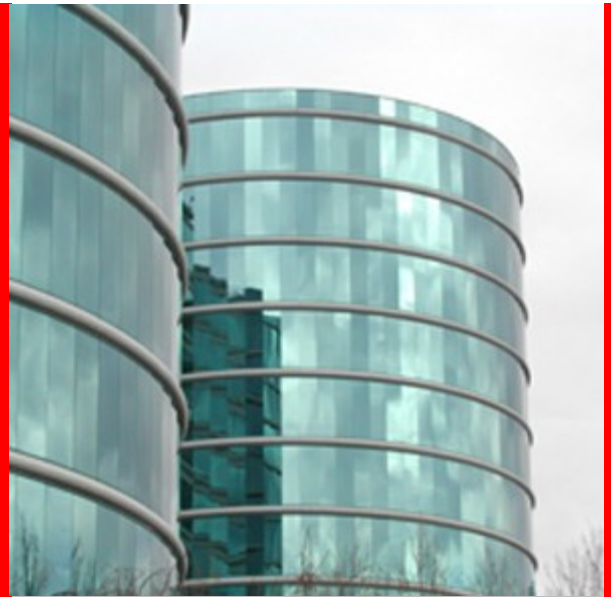
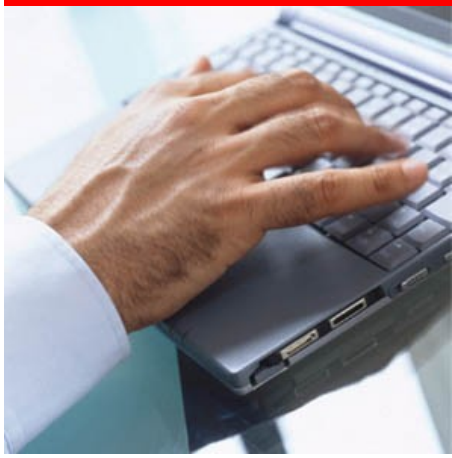


ORACLE

Things you always wanted to know about Oracle Partitioning

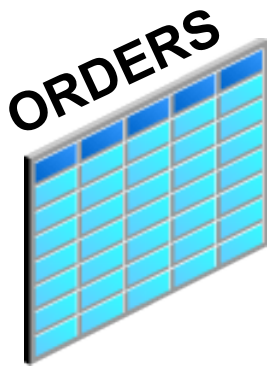
Hermann Baer
Director Product Management, Data Warehousing

Agenda



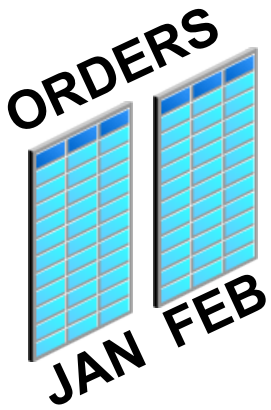
- Partitioning in a nutshell
  - “Basics in 5 minutes”
- Partitioning benefits
  - Faster - cheaper
- Partitioning for your business needs
  - Better - new functionality of Oracle Database 11g
- Things you might not know ..
  - Focus on Interval Partitioning versus Range Partitioning
- Q&A

The Concept of Partitioning
Simple Yet Powerful



Large Table

Difficult to Manage

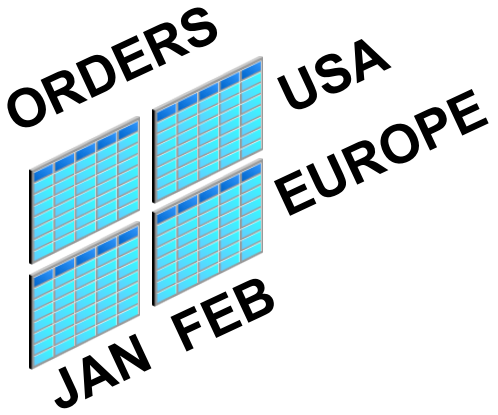


Partition

Divide and Conquer

Easier to Manage

Improve Performance



Composite Partition

Better Performance

More flexibility to match business needs

Transparent to applications

What is Oracle Partitioning?

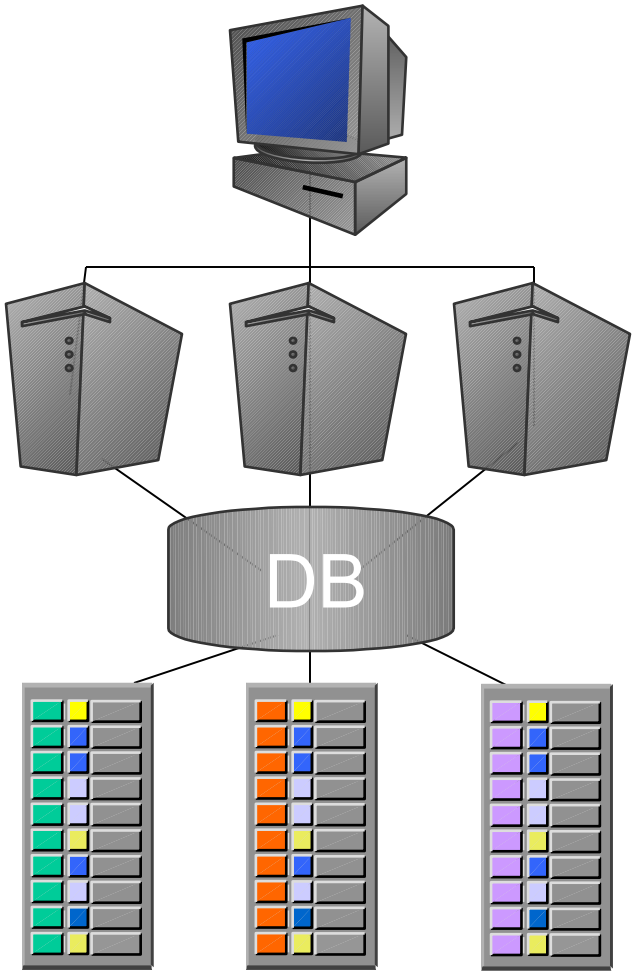
It is

- Powerful functionality to logically partition objects into smaller pieces
- Only driven by business requirements
- Partitioning for Performance, Manageability, and Availability

It is not

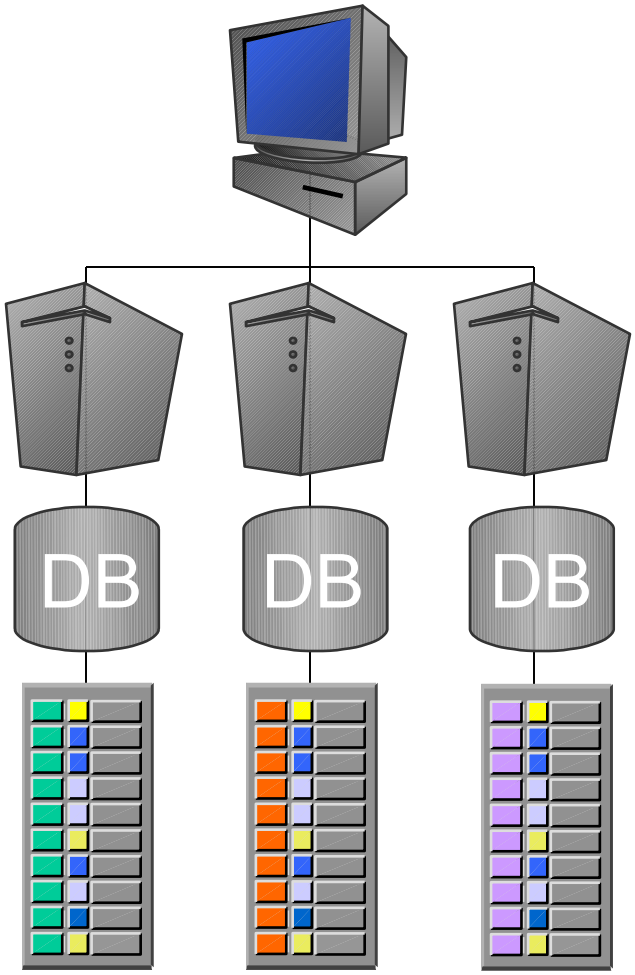
- Just a way to physically divide – or clump - any large data set into smaller buckets
- Enabling pre-requirement to support a specific hardware/software design
  - Hash mandatory for shared nothing systems

Physical versus Logical Partitioning
Shared Everything Architecture - Oracle



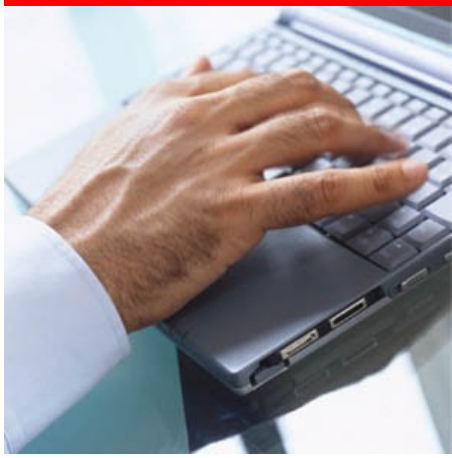
- **Logical Partitioning**
- Does not underlie any constraints
  - SMP, MPP, Cluster, Grid does not matter
- Purely based on the business requirement
  - Availability. Manageability, Performance
- Beneficial for every environment
  - Provides the most comprehensive functionality

Physical versus Logical Partitioning
Shared Nothing Architecture



- **Physical Partitioning**
- Fundamental system setup requirement
  - Node owns piece of DB
- Enables parallelism
  - Number of partitions is equivalent to min. parallelism
- Always needs HASH distribution
  - Equally sized partitions per node required for proper load balancing

Agenda



- Partitioning in a nutshell
  - “Basics in 5 minutes”
- Partitioning benefits
  - Faster - cheaper
- Partitioning for your business needs
  - Better - new functionality of Oracle Database 11g
- Things you might not know ..
  - Focus on Interval Partitioning versus Range Partitioning
- Q&A

Faster

“Only work on the data that is relevant”

- Data Access
- Data Load
- Data Maintenance

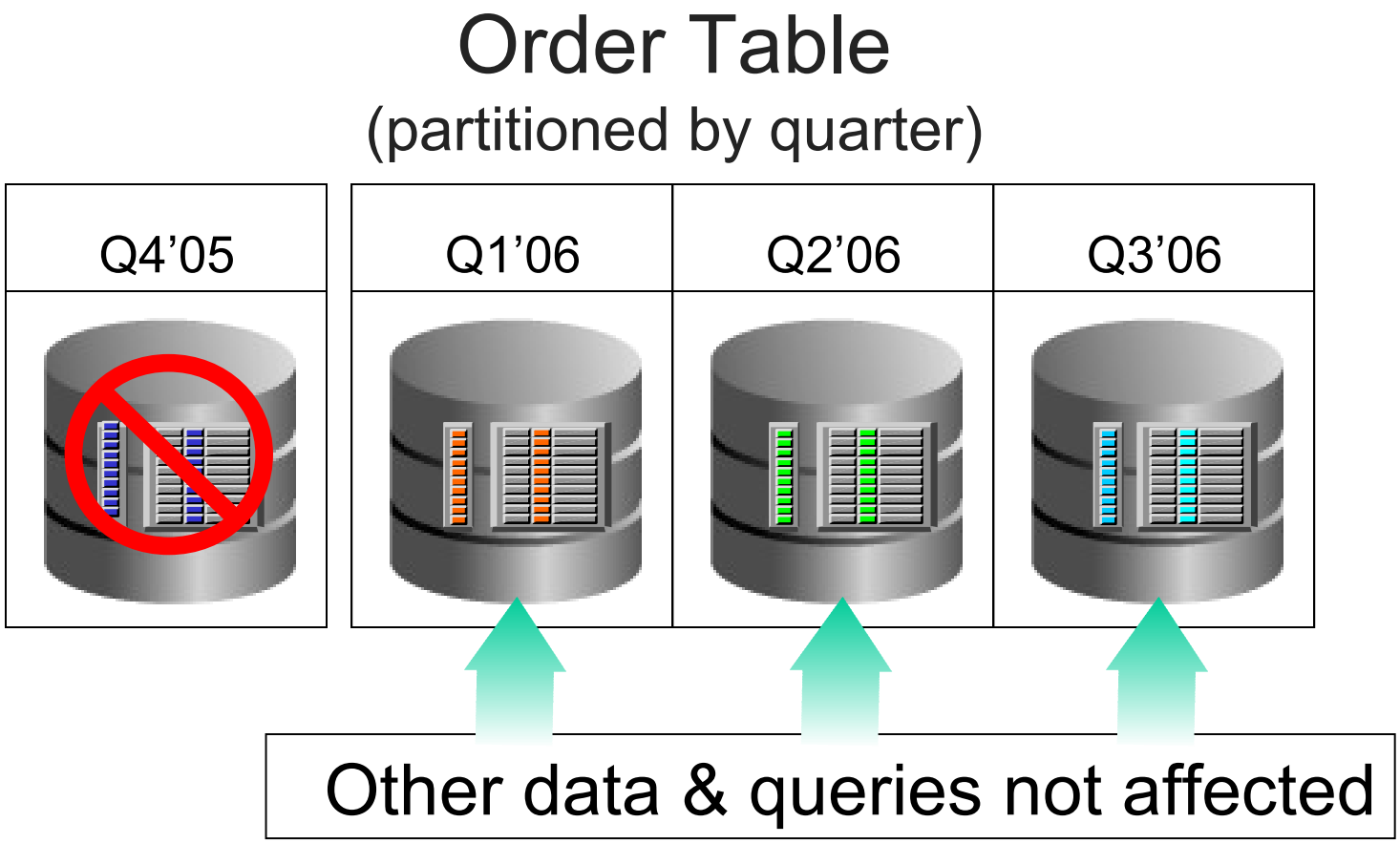
- Faster response times
- Less system resources required
- Support for more users and/or a higher workload

Partition for Performance

- Only relevant partitions will be accessed
  - Static pruning with known values in advance
  - Dynamic pruning uses internal recursive SQL to find the relevant partitions
- Minimizes I/O operations
  - Provides massive performance gains

```
SELECT sum(sales_amount)
FROM   sales
WHERE sales_date
BETWEEN '01-MAR-2006' AND '31-MAY-2006';
```

Partition for Manageability/Availability

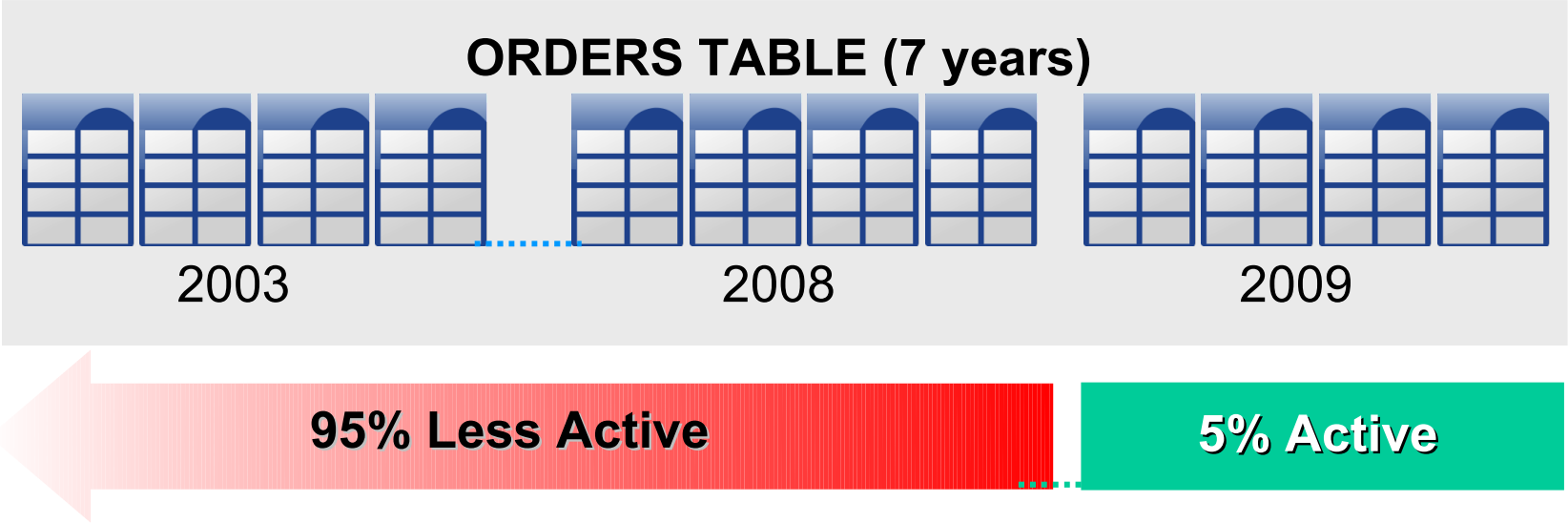


Cheaper

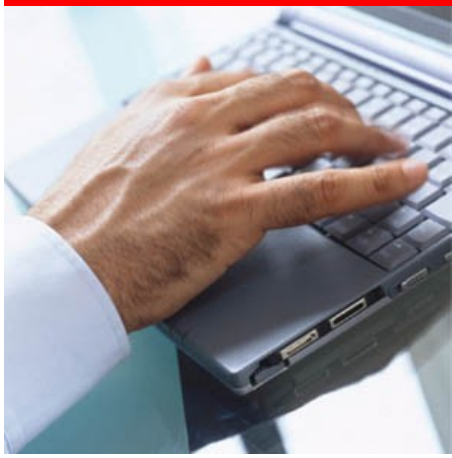
“Store data in the most appropriate manner”

- Find the balance between data importance, storage performance, storage reliability, and storage form

Partition for Tiered Storage



Agenda



- Partitioning in a nutshell
  - “Basics in 5 minutes”
- Partitioning benefits
  - Faster - cheaper
- Partitioning for your business needs
  - Better - new functionality of Oracle Database 11g
- Things you might not know ..
  - Focus on Interval Partitioning versus Range Partitioning
- Q&A

Model for your Business Needs

Power and flexibility comes with choices

- Partitioning method
  - Range – List – Hash
  - Single level and composite level Partitioning
- Partitioning key
  - Real,virtual, or even referenced column(s),
- Partition granularity
  - Year – month – day – hour - minute

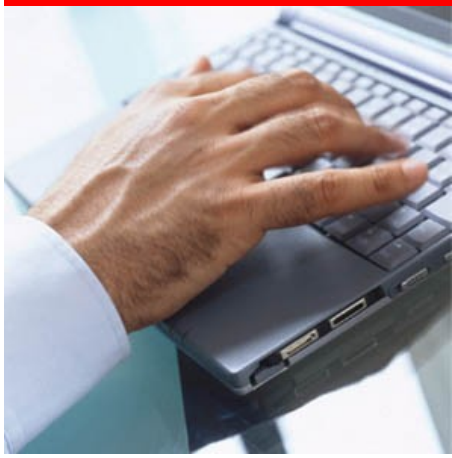
Transparency is equally important

- Concurrent and automatic partition maintenance operations
  - Partition creation on demand

Oracle Partitioning:
Over Ten Years of Development

	Core functionality	Performance	Manageability
Oracle8	Range partitioning Global range indexes	"Static" partition pruning	Basic maintenance operations: add, drop, exchange
Oracle8*i*	Hash and composite range-hash partitioning	Partition-wise joins "Dynamic" pruning	Merge operation
Oracle9*i*	List partitioning		Global index maintenance
Oracle9*i* R2	Composite range-list partitioning	Fast partition split	
Oracle10*g*	Global hash indexes		Local Index maintenance
Oracle10*g* R2	1M partitions per table	"Multi-dimensional" pruning	Fast drop table
Oracle Database 11*g*	More composite choices REF Partitioning Virtual Column		Interval Partitioning Partition Advisor

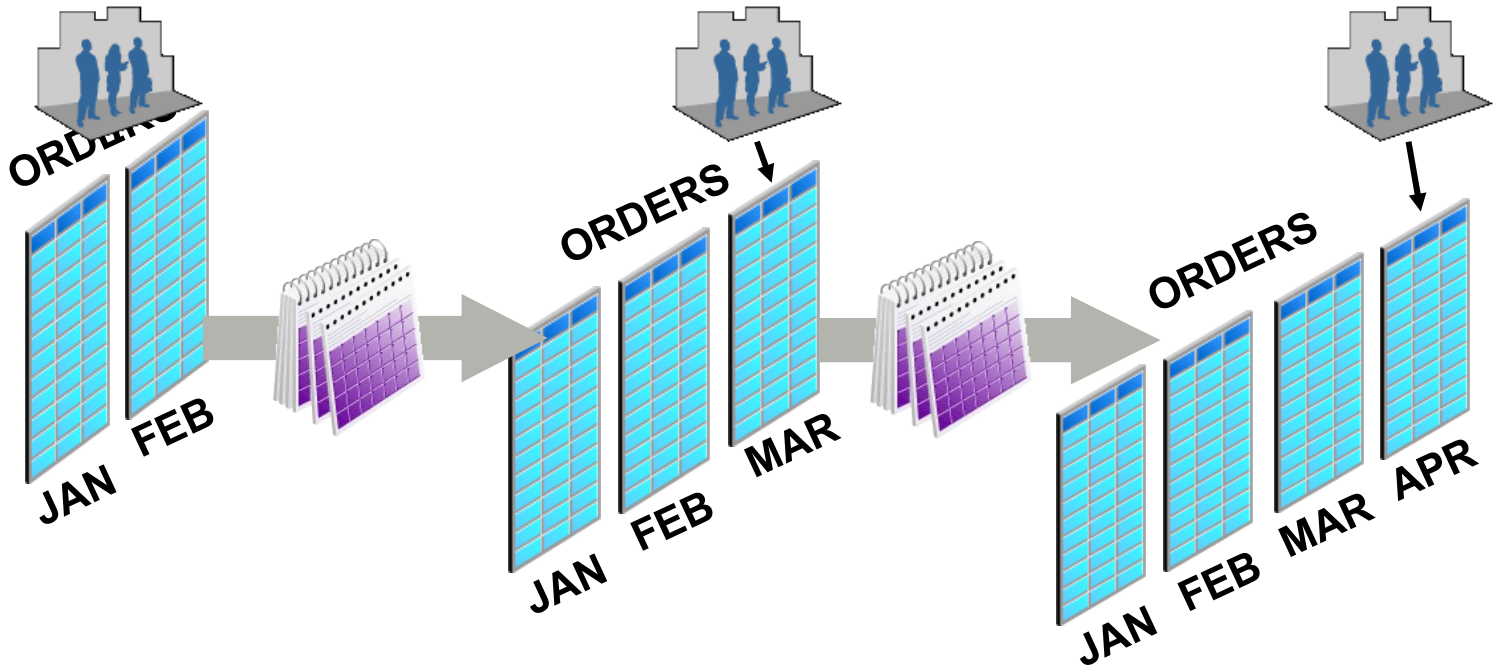
Agenda



- Partitioning in a nutshell
  - “Basics in 5 minutes”
- Partitioning benefits
  - Faster - cheaper
- Partitioning for your business needs
  - Better - new functionality of Oracle Database 11g
- Things you might not know ..
  - Focus on Interval Partitioning versus Range Partitioning
- Q&A

Partitioning in Oracle Database 11g
Interval Partitioning

- Partitions are created automatically as data arrives



Interval Partitioning

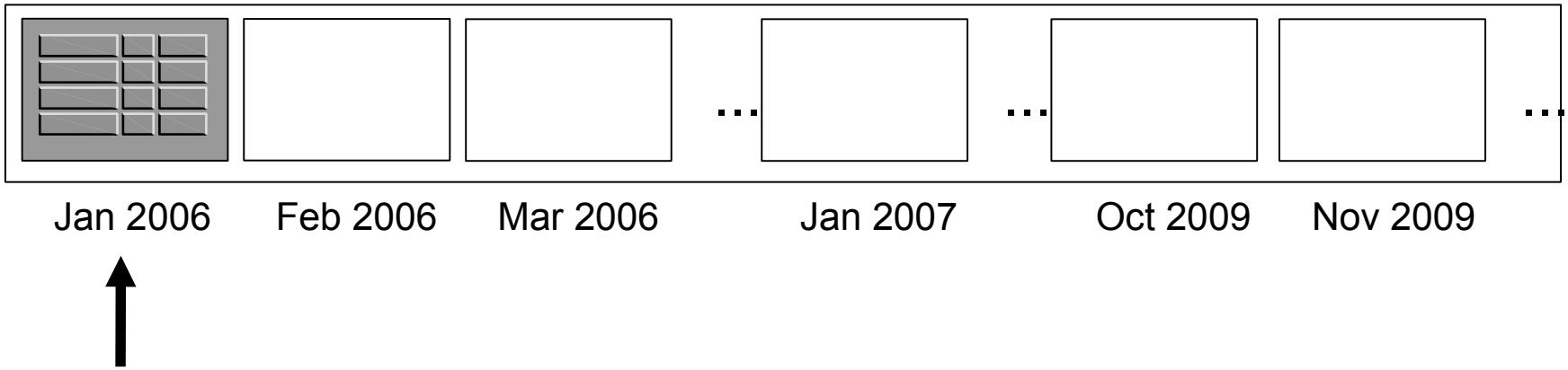
- Interval Partitioning
  - Full automation for equi-sized range partitions
- Partitions are created as metadata information only
  - Start Partition is made persistent
- Segments are allocated as soon as new data arrives
  - No need to create new partitions
  - Local indexes are created and maintained as well
- Interval Partitioning is **almost** a transparent extension to range partitioning
  - .. But interval implementation introduces some subtle differences

Interval Partitioning

- As easy as One, Two, Three ..

```
CREATE TABLE sales (order_date DATE, ...)
PARTITON BY RANGE (order_date)
INTERVAL(NUMTOYMINTERVAL(1,'month')
(PARTITION p_first VALUES LESS THAN ('01-JAN-2006');
```

Table SALES



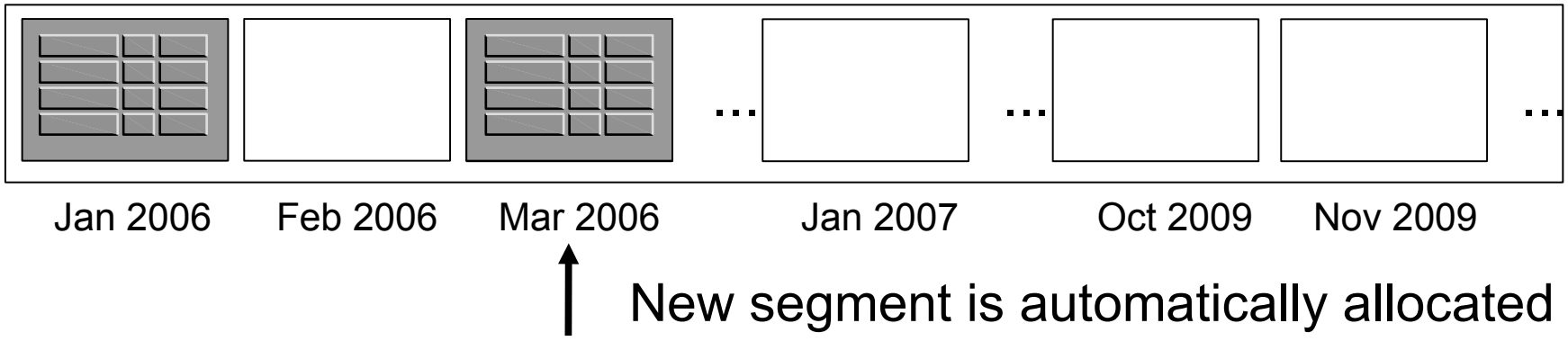
First segment is created

Interval Partitioning

- As easy as One, Two, Three ..

```
CREATE TABLE sales (order_date DATE, ...)
PARTITON BY RANGE (order_date)
INTERVAL(NUMTOYMINTERVAL(1,'month')
(PARTITION p_first VALUES LESS THAN ('01-JAN-2006');
```

Table SALES



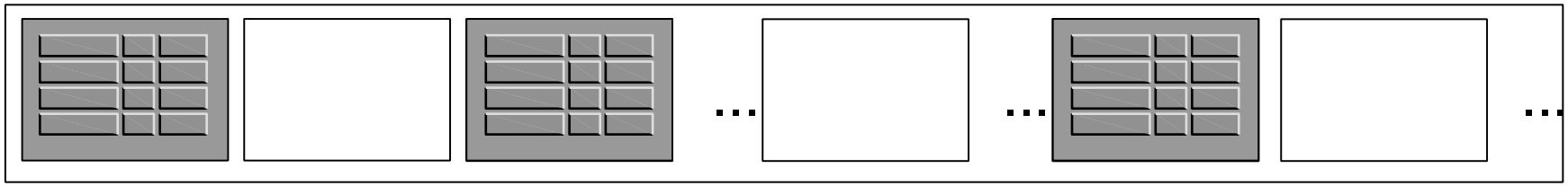

```
INSERT INTO sales (order_date
  DATE, ...)
VALUES ('04-MAR-2006',...);
```

Interval Partitioning

- As easy as One, Two, Three ..

```
CREATE TABLE sales (order_date DATE, ...)
PARTITON BY RANGE (order_date)
INTERVAL(NUMTOYMINTERVAL(1,'month')
(PARTITION p_first VALUES LESS THAN ('01-JAN-2006');
```

Table SALES

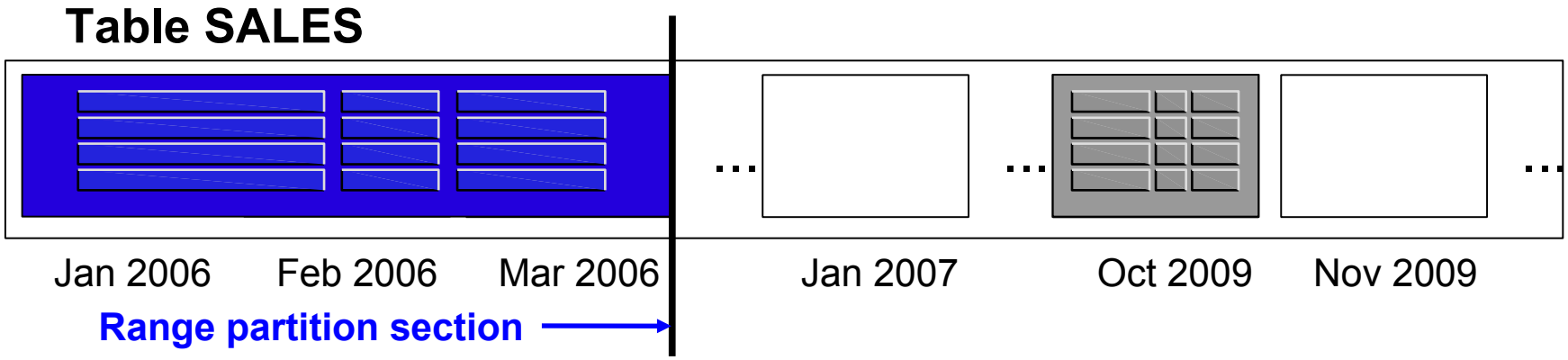


... whenever data for a new partition arrives

```
INSERT INTO sales (order_date
  DATE, ...)
VALUES ('17-OCT-2009',...);
```

Interval Partitioning

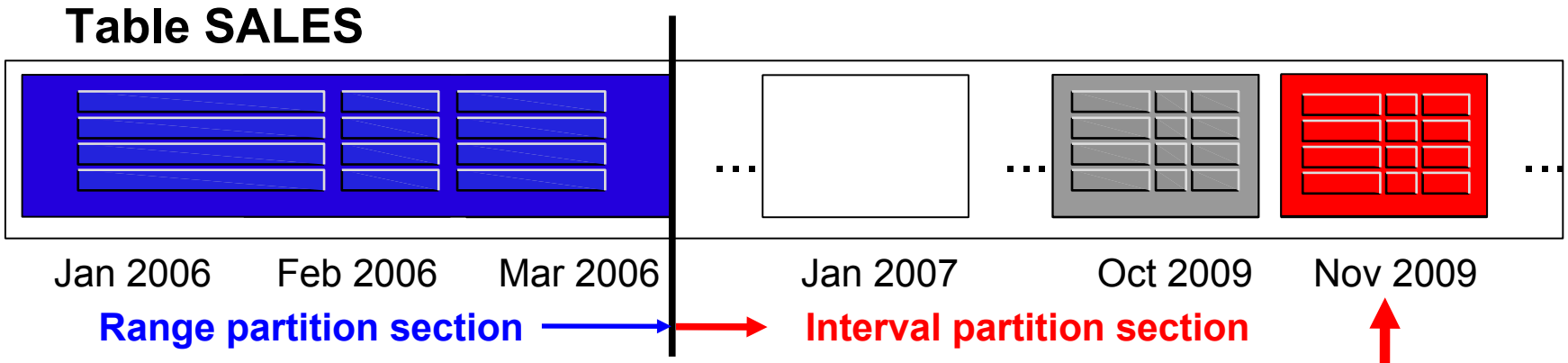
- Interval partitioned table can have classical range and automated interval section
  - Automated new partition management plus full partition maintenance capabilities: ***"Best of both worlds"***



1. MERGE and move old partitions for ILM

Interval Partitioning

- Interval partitioned table can have classical range and automated interval section
  - Automated new partition management plus full partition maintenance capabilities: ***"Best of both worlds"***



1. MERGE and move old partitions for ILM

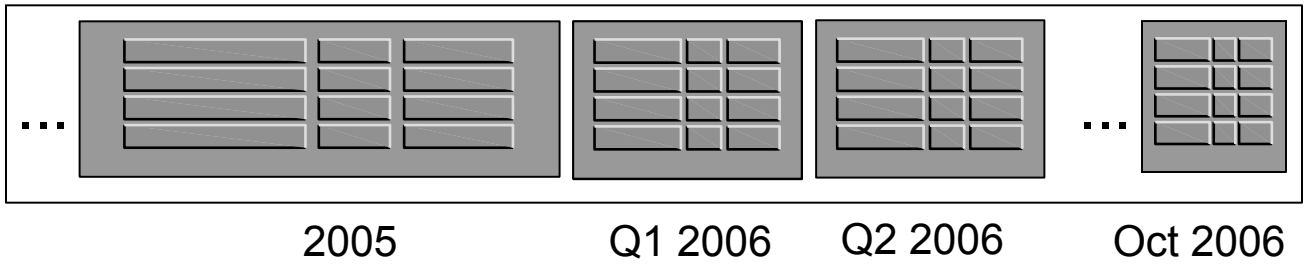
1. Insert new data
   - - Automatic segment creation

VALUES ('13-NOV-2009')

Interval Partitioning

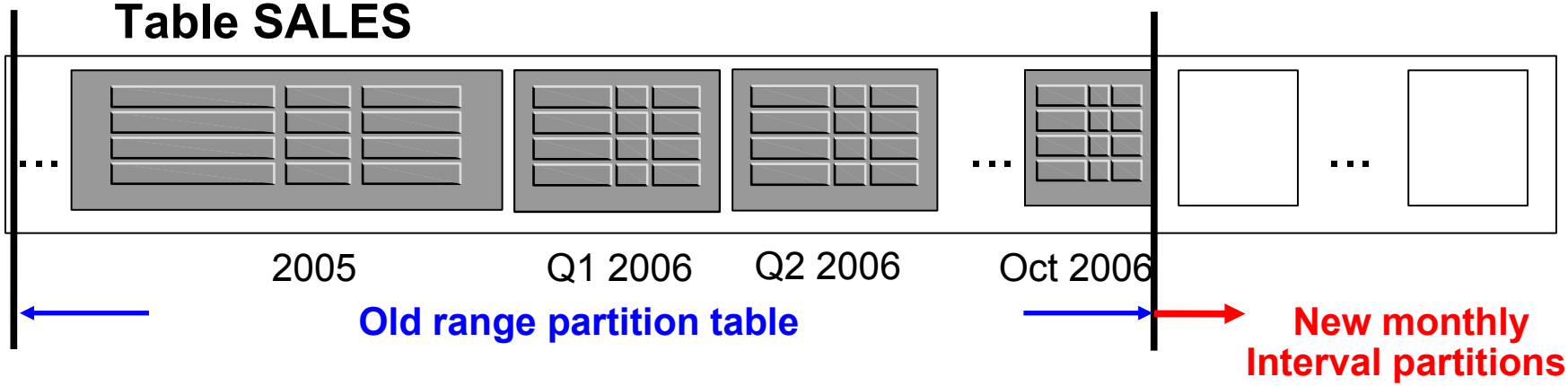
- Range partitioned tables can be extended into interval partitioned tables
  - Simple metadata command
  - Investment protection

Table SALES



Interval Partitioning

- Range partitioned tables can be extended into interval partitioned tables
  - Simple metadata command
  - Investment protection




```
ALTER TABLE sales (order_date DATE, ...)
SET INTERVAL(NUMTOYMINTERVAL(1,'month');
```

Interval Partitioning
Transition Point

How to identify range and interval partitions?

- Internal concept of a 'transition point'
  - Separates range from the interval partitions
  - Starting point of equi-sized partitions
  - Starting point for data placement calculation
- Exposed in data dictionary views, e.g. `*_TAB_PARTITIONS`
  - New column `INTERVAL = [YES|NO]` (11.2)
  - Extract from the dictionary prior to 11.2

```
SELECT subname,
decode(bitand(tp.flags, 32768), 32768, 'YES', 'NO')
FROM tabpart$ tp, obj$ o
WHERE tp.obj#=o.obj# and o.name='TOTO123';

SUBNAME                        DEC
------------------------------ ---
P1                             NO
SYS_P81                        YES
```

Interval versus Range Partitioning

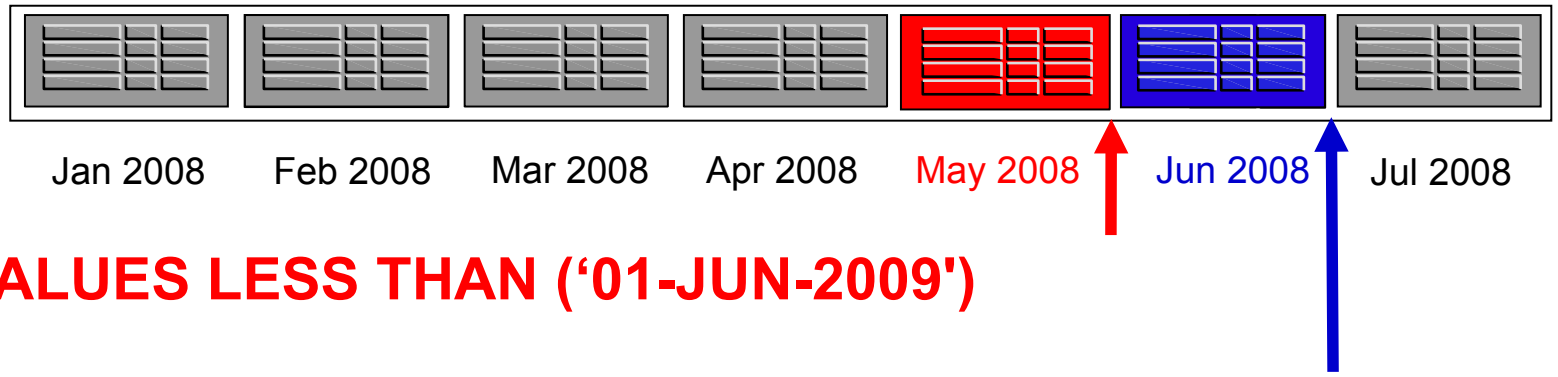
- Partition bounds
  - Interval partitions have lower and upper bound
  - Range partitions only have upper bounds
    - Lower bound derived by previous partition
- Partition naming
  - Interval partitions cannot be named in advance
    - Use the PARTITION FOR (<value>) clause
  - Range partitions must be named

Interval versus Range Partitioning, cont.

- Partition merge
  - Multiple non-existent interval partitions are silently merged
  - Only two adjacent range partitions can be merged at any point in time
- Number of partitions
  - Interval partitioned tables have always one million partitions
    - Non-existent partitions “exist” through INTERVAL clause
    - No MAXVALUES clause for interval partitioning
      - Maximum value defined through number of partitions and INTERVAL clause
  - Range partitioning can have up to one million partitions
    - MAXVALUES clause defines most upper partition

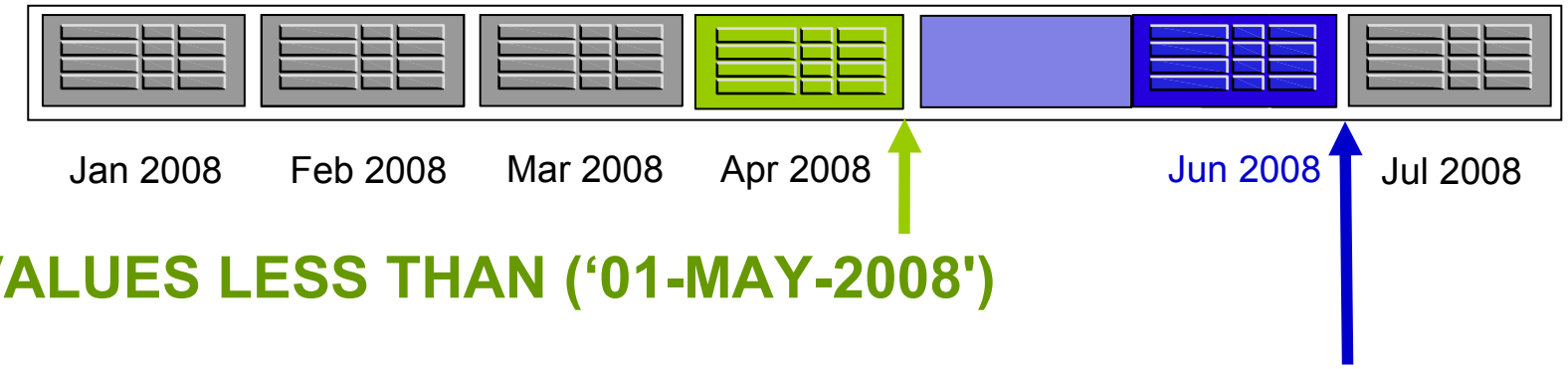
Interval versus Range Partitioning
Partition Bounds

- Partition bounds for range partitioning
  - Partitions only have upper bounds
    - Lower bound derived through upper bound of previous partition



Interval versus Range Partitioning
Partition Bounds

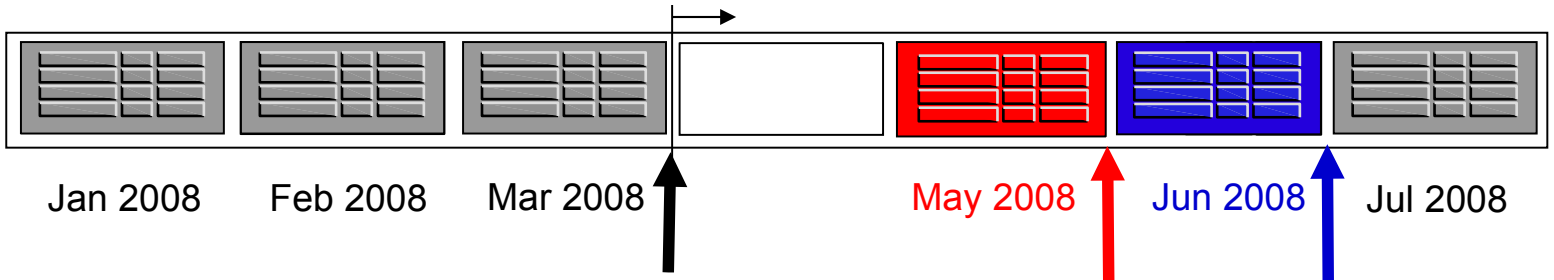
- Partition bounds for range partitioning
  - Partitions only have upper bounds
    - Lower bound derived through upper bound of previous partition



- **Drop of previous partition moves lower boundary**
  - "June 2008" now spawns 01-MAY-2008 to 30-JUN-2008

Interval versus Range Partitioning
Partition Bounds

- Partition bounds for interval partitioning
  - Partitions have upper and lower bounds
    - Derived by INTERVAL function and last range partition



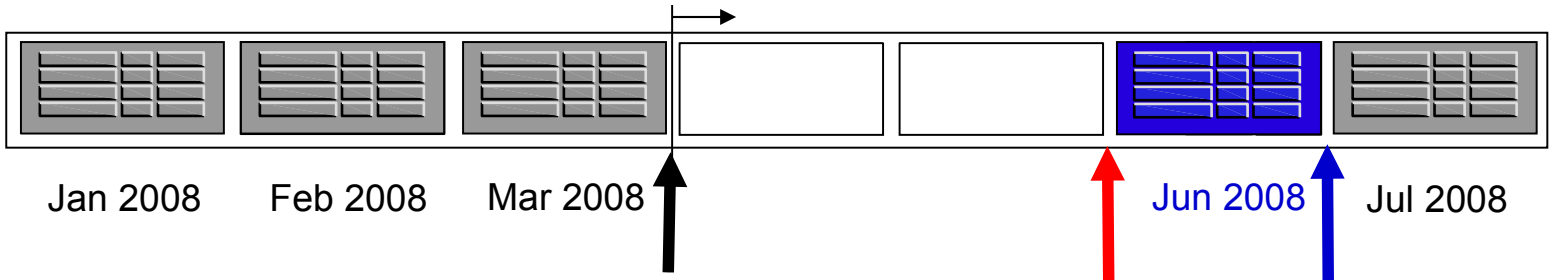
VALUES LESS THAN ('01-APR-2008')

LESS THAN ('01-APR-2008' + 2 x INTERVAL (1 MONTH))

LESS THAN ('01-APR-2008' + 3 x INTERVAL (1 MONTH))

Interval versus Range Partitioning
Partition Bounds

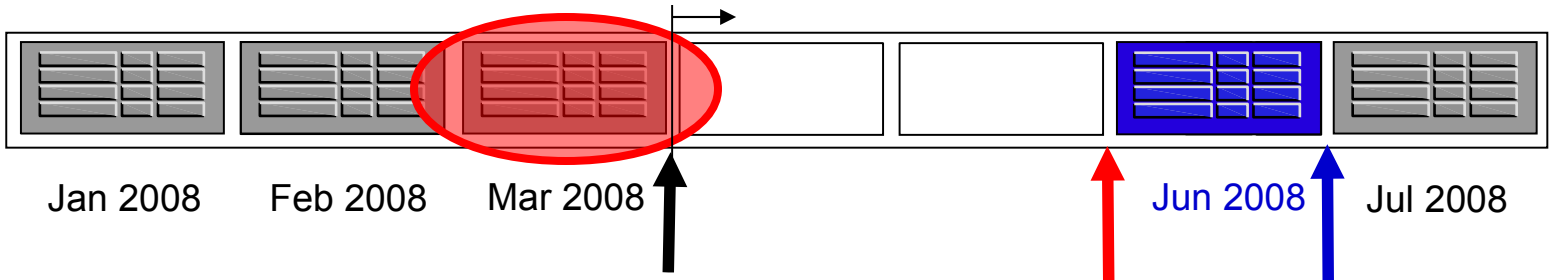
- Partition bounds for interval partitioning
  - Partitions have upper and lower bounds
    - Derived by INTERVAL function and last range partition



- **Drop does not impact partition boundaries**
  - "June 2008" still spawns 01-JUN-2008 to 30-JUN-2008

Interval versus Range Partitioning
Partition Bounds

- Partition bounds for interval partitioning
  - Partitions have upper and lower bounds
    - Derived by INTERVAL function and last range partition



VALUES LESS THAN ('01-APR-2008')

LESS THAN ('01-APR-2008' + 2 x INTERVAL (1 MONTH))

LESS THAN ('01-APR-2008' + 3 x INTERVAL (1 MONTH))

- **There is on untouchable partitions**
  - You cannot remove the highest range partition because it is needed for the interval calculation

Interval versus Range Partitioning
Partition Naming

- Range partitions can be named
  - System generated name if not specified

```
SQL> alter table t add partition values less than(20);
Table altered.
SQL> alter table t add partition P30 values less than(30);
Table altered.
```

- Interval partitions **cannot** be named
  - Always system generated name

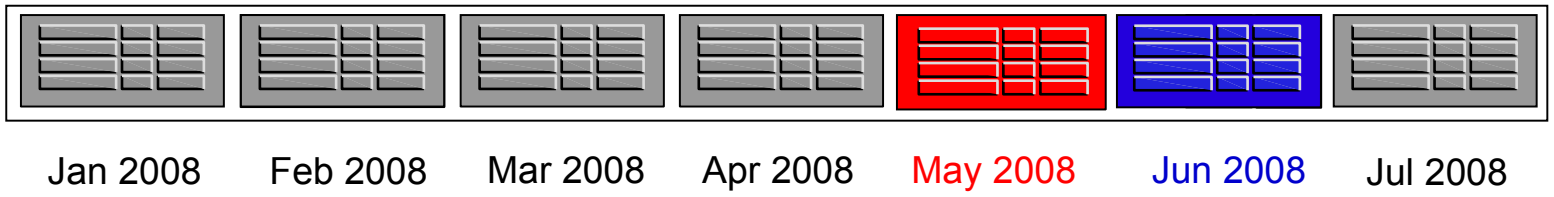
```
SQL> alter table t add partition values less than(20);
                  *
ERROR at line 1: ORA-14760: ADD PARTITION is not permitted on Interval partitioned
objects
```

- Use new deterministic partition extension 'PARTITION FOR (<value>)' clause

```
SQL> alter table t1 rename partition for (9) to p_10;
Table altered.
```

Interval versus Range Partitioning
Partition Merge

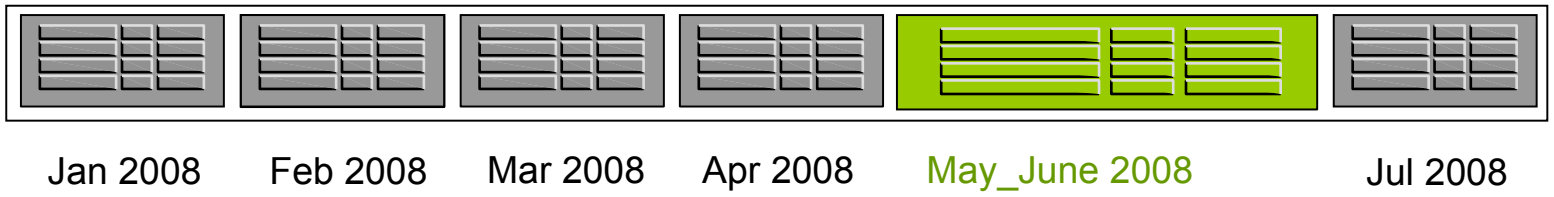
- Merge two adjacent partitions for range partitioning
  - Upper bound of higher partition is new upper bound
  - Lower bound derived through upper bound of previous partition




```
MERGE PARTITIONS MAY_2008, JUN_2008 INTO PARTITION MAY_JUNE_2008
```

Interval versus Range Partitioning
Partition Merge

- Merge two adjacent partitions for range partitioning
  - Upper bound of higher partition is new upper bound
  - Lower bound derived through upper bound of previous partition

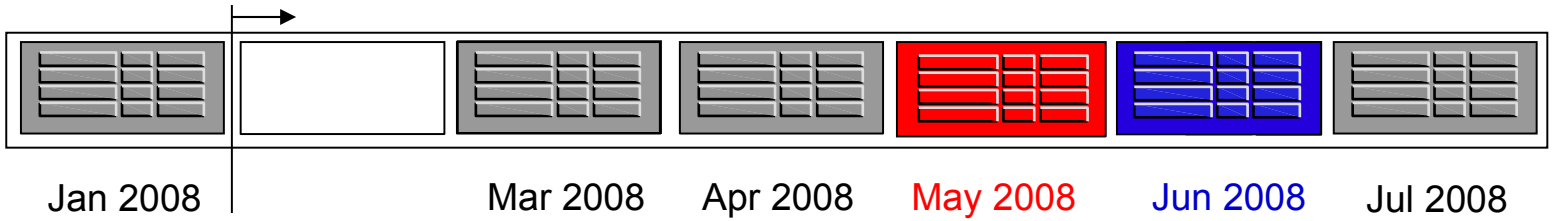



```
MERGE PARTITIONS MAY_2008, JUN_2008 INTO PARTITION MAY_JUNE_2008
```

- New segment for merged partition is created
  - Rest of the table is unaffected

Interval versus Range Partitioning
Partition Merge

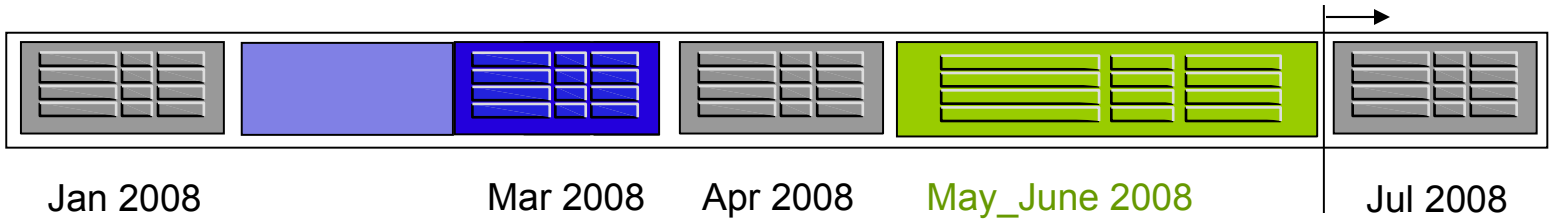
- Merge two adjacent partitions for interval partitioning
  - Upper bound of higher partition is new upper bound
  - Lower bound derived through lower bound of first partition




```
MERGE PARTITIONS MAY_2008, JUN_2008 INTO PARTITION MAY_JUNE_2008
```

Interval versus Range Partitioning
Partition Merge

- Merge two adjacent partitions for interval partitioning
  - Upper bound of higher partition is new upper bound
  - Lower bound derived through lower bound of first partition



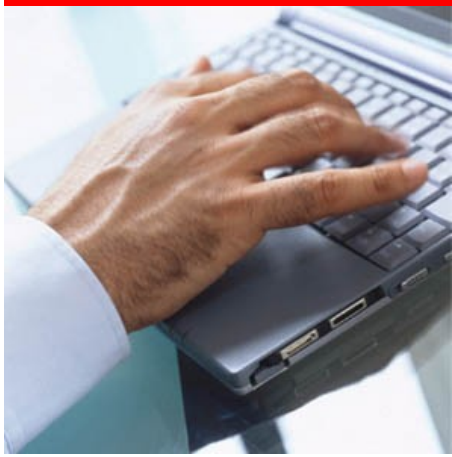

```
MERGE PARTITIONS MAY_2008, JUN_2008 INTO PARTITION MAY_JUNE_2008
```

- New segment for merged partition is created
- Holes before the highest non-interval partition will be silently “merged” as well
  - Interval only valid beyond the highest non-interval partition

Interval Partitioning

- Interval Partitioning
  - Benefit of full automation
  - Should be considered for every range partitioned table
- Segments are allocated as soon as new data arrives
  - No need to create new partitions
  - Local indexes are created and maintained as well
- Interval Partitioning can be considered a **natural** extension of range partitioning
  - Most people will not experience any difference
    - … but be aware of the subtle differences

Summary



- Partitioning in a nutshell
  - Partitioning is simple yet very powerful
- Partitioning benefits
  - Everybody will benefit
- Partitioning for your business needs
  - Model for your needs
- Things you might not know ..
  - Interval Partitioning versus Range Partitioning
- **There is always room for improvements**
  - **Any ideas?? Email hermann.baer@oracle.com**
- Q&A

Q & A

For More Information

search.oracle.com

or

oracle.com


ORACLE®