHERE <uk-cols not null>
                                   UNION ALL SELECT ID,... FROM FE_ARTIKEL_BASIS   WHERE <uk-cols not null>
                                  )
SELECT MAX(ART_NR) AS ART_NR ,MAX(MANDANT) AS MANDANT ,MAX(WG) AS WG
     , CAST
       ( MULTISET
         ( SELECT SE_ARTIKEL_BASIS_T(ID,...)
             FROM my_source t2
            WHERE t1.ART_NR=t2.ART_NR AND t1.MANDANT=t2.MANDANT AND t1.WG=t2.WG
            ORDER BY t2.gueltig_von, t2.gueltig_bis
         ) AS SE_ARTIKEL_BASIS_N
       ) AS my_collection
  FROM my_source t1
 GROUP BY ART_NR,MANDANT,WG
), p_set_trunc_history_on => 1))
/
```



Übernahme Historisierung Lösungsansatz



Übernahme Historisierung

Table Function

```
CREATE OR REPLACE package body SE_ARTIKEL_BASIS_P is
  subtype                 t_stunde is varchar2(100);
  k_trunc_mask            constant varchar2(30) := 'YYYYMMDDHH24';
  v_limit                 number := 1000;
  v_global_trunc_history  boolean := false;
procedure set_trunc_history_on  is begin v_global_trunc_history := true;  end;
procedure set_trunc_history_off is begin v_global_trunc_history := false; end;
function SE_ARTIKEL_BASIS (c in t_ref_cursor, p_set_trunc_history_on in number default null) return SE_ARTIKEL_BASIS_N pipelined is
  type t_in_rec is record
  (ART_NR NUMBER(6,0),MANDANT NUMBER(2,0),WG NUMBER(4,0)
  , my_collection SE_ARTIKEL_BASIS_N);
  type t_in_col is table of t_in_rec; col t_in_col;
  type a_ret_t is table of SE_ARTIKEL_BASIS_T index by t_stunde; a_ret a_ret_t;
  empty_rec SE_ARTIKEL_BASIS_T;
  v_stunde  t_stunde;
  v_local_trunc_history boolean := false;
begin
  if p_set_trunc_history_on is not null then
    v_local_trunc_history := case when p_set_trunc_history_on=1 then true else false end;
  else
    v_local_trunc_history := v_global_trunc_history;
  end if;
  --
  loop
    fetch c bulk collect into col limit v_limit;
    exit when col.count = 0;
    for i in nvl(col.first,1) .. nvl(col.last,0) loop
      a_ret.delete;
      for j in nvl(col(i).my_collection.first,1) .. nvl(col(i).my_collection.last,0) loop
        v_stunde := to_char(col(i).my_collection(j).GUELTIG_VON, k_trunc_mask);
        a_ret(v_stunde) := col(i).my_collection(j);
      end loop; -- j.
      --
      if a_ret.count > 0 then
        v_stunde := a_ret.first;
        loop
          if v_local_trunc_history then
            a_ret(v_stunde).gueltig_von := to_date(to_char(a_ret(v_stunde).gueltig_von, k_trunc_mask), k_trunc_mask);
            if a_ret.exists(a_ret.next(v_stunde)) then
              a_ret(v_stunde).gueltig_bis := to_date(to_char(a_ret(a_ret.next(v_stunde)).gueltig_von, k_trunc_mask), k_trunc_mask);
            else
              a_ret(v_stunde).gueltig_bis := to_date(to_char(a_ret(v_stunde).gueltig_bis, k_trunc_mask), k_trunc_mask);
            end if;
          end if;
          pipe row (a_ret(v_stunde));
          exit when v_stunde = a_ret.last;
          v_stunde := a_ret.next(v_stunde);
        end loop; -- k.
      end if;
      --
    end loop; -- i.
  end loop;
  if c%isopen then close c; end if;
  return;
end SE_ARTIKEL_BASIS;
```

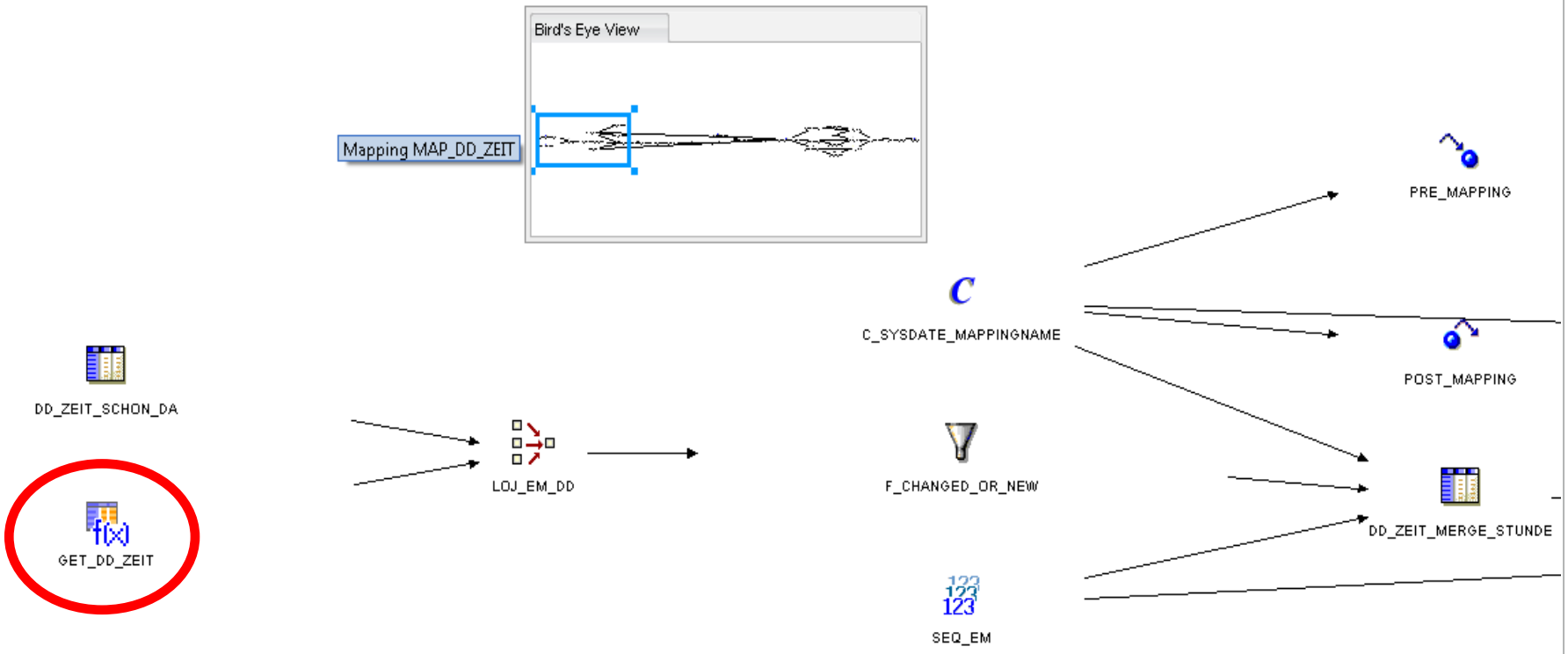
Komplette Logik

2.3 Aufbau Zeitdimension

Table-Function für Zeitreihen — Motivation

- ETL-Strecken werden durch OWB / Workflow betrieben
- OWB-Mapping ist Dokumentation
- Zeitreihen-Interval soll frei wählbar sein
- Zeitreihen-Grenzen sollen frei wählbar sein
- Prozedurale Geschäftslogiken sollen performant eingebunden werden

Table-Function für Zeitreihe OWB-Mapping



- **Table Function lässt sich in OWB-Mapping einbinden wie Table / View o.ä.**

3 Weitere Einsatzvorschläge

3.1 Qualitätsprüfung auf Package-Collections und -Cursor

Table-Functions für Package-QS Motivation

- **ETL-Backends haben Packages für**
  - „Hauskeeping“
  - Geschäftslogiken
  - Analysen, Mailing, …
- **Package-Collections**
  - Sind die PLSQL „Arrays“
  - Werden benötigt für performante, komplexe Programmlogik
  - Sind schwierig zu debuggen
- **Unit-Tests werden gerne durchgeführt mit**
  - Select-Statements
  - Utplsql o.ä.

Table-Functions für Package-QS leicht gemacht!

```
create or replace package body my_package is
...
function query_my_collection return t_my_collection pipelined is
begin
  for i in nvl(a_my_collection.first,1) .. nvl(a_my_collection.last,0) loop
    pipe row (a_my_collection(i));
  end loop;
  return;
end;
...
function query_my_cursor return t_my_cursor_rec_col pipelined is
begin
  for rec in my_cursor loop
    pipe row (rec);
  end loop;
  return;
end;
...
end;
```

- Select * from table(my_package.query_my_collection);
- Select * from table(my_package.query_my_cursor);

3.2 Agile DWH-Entwicklung Sprints mit Table Functions

Entwicklungs-Sprints mit Table Functions

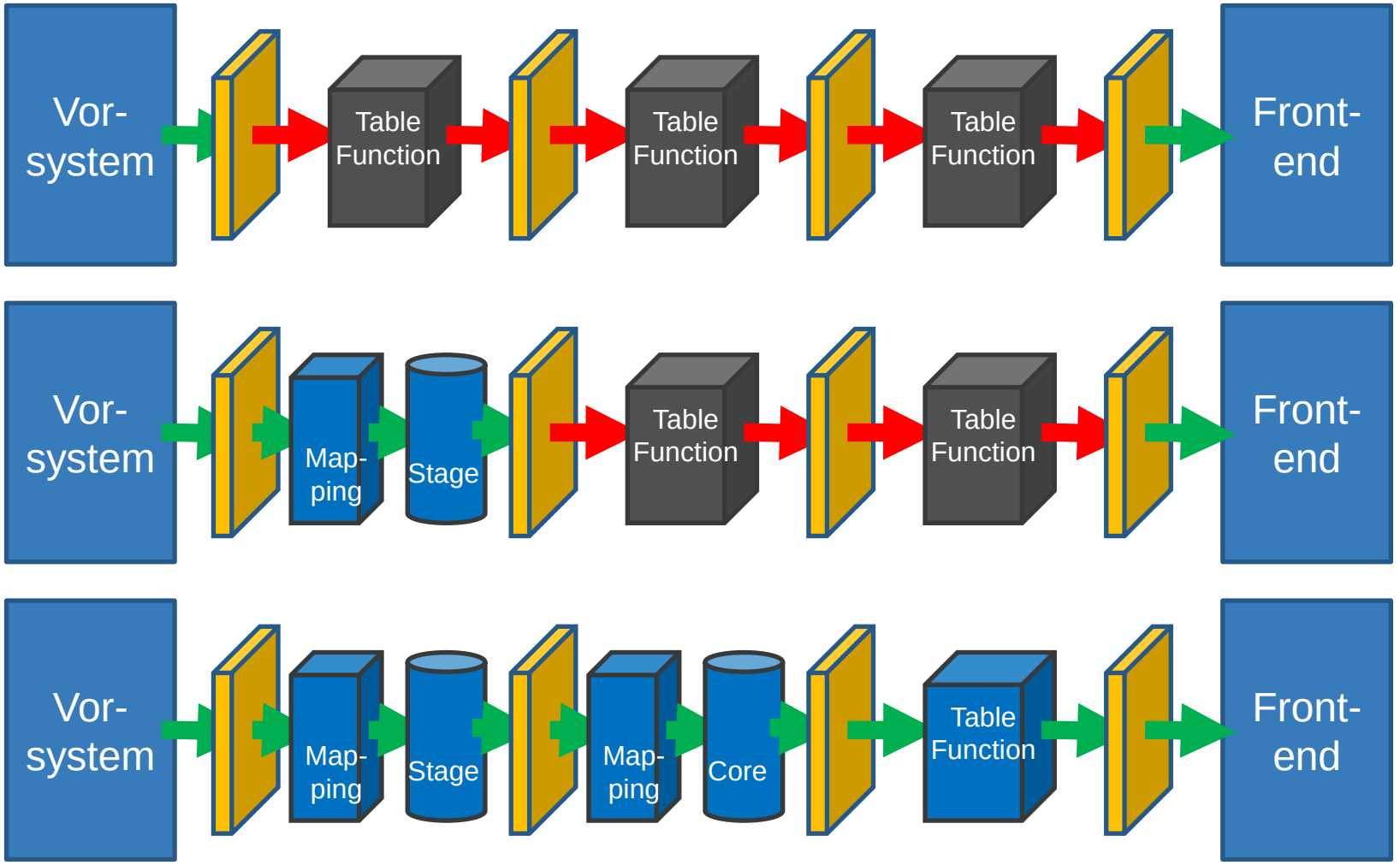
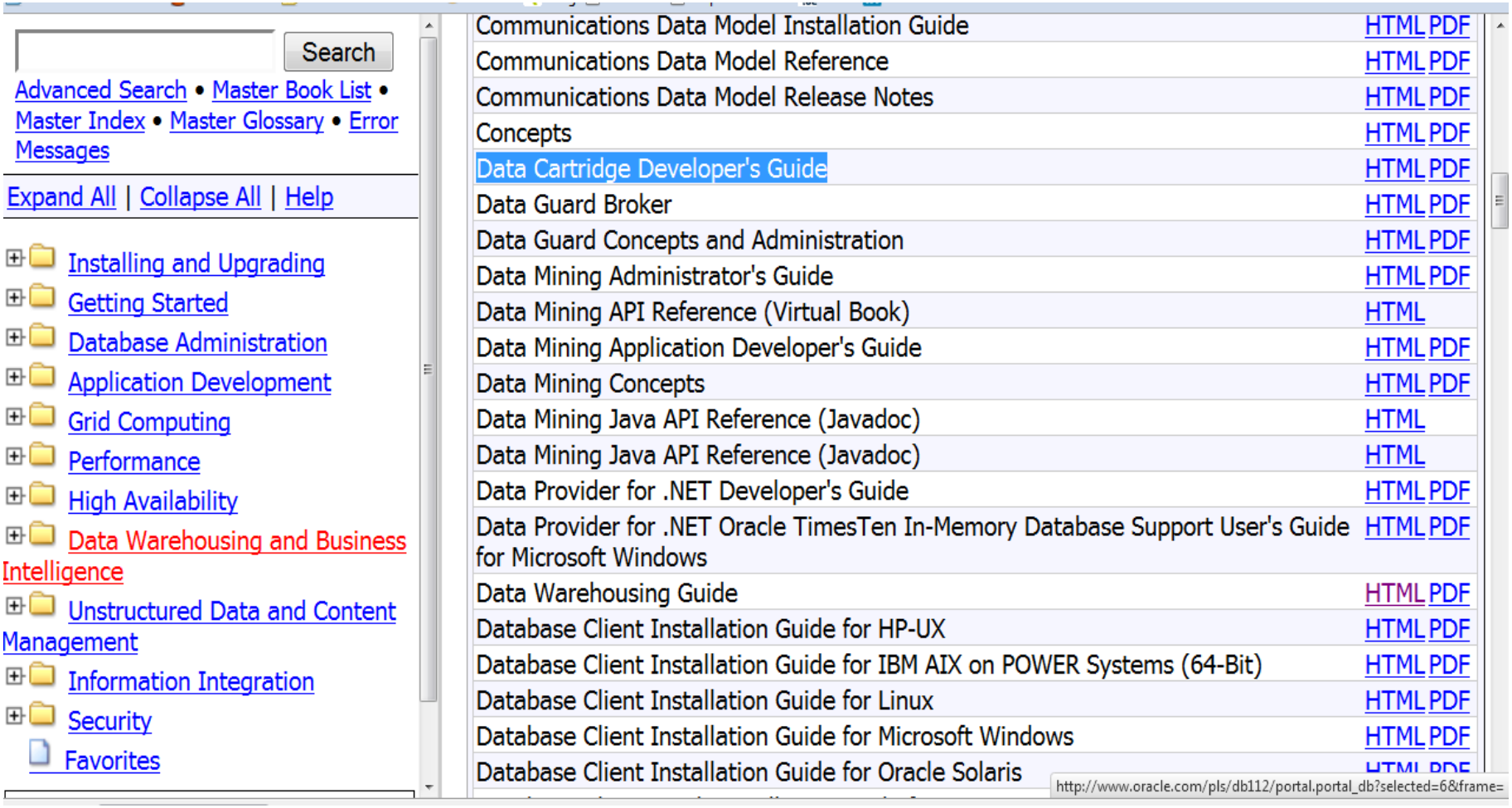


Table Function Online-Dokumentation



Fazit

- **Table Functions muss man nicht unbedingt haben, aber sie machen komplexe Logiken wartbarer und manchmal auch schneller**
- **Table Function basierte Ergebnismengen sind langsamer als reine SQL-Statements, aber sie skalieren und bewältigen DWH-übliche Datenmengen**
- **Transformationsstrecken aus Table Function basierten Views eignen sich für agile DWH-Entwicklung mit schnellen Sprints**
- **Ausblick (Oracle 12.x): Bulk-Verarbeitung wird den Durchsatz deutlich beschleunigen!**

Fragen & Antworten

Ihr Ansprechpartner

Gero Knapstein
Senior Berater

OPITZ CONSULTING GmbH
Kirchstr. 6 | 51647 Gummersbach
Tel. +49 (2261) 60 01-0
gero.knapstein@opitz-consulting.com

youtube.com/opitzconsulting

@OC_WIRE

slideshare.net/opitzconsulting

xing.com/group-51062.460375

Besuchen Sie auch die anderen Vorträge von OPITZ CONSULTING und unseren Stand (Nr. 236)!

Dienstag, 15. November 2011	Mittwoch, 16. November 2011	Donnerstag, 17. November 2011
MySQL in an Oracle driven datacenter 10:00 bis 10:45 Uhr, Raum Singapur	**Das ungleiche Paar – Koexistenz von OWB und ODI** 09:00 bis 09:45 Uhr, Raum Kopenhagen	**Grails – Die Suche ist vorbei** 09:00 bis 09:45 Uhr, Raum Riga
Oracle Forms meets BI 10:00 bis 10:45 Uhr, Raum Kiew	**Praxis Knowhow: Skalierung von SOA Suite 11g Cluster** 09:00 bis 09:45 Uhr, Raum Budapest	**Enterprise Architecture Deliverables – Let‘s talk about results!** 09:00 bis 09:45 Uhr, Raum Prag
Minimale Latenz – Bedarfsgerechte Bereitstellung von Daten im DWH 10:00 bis 10:45 Uhr, Raum Kopenhagen	**RAC ONE Node 11.2.0.2. – Wo ist meine Instanz?** 13:00 bis 13:45 Uhr, Raum St. Petersburg	**Brückentechnologie – Min. Downtime Plattform-Migration / Upgrade von 9 nach 11.2** 10:00 bis 10:45 Uhr, Raum St. Petersburg
ESSBASE und die OBIEE 11g – Aufbruch zu „echten“ OLAP-Analysen 12:00 bis 12:45 Uhr, Raum Helsinki	**Oracle BAM – Die unentdeckten Möglichkeiten** 13:00 bis 13:45 Uhr, Raum Oslo	**Versteckte Schätze in BPM&SOA Suite 11g – gesammelte Projekterfahrungen** 10:00 bis 10:45 Uhr, Raum Oslo
SOA Continuous Integration 12:00 bis 12:45 Uhr, Raum Riga	**Neues zur Oracle Lizenzierung** 15:00 bis 15:45 Uhr, Raum Kopenhagen	**Unterbrechungsfreies Reporting: Hochverfügbarkeit von OWB bis BIEE 11g** 12:00 bis 12:45 Uhr, Raum Stockholm
Agile BI mit OBIEE 11g 14:00 bis 14:45 Uhr, Raum Helsinki		**Oracle Resource Management** 13:00 bis 13:45 Uhr, Raum St. Petersburg
Die Crux mit dem Delta – vom Fullload zum Incremental Load 16:00 bis 16:45 Uhr, Raum Kopenhagen		**Forms Legacy – ein ADF Panorama** 14:00 bis 14:45 Uhr, Konferenzraum EG
Automatisiertes Konfigurationsmanagement mit Puppet 16:00 bis 16:45 Uhr, Konferenzraum EG		**Deseaster Recovery bei Grid Infrastructure 11.2 mit zwei Rechenzentren** 15:00 bis 15:45 Uhr, Raum Hongkong
		Effizientere ETL mit Table Function 16:00 bis 16:45 Uhr, Raum Stockholm

youtube.com/opitzconsulting

slideshare.net/opitzconsulting

xing.com/group-51062.460375

@OC_WIRE