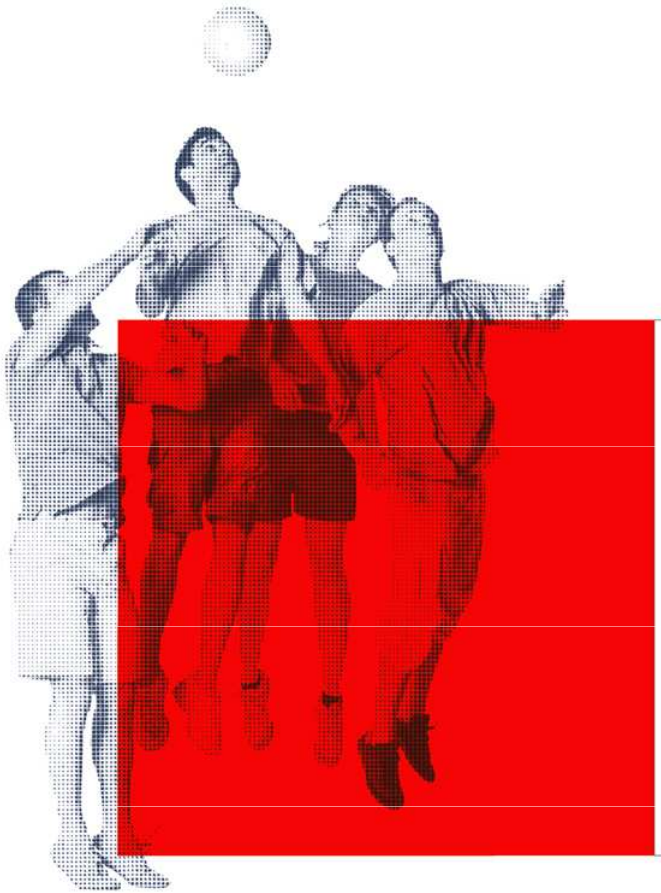
Partitioning Your Oracle Data Warehouse – Just a Simple Task?



Dani Schnider
Principal Consultant
Business Intelligence
dani.schnider@trivadis.com

Oracle Open World 2009,
San Francisco

trivadis
makes IT easier.

About Dani Schnider

- Principal Consultant at Trivadis AG, Zurich, Switzerland
  - Consulting, coaching and development of data warehouse projects for several customers
  - dani.schnider@trivadis.com
- Trainer for Trivadis courses
  - Data Warehousing with Oracle
  - SQL Performance Tuning & Optimizer Workshop
  - Oracle Warehouse Builder
- Working…
  - … with databases since 1990
  - … with Oracle since 1994
  - … with Data Warehouses since 1997
  - … for Trivadis since 1999

About Trivadis

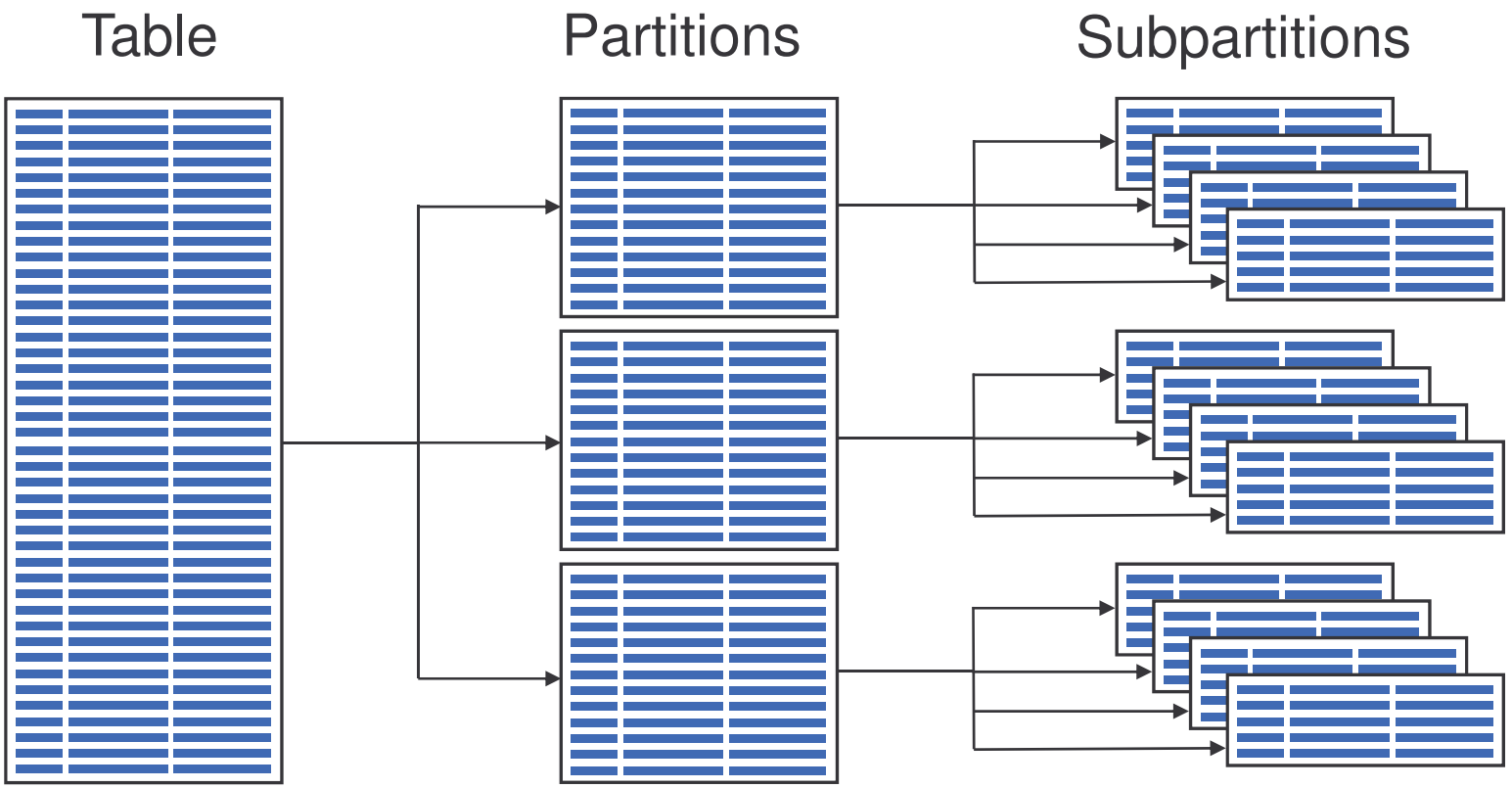
- Swiss IT consulting company
  - Technical consulting with focus on Oracle, SQL Server and DB/2
  - 13 locations in Switzerland, Germany and Austria
  - ~ 550 employees
- Key figures 2008
  - Services for more than 650 clients in over 1‘600 projects
  - Over 150 Service Level Agreements
  - More than 5'000 training participants

Agenda

- **Partitioning Concepts**
- The Right Partition Key
- Large Dimensions
- Partition Maintenance

Partitioning – The Basic Idea

- Decompose tables/indexes into smaller pieces



Partitioning Methods in Oracle

		Partition		
		RANGE	HASH	LIST
Subpartition	(none)	Oracle8	Oracle8i	Oracle9i
	RANGE	ORACLE 11g DATABASE		ORACLE 11g DATABASE
	HASH	Oracle8i		ORACLE 11g DATABASE
	LIST	Oracle9i		ORACLE 11g DATABASE

- Additionally: Interval Partitioning, Reference Partitioning, Virtual Column-Based Partitioning

Benefits of Partitioning

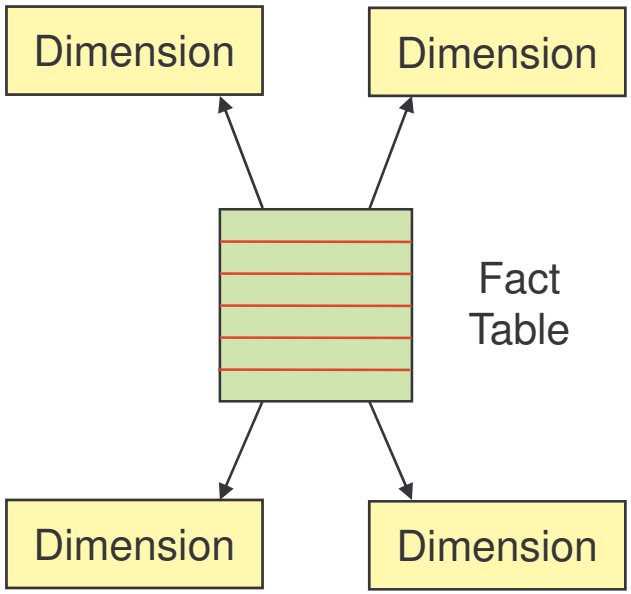
- Partition Pruning
  - Reduce I/O: Only relevant partitions have to be accessed
- Partition-Wise Joins
  - Full PWJ: Join two equi-partitioned tables (parallel or serial)
  - Partial PWJ: Join partitioned with non-partitioned table (parallel)
- Rolling History
  - Create new partitions in periodical intervals
  - Drop old partitions in periodical intervals
- Manageability
  - Backups, statistics gathering, compressing on individual partitions

Agenda

- Partitioning Concepts
- **The Right Partition Key**
- Large Dimensions
- Partition Maintenance

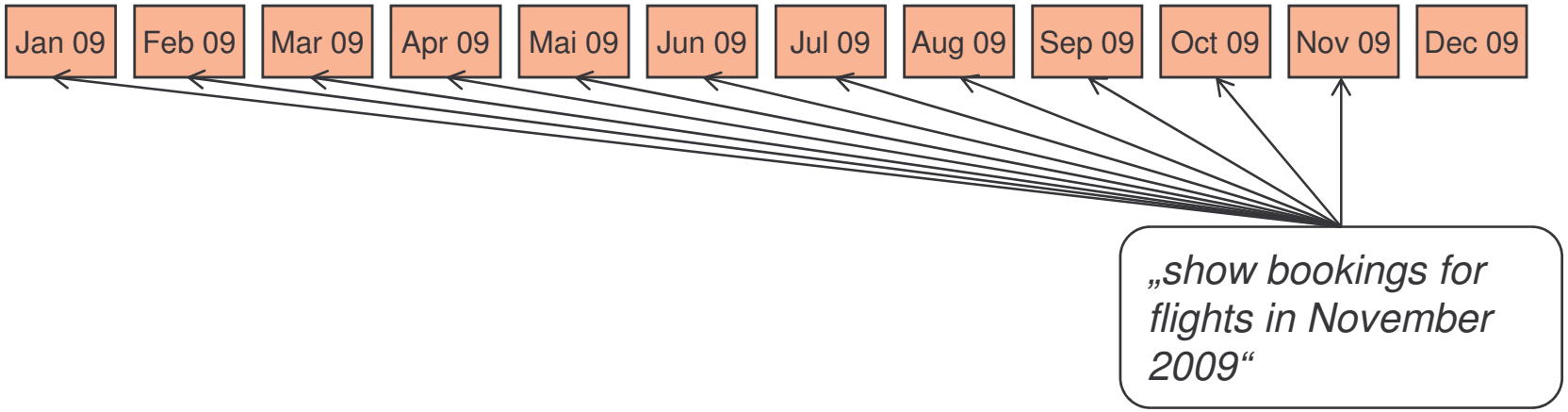
Partition Methods and Partition Keys

- Important questions for Partitioning:
  - Which tables should be partitioned?
  - Which partition method should be used?
  - Which partition key(s) should be used?
- Partition key is important for
  - Query optimization (partition pruning, partition-wise joins)
  - ETL performance (partition exchange, rolling history)
- Typically in data warehouses
  - RANGE partitioning of fact tables on DATE column
  - **But which DATE column?**



Practical Example 1: Airline Company

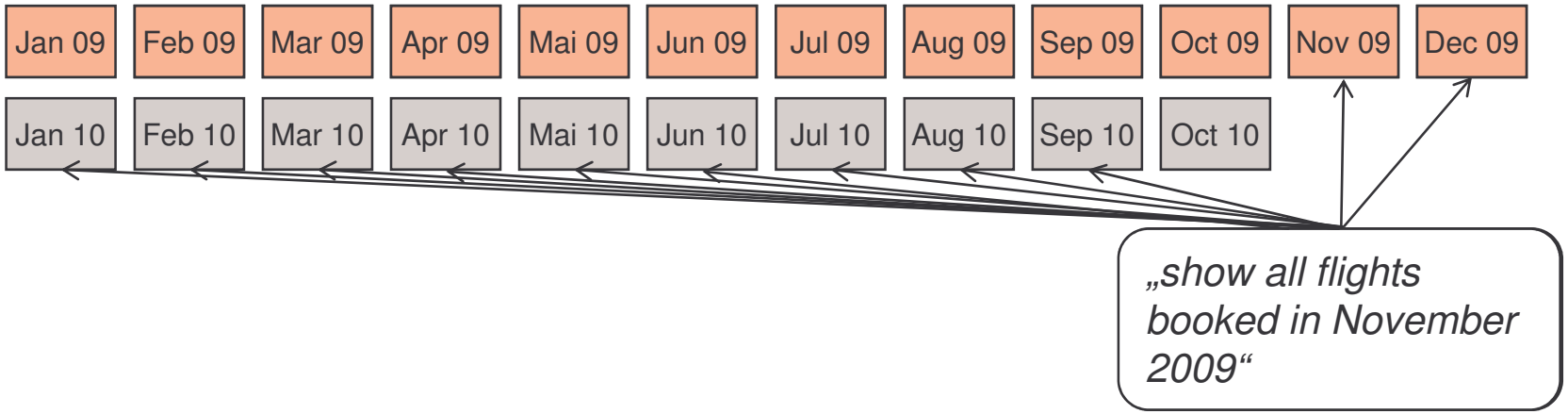
- Flight bookings are stored in partitioned table
- RANGE partitioning per month, partition key: booking date



- Problem: Most of the reports are based on the flight date
  - Flights can be booked 11 months ahead
  - 11 partitions must be read of one particular flight date

Practical Example 1: Airline Company

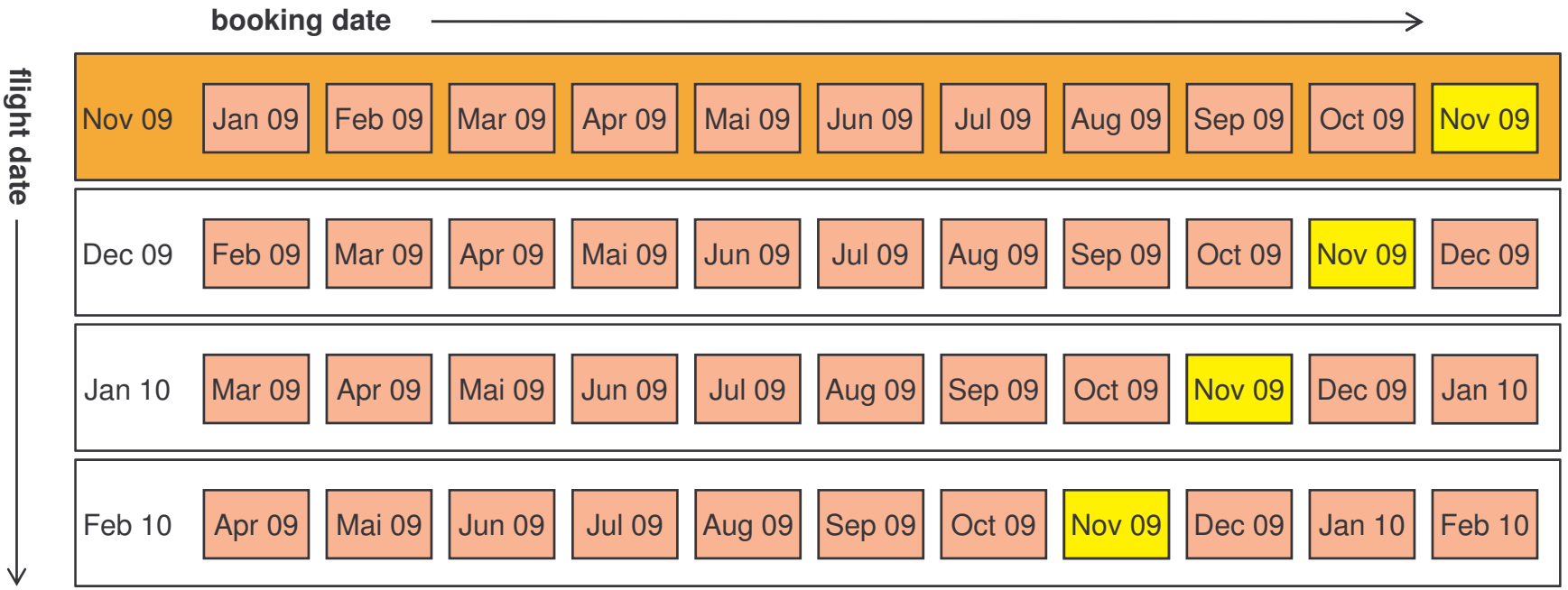
- Solution: partition key flight date instead of booking date
- Data is loaded into current and future partitions



- Reports based on flight date read only one partition
- Reports based on booking date must read 11 (small) partitions

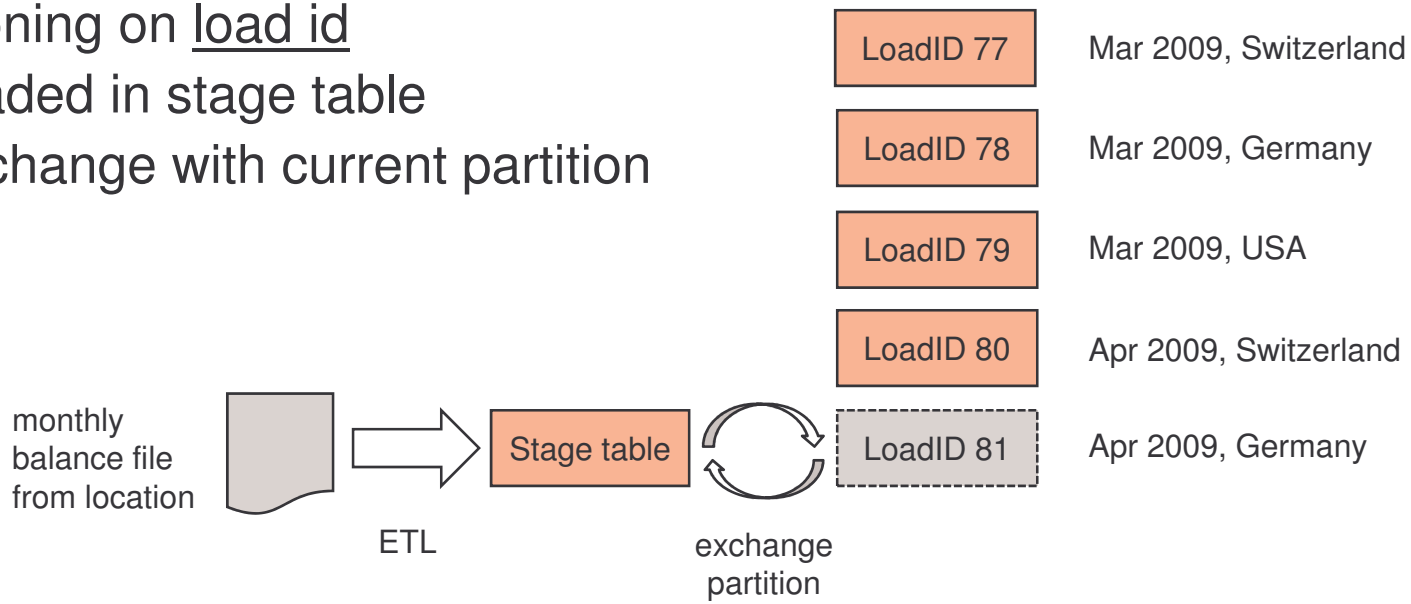
Practical Example 1: Airline Company

- Better solution: Composite RANGE-RANGE partitioning
  - RANGE partitioning on flight date
  - RANGE **sub**partitioning on booking date
- More flexibility for reports on flight date and/or on booking date



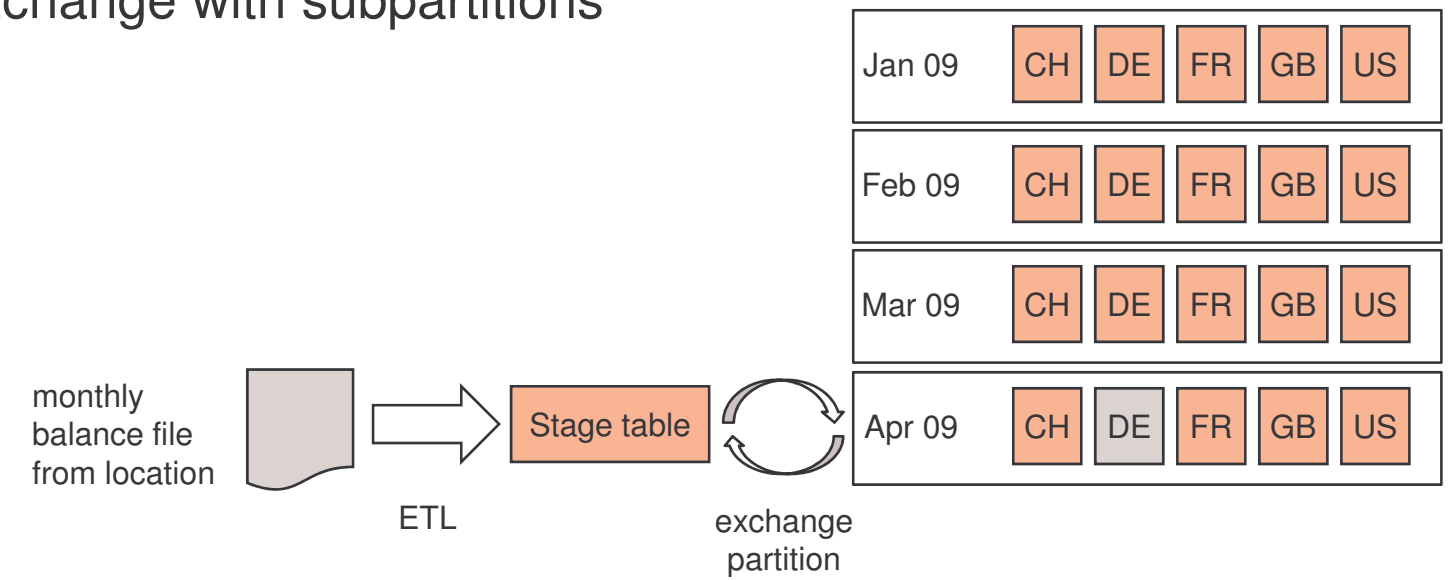
Practical Example 2: International Bank

- Account balance data of international customers
  - Monthly files of different locations (countries)
  - Correction files replace last delivery for the same month/country
- Original approach
  - Technical load id for each combination of month/country
  - LIST partitioning on load id
  - Files are loaded in stage table
  - Partition exchange with current partition



Practical Example 2: International Bank

- Problem: partition key load id is useless for queries
  - Queries are based on balance date
- Solution
  - RANGE partitioning on balance date
  - LIST subpartitions on country code
  - Partition exchange with subpartitions

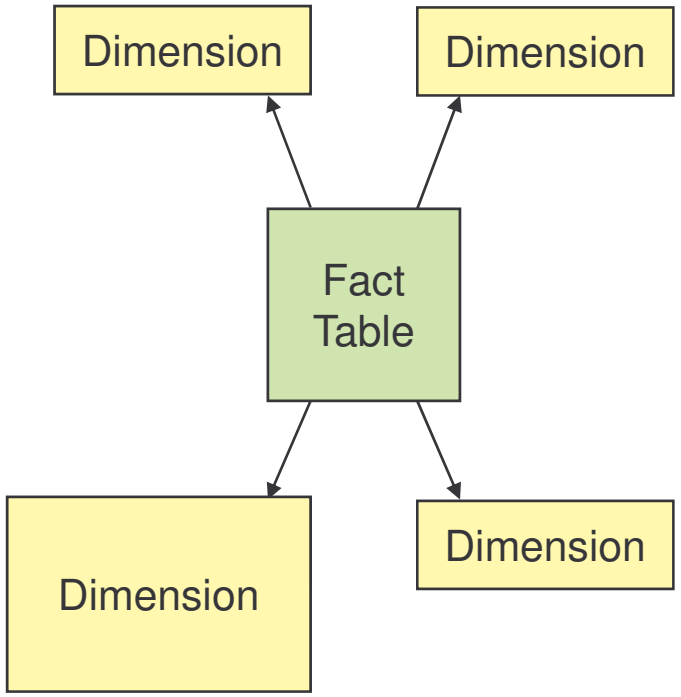


Agenda

- Partitioning Concepts
- The Right Partition Key
- **Large Dimensions**
- Partition Maintenance

Partitioning in Star Schema

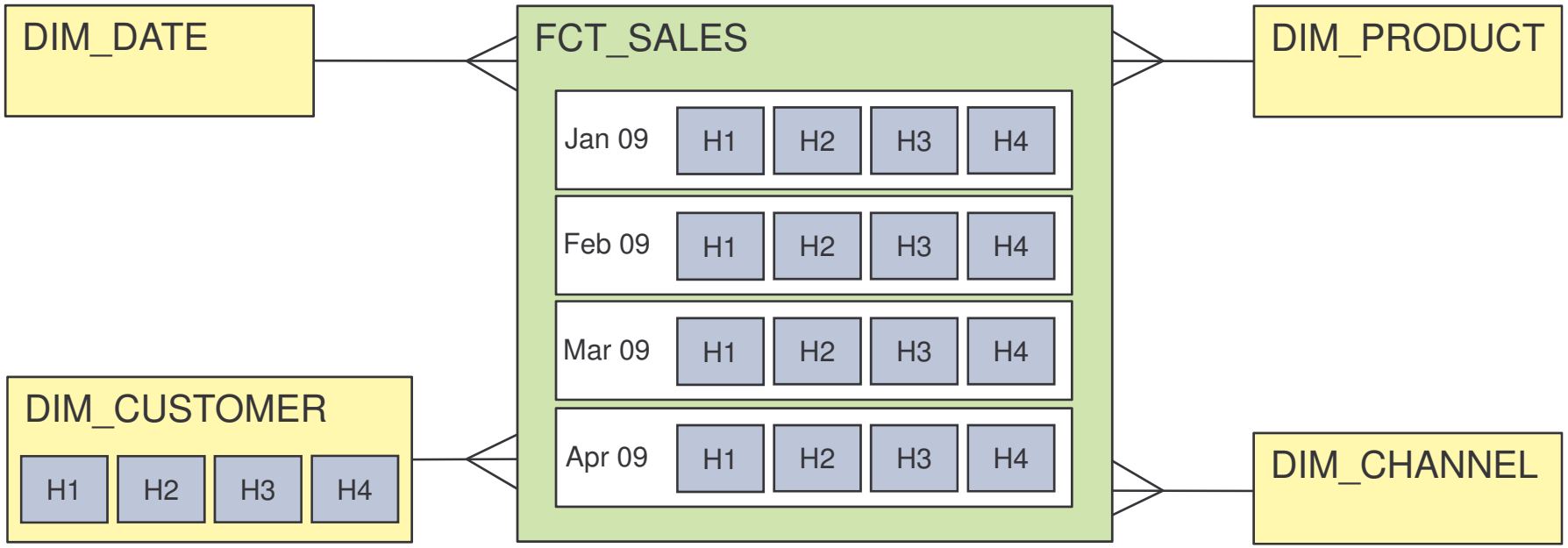
- Fact Table
  - Usually „big“ (millions to billions of rows)
  - RANGE partitioning by DATE column
- Dimension Tables
  - Usually „small“ (10 to 10000 rows)
  - In most cases not partitioned



- But how about large dimensions?
  - e.g. customer dimension with millions of rows

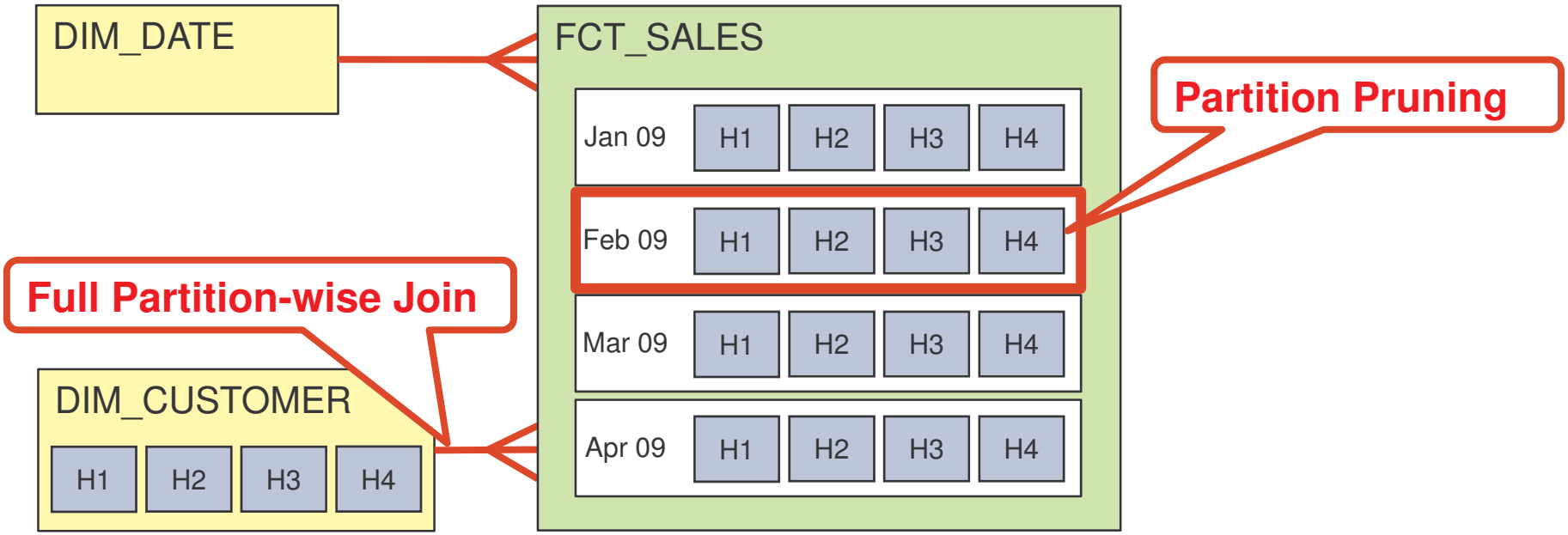
HASH Partitioning on Large Dimension

- DIM_CUSTOMER
  - HASH Partitioning
  - Partition Key: CUSTOMER_ID
- FCT_SALES
  - Composite RANGE-HASH Partitioning
  - Partition Key: SALES_DATE
  - Subpartition Key: CUSTOMER_ID

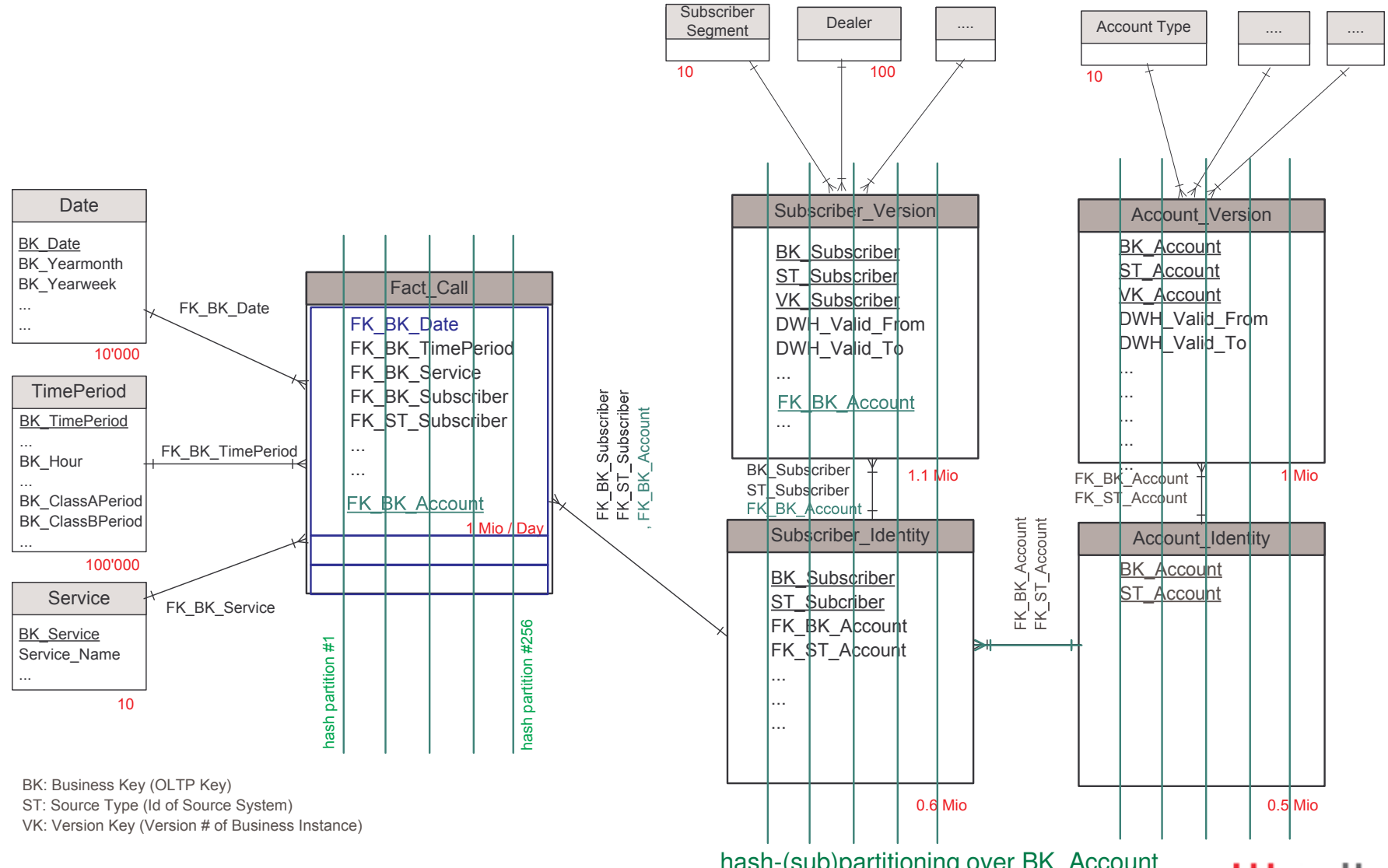


HASH Partitioning on Large Dimension

```
SELECT d.cal_month, c.country_name, SUM(f.amount)
  FROM fct_sales f
  JOIN dim_date d ON (d.cal_date = f.sales_date)
  JOIN dim_customer c ON (c.customer_id = f.customer_id)
 WHERE d.cal_month = 'Feb-2009'
   AND c.country_code = 'DE'
GROUP BY d.cal_month, c.country_name;
```



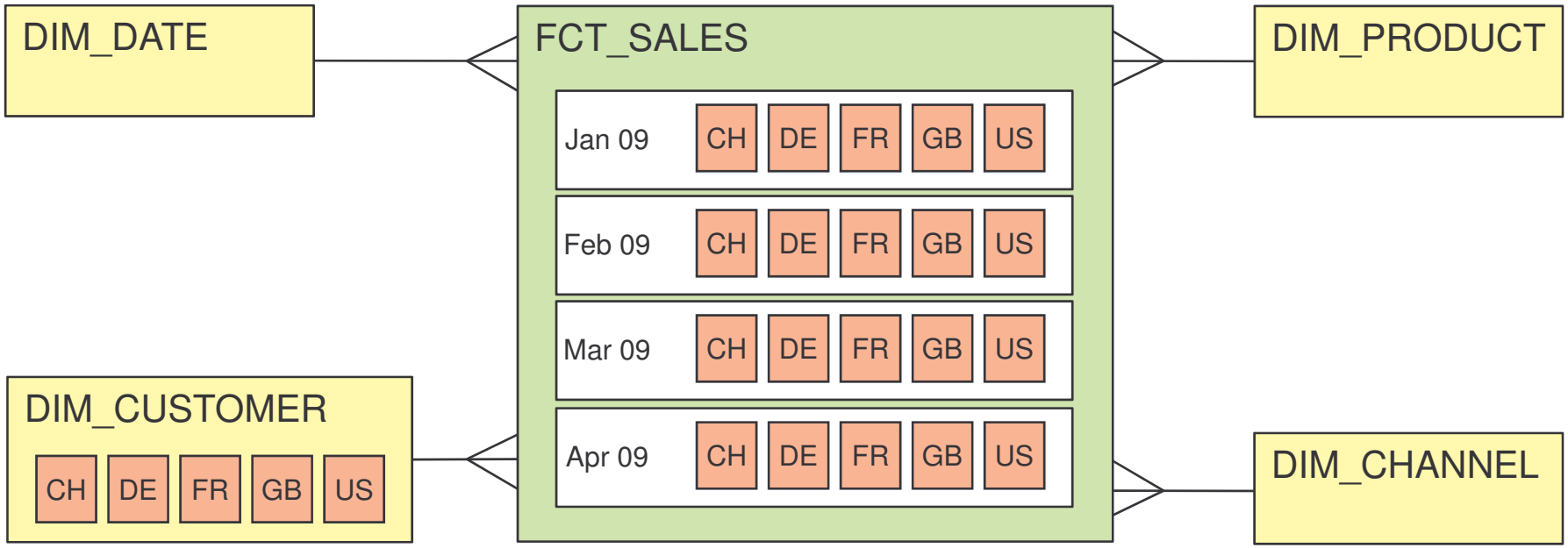
Practical Example 3: Telecommunication Company



BK: Business Key (OLTP Key)
ST: Source Type (Id of Source System)
VK: Version Key (Version # of Business Instance)

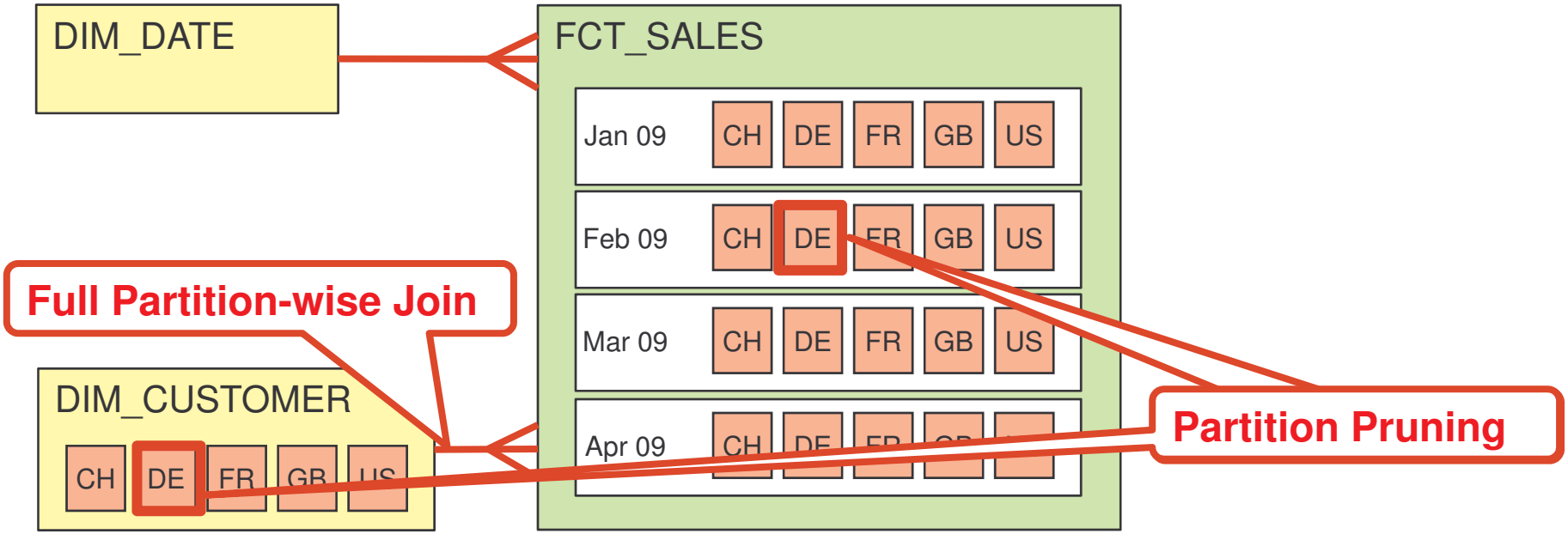
LIST Partitioning on Large Dimension

- DIM_CUSTOMER
  - LIST Partitioning
  - Partition Key: COUNTRY_CODE
- FCT_SALES
  - Composite RANGE-LIST Partitioning
  - Partition Key: SALES_DATE
  - Subpartition Key: COUNTRY_CODE (denormalized column in fact table)



LIST Partitioning on Large Dimension

```
SELECT d.cal_month, c.country_name, SUM(f.amount)
  FROM fct_sales f
  JOIN dim_date d ON (d.cal_date = f.sales_date)
  JOIN dim_customer c ON (c.customer_id = f.customer_id
                          AND c.country_code = f.country_code)
 WHERE d.cal_month = 'Feb-2009'
   AND c.country_code = 'DE'
GROUP BY d.cal_month, c.country_name;
```



Join-Filter Pruning

- New approach for partition pruning on join conditions
  - A bloom filter is created based on the dimension table restriction
  - Dynamic partition pruning based on that bloom filter

```
-------------------------------------------------------------------------------
| Id  | Operation                      | Name          | Rows  | Pstart| Pstop |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |               |     1 |       |       |
|   1 |  HASH GROUP BY                 |               |     1 |       |       |
|*  2 |   HASH JOIN                    |               |  1886 |       |       |
|*  3 |    HASH JOIN                   |               |  1886 |       |       |
|   4 |     PART JOIN FILTER CREATE    | :BF0000       |    30 |       |       |
|*  5 |      TABLE ACCESS FULL         | DIM_DATE      |    30 |       |       |
|   6 |     PARTITION RANGE JOIN-FILTER|               | 21675 |:BF0000|:BF0000|
|   7 |      PARTITION LIST SINGLE     |               | 21675 |   KEY |   KEY |
|   8 |       TABLE ACCESS FULL        | FCT_SALES     | 21675 |   KEY |   KEY |
|   9 |    PARTITION LIST SINGLE       |               |  8220 |   KEY |   KEY |
|  10 |     TABLE ACCESS FULL          | DIM_CUSTOMER  |  8220 |     2 |     2 |
-------------------------------------------------------------------------------
```

Agenda

- Partitioning Concepts
- The Right Partition Key
- Large Dimensions
- **Partition Maintenance**

Practical Example 3: Monthly Partition Maintenance

- Requirements
  - Monthly partitions on all fact tables, daily inserts into current partitions
  - 3 years of history (36 partitions per table)
  - Table compression to increase full table scan performance
  - Backup of current partitions only

TS_01 Jan 08	TS_02 Feb 08	TS_03 Mar 08	TS_04 Apr 08	TS_05 Mai 08	TS_06 Jun 08	TS_07 Jul 08	TS_08 Aug 08	TS_09 Sep 08	TS_10 Oct 08	TS_11 Nov 08	TS_12 Dec 08
TS_13 Jan 09	TS_14 Feb 09	TS_15 Mar 09	TS_16 Apr 09	TS_17 Mai 09	TS_18 Jun 09	TS_19 Jul 09	TS_20 Aug 09	TS_21 **Sep 09**	TS_22 Oct 06	TS_23 Nov 06	TS_24 Dec 06
TS_25 Jan 07	TS_26 Feb 07	TS_27 Mar 07	TS_28 Apr 07	TS_29 Mai 07	TS_30 Jun 07	TS_31 Jul 07	TS_32 Aug 07	TS_33 Sep 07	TS_34 Oct 07	TS_35 Nov 07	TS_36 Dec 07

Practical Example 3: Monthly Partition Maintenance

1. Set next tablespace to read-write

```
ALTER TABLESPACE ts_22
READ WRITE;
```

TS_01	TS_02	TS_03	TS_04	TS_05	TS_06	TS_07	TS_08	TS_09	TS_10	TS_11	TS_12
Jan 08	Feb 08	Mar 08	Apr 08	Mai 08	Jun 08	Jul 08	Aug 08	Sep 08	Oct 08	Nov 08	Dec 08
TS_13	TS_14	TS_15	TS_16	TS_17	TS_18	TS_19	TS_20	TS_21	TS_22	TS_23	TS_24
Jan 09	Feb 09	Mar 09	Apr 09	Mai 09	Jun 09	Jul 09	Aug 09	**Sep 09**	Oct 06	Nov 06	Dec 06
TS_25	TS_26	TS_27	TS_28	TS_29	TS_30	TS_31	TS_32	TS_33	TS_34	TS_35	TS_36
Jan 07	Feb 07	Mar 07	Apr 07	Mai 07	Jun 07	Jul 07	Aug 07	Sep 07	Oct 07	Nov 07	Dec 07

Practical Example 3: Monthly Partition Maintenance

1. Set next tablespace to read-write
2. Drop oldest partition

```
ALTER TABLE sales
DROP PARTITION p_oct_2006;
```

TS_01 Jan 08	TS_02 Feb 08	TS_03 Mar 08	TS_04 Apr 08	TS_05 Mai 08	TS_06 Jun 08	TS_07 Jul 08	TS_08 Aug 08	TS_09 Sep 08	TS_10 Oct 08	TS_11 Nov 08	TS_12 Dec 08
TS_13 Jan 09	TS_14 Feb 09	TS_15 Mar 09	TS_16 Apr 09	TS_17 Mai 09	TS_18 Jun 09	TS_19 Jul 09	TS_20 Aug 09	TS_21 **Sep 09**	TS_22	TS_23 Nov 06	TS_24 Dec 06
TS_25 Jan 07	TS_26 Feb 07	TS_27 Mar 07	TS_28 Apr 07	TS_29 Mai 07	TS_30 Jun 07	TS_31 Jul 07	TS_32 Aug 07	TS_33 Sep 07	TS_34 Oct 07	TS_35 Nov 07	TS_36 Dec 07

Practical Example 3: Monthly Partition Maintenance

1. Set next tablespace to read-write
2. Drop oldest partition
3. Create new partition for next month

```
ALTER TABLE sales
ADD PARTITION p_oct_2009
VALUES LESS THAN
(TO_DATE('01-NOV-2009',
         'DD-MON-YYYY'))
TABLESPACE ts_22;
```

TS_01 Jan 08	TS_02 Feb 08	TS_03 Mar 08	TS_04 Apr 08	TS_05 Mai 08	TS_06 Jun 08	TS_07 Jul 08	TS_08 Aug 08	TS_09 Sep 08	TS_10 Oct 08	TS_11 Nov 08	TS_12 Dec 08
TS_13 Jan 09	TS_14 Feb 09	TS_15 Mar 09	TS_16 Apr 09	TS_17 Mai 09	TS_18 Jun 09	TS_19 Jul 09	TS_20 Aug 09	TS_21 **Sep 09**	TS_22 **Oct 09**	TS_23 Nov 06	TS_24 Dec 06
TS_25 Jan 07	TS_26 Feb 07	TS_27 Mar 07	TS_28 Apr 07	TS_29 Mai 07	TS_30 Jun 07	TS_31 Jul 07	TS_32 Aug 07	TS_33 Sep 07	TS_34 Oct 07	TS_35 Nov 07	TS_36 Dec 07

Practical Example 3: Monthly Partition Maintenance

1. Set next tablespace to read-write
2. Drop oldest partition
3. Create new partition for next month
4. Compress current partition

```
ALTER TABLE sales
MOVE PARTITION p_sep_2009
COMPRESS;
```

TS_01	TS_02	TS_03	TS_04	TS_05	TS_06	TS_07	TS_08	TS_09	TS_10	TS_11	TS_12
Jan 08	Feb 08	Mar 08	Apr 08	Mai 08	Jun 08	Jul 08	Aug 08	Sep 08	Oct 08	Nov 08	Dec 08
TS_13	TS_14	TS_15	TS_16	TS_17	TS_18	TS_19	TS_20	TS_21	TS_22	TS_23	TS_24
Jan 09	Feb 09	Mar 09	Apr 09	Mai 09	Jun 09	Jul 09	Aug 09	Sep 09	**Oct 09**	Nov 06	Dec 06
TS_25	TS_26	TS_27	TS_28	TS_29	TS_30	TS_31	TS_32	TS_33	TS_34	TS_35	TS_36
Jan 07	Feb 07	Mar 07	Apr 07	Mai 07	Jun 07	Jul 07	Aug 07	Sep 07	Oct 07	Nov 07	Dec 07

Practical Example 3: Monthly Partition Maintenance

1. Set next tablespace to read-write
2. Drop oldest partition
3. Create new partition for next month
4. Compress current partition
5. Set tablespace to read-only

```
ALTER TABLESPACE ts_21
READ ONLY;
```

TS_01 Jan 08	TS_02 Feb 08	TS_03 Mar 08	TS_04 Apr 08	TS_05 Mai 08	TS_06 Jun 08	TS_07 Jul 08	TS_08 Aug 08	TS_09 Sep 08	TS_10 Oct 08	TS_11 Nov 08	TS_12 Dec 08
TS_13 Jan 09	TS_14 Feb 09	TS_15 Mar 09	TS_16 Apr 09	TS_17 Mai 09	TS_18 Jun 09	TS_19 Jul 09	TS_20 Aug 09	TS_21 Sep 09	TS_22 **Oct 09**	TS_23 Nov 06	TS_24 Dec 06
TS_25 Jan 07	TS_26 Feb 07	TS_27 Mar 07	TS_28 Apr 07	TS_29 Mai 07	TS_30 Jun 07	TS_31 Jul 07	TS_32 Aug 07	TS_33 Sep 07	TS_34 Oct 07	TS_35 Nov 07	TS_36 Dec 07

Interval Partitioning

- In Oracle11g, the creation of new partitions can be automated
- Example:

```
CREATE TABLE sales (
  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
)
PARTITION BY RANGE (time_id)
INTERVAL(numtoyminterval(1, 'MONTH'))
STORE IN (ts_01,ts_02,ts_03,ts_04)
(
  PARTITION p_before_1_jan_2005
    VALUES LESS THAN (to_date('01-01-2008','dd-mm-yyyy'))
)
```

Gathering Optimizer Statistics

- DBMS_STATS parameter GRANULARITY defines statistics level

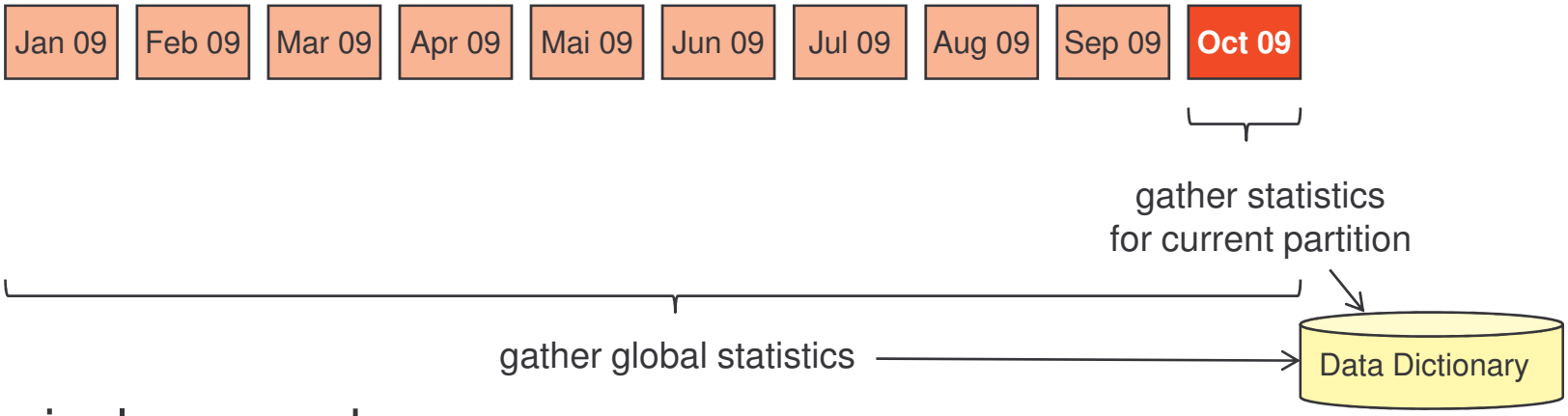
GRANULARITY Parameter Value	Table Statistics	Partition Statistics	Subpartition Statistics
GLOBAL	☑		
GLOBAL AND PARTITION	☑	☑	
PARTITION		☑	
SUBPARTITION			☑
ALL	☑	☑	☑
AUTO	☑	☑	(☑)
APPROX_GLOBAL AND PARTITION	☑	☑	

- Example:

```
dbms_stats.gather_table_stats
    (ownname      => USER
    ,tabname      => 'SALES'
    ,granularity => 'GLOBAL AND PARTITION');
```

Problem of Global Statistics

- Global statistics are essential for good execution plans
  - **num_distinct, low_value, high_value, density, histograms**
- Gathering global statistics is time-consuming
  - All partitions must be scanned



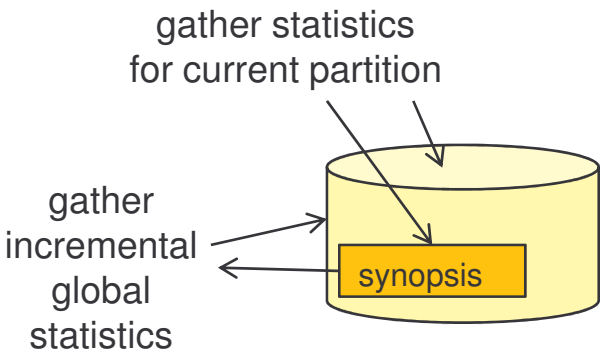
- Typical approach
  - After loading data, only modified partition statistics are gathered
  - Global statistics are gathered on regular time base (e.g. weekends)

Incremental Global Statistics

- Synopsis-based gathering of statistics
- For each partition a synopsis is stored in SYSAUX tablespace
  - Statistics metadata for partition and columns of partition
- Global statistics by aggregating the synopses from each partition

- Activate incremental global statistics:

```
dbms_stats.set_table_prefs
    (ownname => USER
    ,tabname => 'SALES'
    ,pname   => 'incremental'
    ,pvalue  => 'true');
```



Partitioning Your Data Warehouse – Core Messages

- Oracle Partitioning is a powerful option – not only for data warehouses
- The concept is simple, but the reality can be complex
- Many new partitioning features added in Oracle Database 11g
  - ☺ New Composite Partitioning methods
  - ☺ Interval Partitioning
  - ☺ Join-Filter Pruning
  - ☺ Incremental Global Statistics

Thank you!