metafinanz

Partitionierung im Data Warehouse mit ORACLE 11g und 12c

Reinhard Wahl

DOAG Konferenz –
Nürnberg, 18.-20. November 2014

Wir fokussieren mit unseren Services die Herausforderungen des Marktes und verbinden Mensch und IT.

Business Intelligence

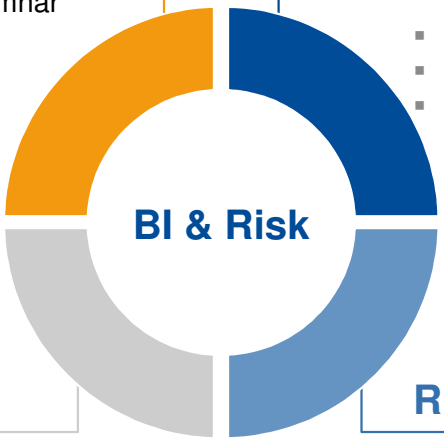
Themenbereiche

Enterprise DWH

- Data Modeling & Integration & ETL
- Architecture: DWH & Data Marts
- Hadoop & Columnar DBs
- Data Quality & Data Masking

Analytics

- Predictive Models, Data Mining & Statistics
- Social Media Analytics
- Customer Intelligence
- Scorecarding
- Fraud & AML



Insurance Reporting

- Standard & Adhoc Reporting
- Dashboarding
- BI Office Integration
- Mobile BI & In-memory
- SAS Trainings for Business Analysts

Risk

- Solvency II (Standard & internal Model)
- Regulatory Reporting
- Compliance
- Risk Management

Über metafinanz

metafinanz gehört seit fast 25 Jahren zu den erfahrensten **Software- und Beratungshäusern** mit Fokus auf die Versicherungsbranche.
Mit einem Jahresumsatz von 270 Mio. EUR und über 450 Mitarbeitern entwickeln wir für unsere Kunden **intelligente zukunftsorientierte Lösungen** für komplexe Herausforderungen

Ihr Kontakt: Reinhard Wahl

Senior Consultant

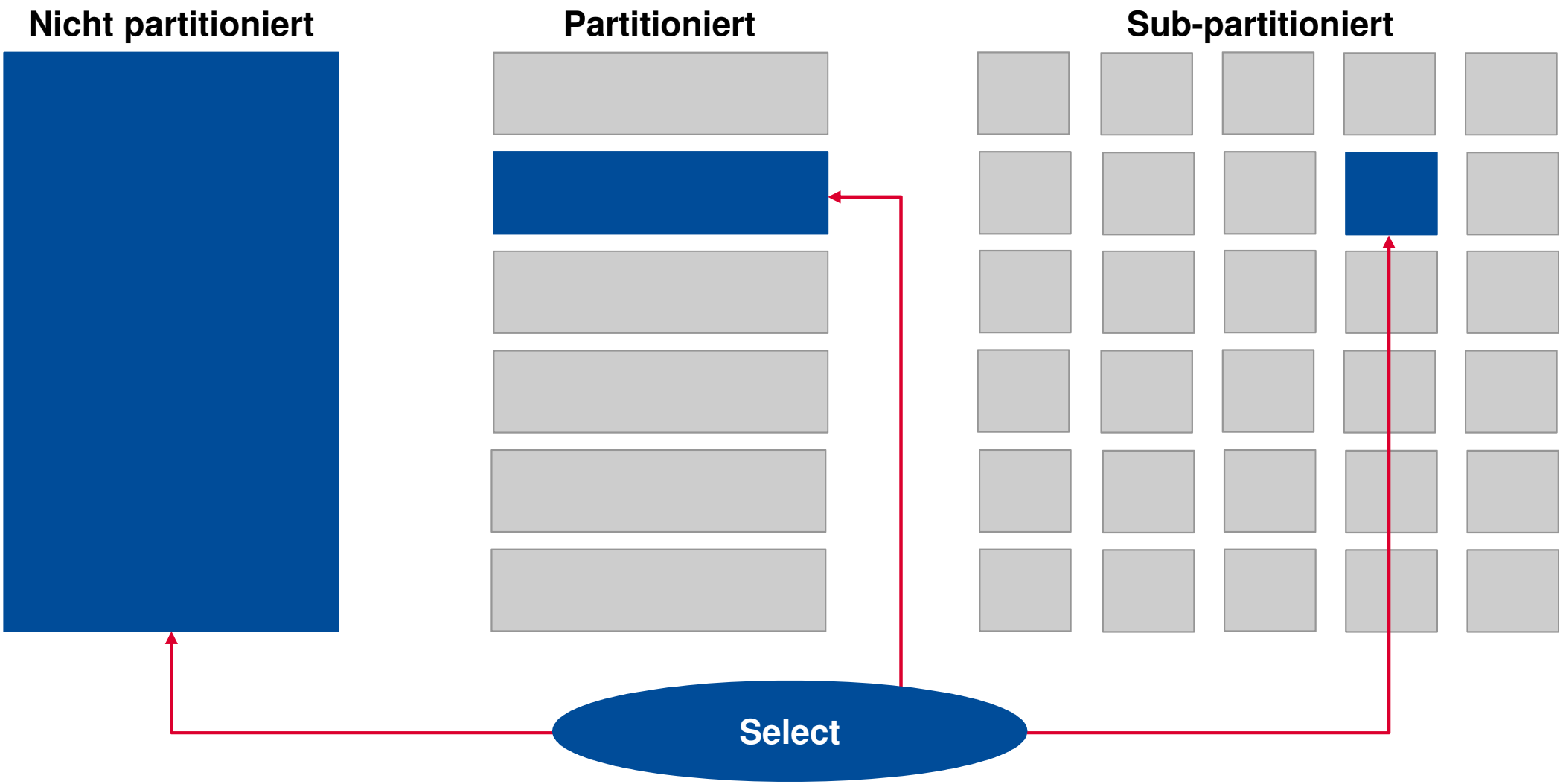
- Diplom Elektrotechnik
- Mehr als 20 Jahre Oracle-Erfahrung
- Überwiegend PL/SQL, SQL, DWH, OWB

Mail: reinhard.wahl@metafinanz.de
Phone: +49 89 360531 5082

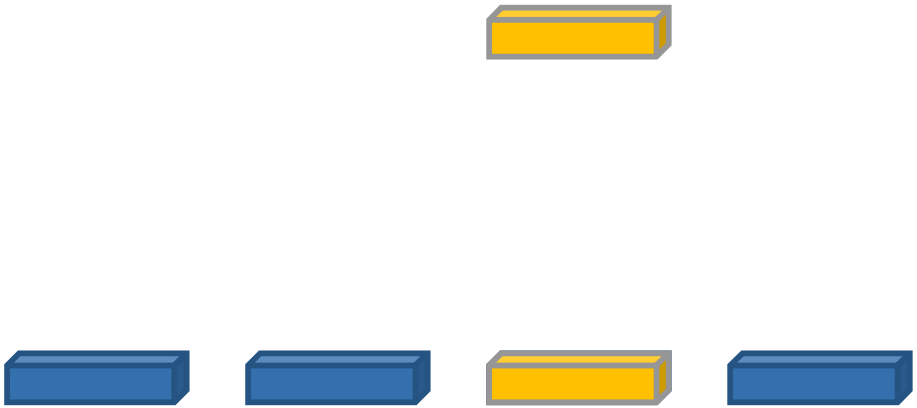
Inhalte des Vortrags

1. **Motivation**
2. **Das Prinzip**
3. **Rückblick**
4. **Methoden in 11g**
5. **Methoden in 12c**
6. **ILM**
7. **Fazit**

Partitionierung verringert die abzufragende Datenmenge.



Mit **Partition Exchange Load** lässt sich der Aufbau der Daten asynchron durchführen.

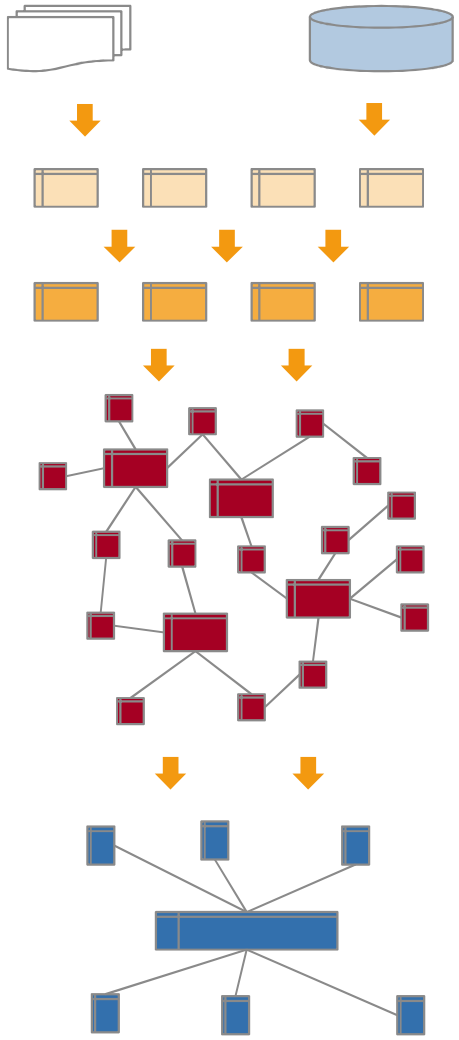


Partitionierung bringt Benefit im dimensionalen Data Warehouse.

- Partition Pruning
- Bessere Performance
- Ordnung
- Sicherheit
- Höhere Verfügbarkeit
- Vereinfachung der Verwaltung

Partitionierung findet man in allen Bereichen des dimensionalen Data Warehouses.

- Staging/Cleansing Areas ✓
- Core DWH ✓
- Dimensionen ✓
- Fakten ✓



1997 führte ORACLE die erste Partitionierungsmethode ein.

Staging Core Dimensionen Fakten

8.0: Range-Partitioning und **Range-Partitioned Global Indexes**

```
CREATE TABLE fact_sales
( sale_id  NUMBER(5)
, …
, sale_date    DATE
) PARTITION BY RANGE(sales_date)
( PARTITION sales_jan2014 VALUES LESS THAN(TO_DATE('02/01/2014','MM/DD/YYYY'))
, PARTITION sales_feb2014 VALUES LESS THAN(TO_DATE('03/01/2014','MM/DD/YYYY'))
, PARTITION sales_mar2014 VALUES LESS THAN(TO_DATE('04/01/2014','MM/DD/YYYY')));
```

```
CREATE INDEX month_ix ON fact_sales(sales_month)
GLOBAL PARTITION BY RANGE(sales_month)
(PARTITION pm1_ix VALUES LESS THAN (2)
 PARTITION pm2_ix VALUES LESS THAN (3)
 PARTITION pm3_ix VALUES LESS THAN (4)
 PARTITION pm4_ix VALUES LESS THAN (5)
 PARTITION pm5_ix VALUES LESS THAN (6)
 PARTITION pm6_ix VALUES LESS THAN (7)
 PARTITION pm7_ix VALUES LESS THAN (8)
 PARTITION pm8_ix VALUES LESS THAN (9)
 PARTITION pm9_ix VALUES LESS THAN (10)
 PARTITION pm10_ix VALUES LESS THAN (11)
 PARTITION pm11_ix VALUES LESS THAN (12)
 PARTITION pm12_ix VALUES LESS THAN (MAXVALUE));
```

Hash-Partitionierung sorgt für eine Gleichverteilung der Daten.

Staging Core Dimensionen Fakten

8i:
Hash-Partitioning
und die erste
Composite-Methode Range-Hash

```
CREATE TABLE sta_customer
( id   NUMBER
, name VARCHAR2 (60)
, …
) PARTITION BY HASH (id)
PARTITIONS 4
STORE IN (ts1, ts2, ts3, ts4);
```

```
CREATE TABLE dim_product
( prod_id NUMBER
, prod_no NUMBER
, prod_name VARCHAR(32)
, price NUMBER
) PARTITION BY RANGE ( prod_no )
SUBPARTITION BY HASH ( prod_name )
SUBPARTITIONS 8 STORE IN (ts1, ts3, ts5, ts7)
( PARTITION p1 VALUES LESS THAN (1000)
, PARTITION p2 VALUES LESS THAN (2000)
, PARTITION p3 VALUES LESS THAN (MAXVALUE));
```

List-Partitionierung hilft diskrete Werte zu strukturieren.

Staging Core Dimensionen Fakten

9i:
List-Partitioning

```
CREATE TABLE sales_list
( salesman_id  NUMBER(5)
, salesman_name VARCHAR2(30)
, sales_state   VARCHAR2(20)
, sales_amount  NUMBER(10)
, sales_date    DATE
) PARTITION BY LIST(sales_state)
( PARTITION sales_west VALUES('California', 'Hawaii')
, PARTITION sales_east VALUES ('New York', 'Virginia', 'Florida')
, PARTITION sales_central VALUES('Texas', 'Illinois'),
PARTITION sales_other VALUES(DEFAULT)
);
```

und mit Release 2 dann
Range-List

```
CREATE TABLE quarterly_regional_sales
( ...
, sale_date DATE
, state_code VARCHAR2(2)
) PARTITION BY RANGE (sale_date)
SUBPARTITION BY LIST (state_code)
SUBPARTITION TEMPLATE
( SUBPARTITION east VALUES('NY', 'VA', 'FL') TABLESPACE ts1
, SUBPARTITION west VALUES('CA', 'OR', 'HI') TABLESPACE ts2
, SUBPARTITION central VALUES('IL', 'TX', 'MO') TABLESPACE ts3
, ... )
( PARTITION q1_2000 VALUES LESS THAN (TO_DATE('1-APR-2000','DD-MON-YYYY'))
, PARTITION q2_2000 VALUES LESS THAN (TO_DATE('1-JUL-2000','DD-MON-YYYY'))
, PARTITION q3_2000 VALUES LESS THAN (TO_DATE('1-OCT-2000','DD-MON-YYYY'))
, ... );
```

Auch Indizes können seit 10g global Hash-partitioniert werden.
Zudem wurde die Verwaltung erheblich verbessert.

10g:
Global Hash Partitioned Index

```
CREATE INDEX i_knd_nachname
    ON kunden (nachname)
GLOBAL PARTITION BY HASH (nachname)
PARTITIONS 4;
```

Maintenance of Global Partitioned Indexes

- ADD (HASH)
- COALESCE (HASH)
- DROP
- EXCHANGE
- MERGE
- MOVE
- SPLIT
- TRUNCATE

und mit Release 2 waren eine Million Partitionen pro Tabelle möglich.

Interval-Partitionierung mit 11g hilft bei der Verwaltung von Range-Partitionen.

Staging Dimensionen Fakten

ORA-14400:
Inserted partition key does not map to any partition

1 Interval-Partitioning

```
ALTER TABLE sales_range
SET INTERVAL (NUMTOYMINTERVAL(1, 'MONTH'));

ALTER TABLE sales_range
RENAME PARTITION sys_p123 TO sales_may2000;
```

2 Noch **mehr Composite-Partitioning**

I. Range-Range
II. List-Range
III. Hash-Hash
… *List-Hash* … *List-List*

Mit 11g sind weitere Composite-Partitionierungsmethoden hinzugekommen.

Staging Core Dimensionen Fakten

I) Range-Range

```
CREATE TABLE account_balance_history
( id                  NUMBER NOT NULL
, account_number      NUMBER NOT NULL
, customer_id         NUMBER NOT NULL
, transaction_date    DATE NOT NULL
, amount_credited     NUMBER
, amount_debited      NUMBER
, end_of_day_balance NUMBER NOT NULL
) PARTITION BY RANGE (transaction_date) INTERVAL (NUMTODSINTERVAL(7,'DAY'))
SUBPARTITION BY RANGE(end_of_day_balance)
SUBPARTITION TEMPLATE
( SUBPARTITION unacceptable VALUES LESS THAN (-1000)
, SUBPARTITION credit VALUES LESS THAN (0)
, SUBPARTITION low VALUES LESS THAN (500)
, SUBPARTITION normal VALUES LESS THAN (5000)
, SUBPARTITION high VALUES LESS THAN (20000)
, SUBPARTITION extraordinary VALUES LESS THAN (MAXVALUE))
( PARTITION p0 VALUES LESS THAN (TO_DATE('01-JAN-2007','dd-MON-yyyy')));
```

Mit 11g sind weitere Composite-Partitionierungsmethoden hinzugekommen.

Staging Core Dimensionen Fakten

II) List-Range

```
CREATE TABLE donations
( id              NUMBER
, ...
, currency        VARCHAR2(3)
, amount          NUMBER
) PARTITION BY LIST (currency)
SUBPARTITION BY RANGE (amount)
( PARTITION p_eur VALUES ('EUR')
( SUBPARTITION p_eur_small VALUES LESS THAN (8)
, SUBPARTITION p_eur_medium VALUES LESS THAN (80)
, SUBPARTITION p_eur_high VALUES LESS THAN (MAXVALUE)
),PARTITION p_gbp VALUES ('GBP')
( ...
),PARTITION p_aud_nzd_chf VALUES ('AUD','NZD','CHF')
( SUBPARTITION p_aud_nzd_chf_small VALUES LESS THAN (12)
, ...
),PARTITION p_jpy VALUES ('JPY')
( SUBPARTITION p_jpy_small VALUES LESS THAN (1200)
, ...
),PARTITION p_inr VALUES ('INR')
( SUBPARTITION p_inr_small VALUES LESS THAN (400)
, ...)  )
, PARTITION p_zar VALUES ('ZAR')
( SUBPARTITION p_zar_small VALUES LESS THAN (70)
, ...
),PARTITION p_default VALUES (DEFAULT)
( SUBPARTITION p_default_small VALUES LESS THAN (10)
, SUBPARTITION p_default_medium VALUES LESS THAN (100)
, SUBPARTITION p_default_high VALUES LESS THAN (MAXVALUE)
))
ENABLE ROW MOVEMENT;
```

Mit 11g sind weitere Composite-Partitionierungsmethoden hinzugekommen.

Staging Core Dimensionen Fakten

III) Hash-Hash

```
CREATE TABLE accounts
( id NUMBER
, account_number NUMBER
, customer_id NUMBER
, balance NUMBER
, branch_id NUMBER
, region VARCHAR(2)
, status VARCHAR2(1)
) PARTITION BY HASH (id)
PARTITIONS 4
SUBPARTITION BY HASH (customer_id)
SUBPARTITIONS 8;
```

… List-Hash … List-List

Partitionierung ist mit 11g auch auf virtuelle Spalten möglich

Staging Core Dimensionen Fakten

3)
Partitionen auf virtuelle Spalten

Ein Beispiel mit RANGE-RANGE-Partitionierung auf Fakten

```
CREATE TABLE fact_sales
( sale_id NUMBER
, prod_id NUMBER(6) NOT NULL
, cust_id NUMBER NOT NULL
, time_id DATE NOT NULL
, channel_id CHAR(1) NOT NULL
, promo_id NUMBER(6) NOT NULL
, quantity_sold NUMBER(3) NOT NULL
, amount_sold NUMBER(10,2) NOT NULL
, total_amount AS (quantity_sold * amount_sold)
) PARTITION BY RANGE (time_id) INTERVAL (NUMTOYMINTERVAL(1,'MONTH'))
SUBPARTITION BY RANGE (total_amount)
SUBPARTITION TEMPLATE
( SUBPARTITION p_small   VALUES LESS THAN (1000)
, SUBPARTITION p_medium  VALUES LESS THAN (5000)
, SUBPARTITION p_large   VALUES LESS THAN (10000)
, SUBPARTITION p_extreme VALUES LESS THAN (MAXVALUE)
)
( PARTITION sales_before_2014 VALUES LESS THAN(TO_DATE('01-JAN-2014','dd-MON-yyyy'))
)
ENABLE ROW MOVEMENT
PARALLEL
NOLOGGING;
```

11g: REFERENCE-Partitionierung kann bei Master-Detail-Beziehungen eingesetzt werden.

Staging Core Dimensionen Fakten

4) REFERENCE-Partitioning

```
-- =========================
-- Referenced table (master)
-- =========================
CREATE TABLE hash_products
( product_id          NUMBER(6)    PRIMARY KEY
, product_name        VARCHAR2(50)
, ...
)
PARTITION BY HASH (product_id)
PARTITIONS 4
STORE IN (tbs_01, tbs_02, tbs_03, tbs_04);


-- ====================================
-- Reference partitioned table (detail)
-- ====================================
CREATE TABLE part_order_items
( order_id        NUMBER(12) PRIMARY KEY
, product_id      NUMBER(6) NOT NULL
, ...
, CONSTRAINT product_id_fk FOREIGN KEY (product_id) REFERENCES hash_products(product_id)
)
PARTITION BY REFERENCE (product_id_fk);
```

11g: Partitionierung lässt sich auch manuell durchführen.

Staging Core Dimensionen Fakten

5) SYSTEM-Partitioning

```
CREATE TABLE tabelle_system_partitioniert
( id                    NUMBER       NOT NULL
, bez                   VARCHAR2(255) NOT NULL
, kbez                  VARCHAR2(8)  NOT NULL
, datum                 DATE
)
) PARTITION BY SYSTEM
( PARTITION p1 TABLESPACE users
, PARTITION p2 TABLESPACE users);
```

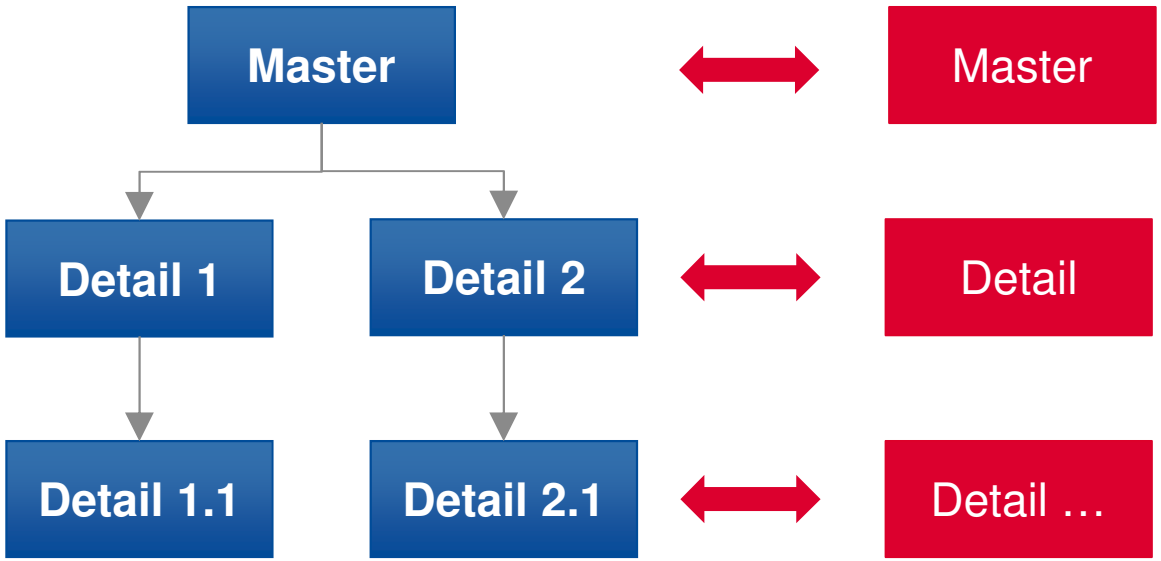
```
INSERT INTO tabelle_system_partitioniert
   VALUES (1, 'lange Bezeichnung', 'Kurzbez.', SYSDATE);

* ERROR at line 1: ORA-14701: partition-extended name or bind variable must be used for
DMLs on tables partitioned by the System method
```

```
-- richtig:
INSERT INTO tabelle_system_partitioniert PARTITION (p1)
   VALUES (1, 'lange Bezeichnung', 'Kurzbez.', SYSDATE);
```

REFERENCE-Partitionierung wurde mit 12c erheblich erweitert.

1 Drei Erweiterungen der REFERENCE-Partitionierung

- Die Master-Tabelle kann auch INTERVAL-partitioniert sein
- Die CASCADE-Option gibt es auch für die TRUNCATE-Operation auf die Master-Tabelle
- Die CASCADE-Option gibt es auch für Partition-EXCHANGE-Operationen, so dass die Partitionen der Detail-Tabelle an die Änderung der Master-Tabelle angepasst wird.



12c: Die ONLINE-Verfügbarkeit wurde ebenfalls verbessert

2 Online Move Partition

- Keine geblockten DML-Transaktionen bei z.B. ALTER PARTITION MOVE
  Jetzt funktioniert dies ONLINE ohne Ausfall!
- Merge und Split-Operationen effektiver

```
ALTER TABLE sales SPLIT PARTITION sales_Q4_2007 INTO
( PARTITION sales_Q4_2007
     VALUES LESS THAN (TO_DATE('01-JAN-2008','dd-MON-yyyy'))
, PARTITION sales_Q1_2008
     VALUES LESS THAN (TO_DATE('01-APR-2008','dd-MON-yyyy'))
, PARTITION sales_Q2_2008
     VALUES LESS THAN (TO_DATE('01-JUL-2008','dd-MON-yyyy'))
, PARTITION sales_Q3_2008
     VALUES LESS THAN (TO_DATE('01-OCT-2008','dd-MON-yyyy'))
, PARTITION sales_Q4_2008);

ALTER TABLE sales  MERGE PARTITIONS sales_Q1_2008
, sales_Q2_2008
, sales_Q3_2008
, sales_Q4_2008
INTO PARTITION sales_2008;
```

- ADD, DROP, TRUNCATE PARTITION auf mehrere Partitionen

12c: Die ONLINE-Verfügbarkeit wurde ebenfalls verbessert

3 Asynchronous Global Index Maintenance

- Für DROP und TRUNCATE PARTITION
- Kein UNUSABLE-Status mehr
- PMO_DEFERRED_GIDX_MAINT_JOB
- Mit DBMS_PART.CLEANUP_GIDX
  oder
  ALTER INDEX REBUILD|COALESCE

Überblick über die Methoden zur Partitionierung

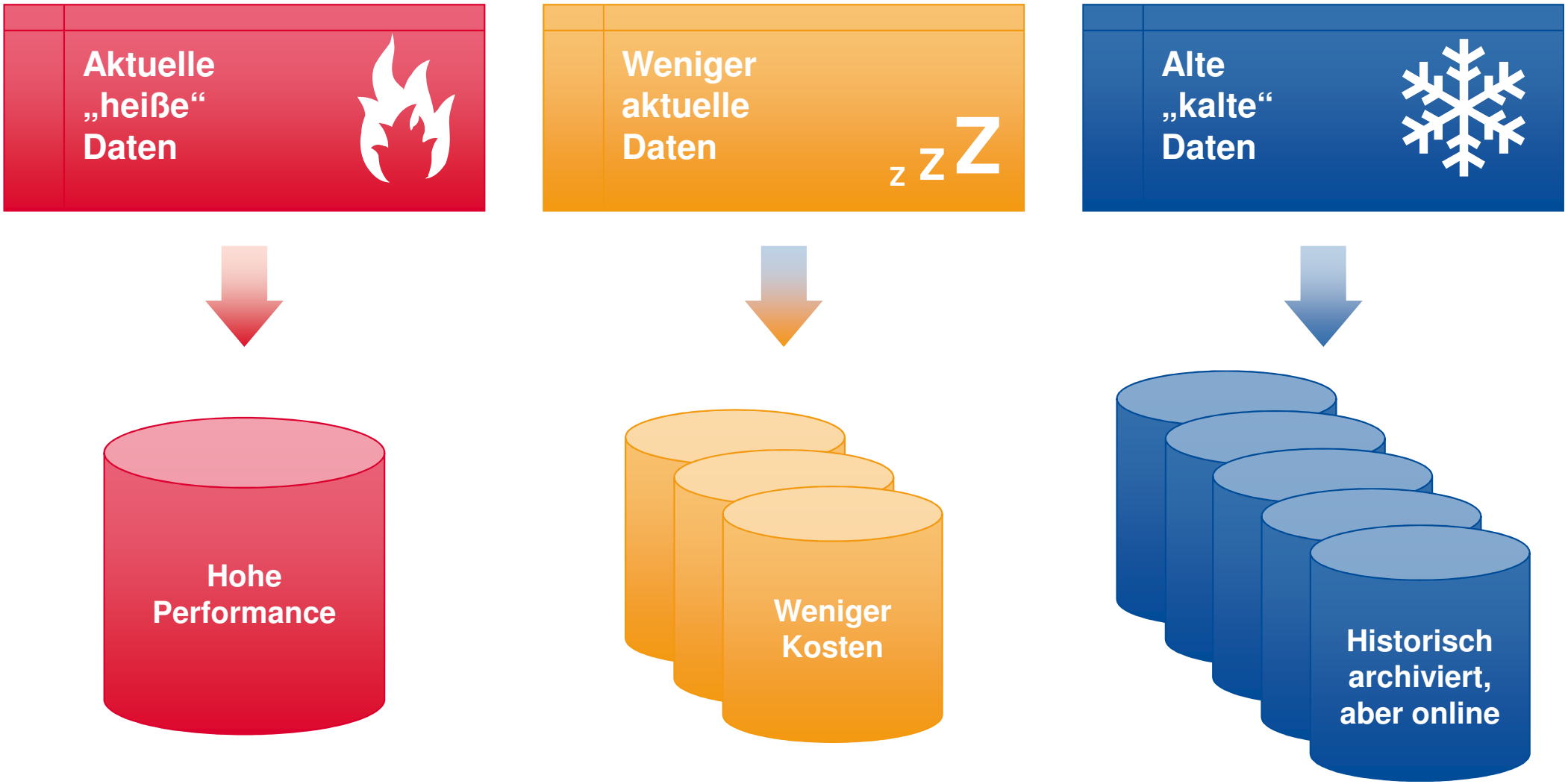
Version	Partitionierungsmethode
8.0	Range, Global-Range-Partitioned Index
8i	Hash, Range-Hash
9i	List, Range-List
10g	Global-Hash-Partitioned Index, Maintenance GPI
11g	Interval, Reference, Range-Range, List-Range, Hash-Hash, List-Hash, List-List, Virtual Columns, System
12c	Reference-Extensions, Online Move Partition, Asynchronous Global Index Maintenance

Die Verwaltung von Partitionen wird erheblich vereinfacht.

- Online Move Partition
- Interval-Partitioning
- Partition Maintenance Operations
- Asynchronous Global Index Maintenance
- Exchange Partition

ILM kann die Kosten reduzieren.

Information Lifecycle Management



Fazit.

- Nützliche mächtige Features zur Partitionierung in 11g und 12c
- Hilfe bei Ladevorgängen, Backup, Archivierung, Performanceoptimierung und Administration
- Das bestehende Data Warehouse auf die neuen Möglichkeiten hin untersuchen.

!!! Einsetzen !!!

Literatur.

- Oracle® Database – VLDB and Partitioning Guide 11g Release 2 (11.2) E16541-05
- Oracle Partitioning with Oracle Database 12c – Efficient Data Management and Performance Acceleration for every System
- An Oracle White Paper – October 2010 Partitioning with Oracle Database 11g Release 2
- docs.oracle.com
- Partitionierung für Fortgeschrittene von Frank Schneede, ORACLE Deutschland B.V. & Co. KG
- Partitionspflege mit Oracle 11g leicht gemacht von Frank Schneede, ORACLE Deutschland GmbH
- http://www.databasejournal.com/features/oracle/oracle-database-12c-new-features-for-partitioned-tables.html
- Information Lifecycle Management for Business Data An Oracle White Paper June 2007
- ORACLE DATABASE 12c: INFORMATION LIFECYCLE MANAGEMENT http://www.oracle.com/us/solutions/sap/nl23-db12c-ilm-en-2209394.pdf

metafinanz unterstützt Ihre Kunden im **Bereich Oracle und DWH** end-to-end: Von Entscheidung und Konzeption bis hin zu Einführung und Optimierung.

DECIDE | PLAN | BUILD | OPTIMIZE

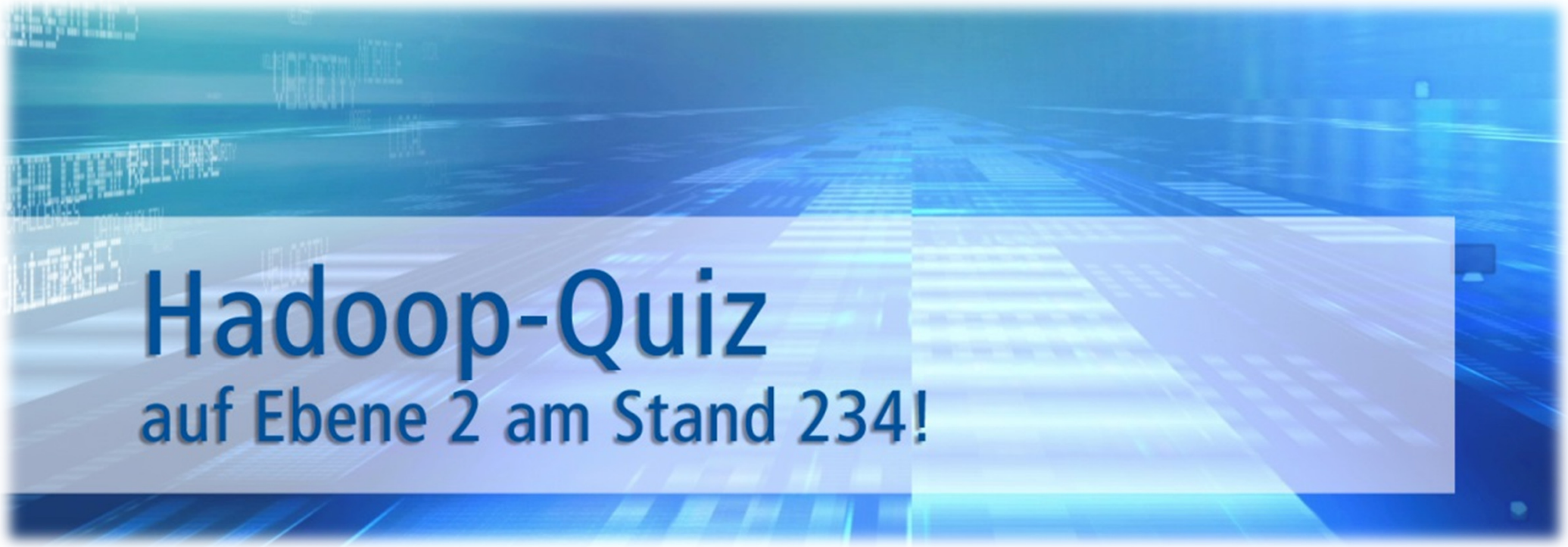
metafinanz ist herstellerunabhängig, erfahren, kompetent, zuverlässig und lieferfähig

metafinanz Consulting Leistungen rund um Data Warehouse und Enterprise Information Management

- Businessanalyse, Systemauswahl und Integrationskonzept
- Gesamtheitliche IT Architektur Strategie und Designkonzeption (IT Strategie)
- Customizing von Big Data Lösungen für Ihre besonderen Business Anforderungen (Zukunftsfähigkeit)
- DWH Configuration & Optimierung für mehr Leistungsfähigkeit (Effizienzpotential)

Umsetzung DWH Projekte	▪ Anforderungsanalysen ▪ Architekturdesign ▪ ETL (PL/SQL, OWB, SAS DI, …)
DWH Tuning	▪ Analyse ▪ SQL & Physikalisches Tuning ▪ Prozessoptimierung
Integration Hadoop	▪ Gesamtarchitektur & Use Case Discovery ▪ DataMart, ETL & Computation Offload
OWB	▪ ETL Implementierung ▪ Migrationskonzeption und Umsetzung
Reporting	▪ Business- und Anforderungsanalysen ▪ Standard Reporting & Analytics ▪ SAS (classic & Visual Analytics), Cognos, R

Interesse? Fragen? Austausch?



Treffen Sie uns an unserem Stand
… und gewinnen ein iPad Mini!

metafinanz

Herzlichen Dank!

metafinanz Informationssysteme GmbH
Leopoldstr. 146
Phone: +49 89 360531-0
Fax: +49 89 350531-5015
Email: kontakt@metafinanz.de
www.metafinanz.de

Besuchen Sie uns auch auf: