Partitioning Tables and Indexing Them

Hemant K Chitale

Product Specialist, Standard Chartered Bank

Oracle ACE

© Hemant K Chitale http://hemantoracledba.blogspot.com

`whoami`

- DBA with 20 years experience on wide variety of platforms
- DBA team lead and consultant
- Financial, Manufacturing, Government, not-for-profit
- Mission critical, reporting and “nice-to-have” databases

Introduction

- Very Large Databases
- Growth of Transaction Volumes
- Need to retain Historical Data
- Performance Issues from querying large tables
- Separation of OLTP and DWH/DSS systems
- *Question : Is Partitioning only for Very Large Tables ?*
- What you might not find on Google

Pre-Oracle 8

- Partitioning was introduced in V8
- UNION-ALL joins of distinct tables encapsulated in Views
- You have to control which table a new record goes into
- Check Constraints in V7.3

Elements

- Each Partition has the same Logical Attributes as the Table :
  - Column Names, Column Ordering, Datatypes
  - Constraints
- Partitions are distinct Segments. Therefore, they have distinct Physical Attributes :
  - PCTFREE
  - COMPRESSION
  - Tablespace

Elements – Example

```
SQL> create table SALES_TABLE(sale_date date not null, region varchar2(8), sale_qty
   number)
  2  partition by range (sale_date) subpartition by list (region)
  3  (
  4  partition p_2010 values less than (to_date('01-JAN-2011','DD-MON-YYYY'))
  5    (subpartition p_2010_s_east values ('EAST'),
  6     subpartition p_2010_s_north values ('NORTH'),
  7     subpartition p_2010_s_south values ('SOUTH'),
  8     subpartition p_2010_s_west values ('WEST')
  9    )
 10  ,
 11  partition p_2011 values less than (to_date('01-JAN-2012','DD-MON-YYYY'))
 12    (subpartition p_2011_s_east values ('EAST'),
 13     subpartition p_2011_s_north values ('NORTH'),
 14     subpartition p_2011_s_south values ('SOUTH'),
 15     subpartition p_2011_s_west values ('WEST')
 16    )
 17  )
 18  /

Table created.

SQL>
```

Elements – Example (2)

```
SQL> select object_id, object_name, subobject_name, object_type
  2  from user_objects
  3  order by object_type, object_name, subobject_name
  4  /

 OBJECT_ID OBJECT_NAME          SUBOBJECT_NAME            OBJECT_TYPE
---------- -------------------- ------------------------- -------------------
     54889 SALES_TABLE                                    TABLE
     54890 SALES_TABLE          P_2010                    TABLE PARTITION
     54891 SALES_TABLE          P_2011                    TABLE PARTITION
     54892 SALES_TABLE          P_2010_S_EAST             TABLE SUBPARTITION
     54893 SALES_TABLE          P_2010_S_NORTH            TABLE SUBPARTITION
     54894 SALES_TABLE          P_2010_S_SOUTH            TABLE SUBPARTITION
     54895 SALES_TABLE          P_2010_S_WEST             TABLE SUBPARTITION
     54896 SALES_TABLE          P_2011_S_EAST             TABLE SUBPARTITION
     54897 SALES_TABLE          P_2011_S_NORTH            TABLE SUBPARTITION
     54898 SALES_TABLE          P_2011_S_SOUTH            TABLE SUBPARTITION
     54899 SALES_TABLE          P_2011_S_WEST             TABLE SUBPARTITION

11 rows selected.

SQL>
```

Elements – Example (3)

```
SQL> select segment_name, partition_name, segment_type, tablespace_name
  2  from user_segments
  3  order by segment_name, partition_name
  4  /

SEGMENT_NAME    PARTITION_NAME       SEGMENT_TYPE       TABLESPACE_NAME
--------------- -------------------- ------------------ ----------------
SALES_TABLE     P_2010_S_EAST        TABLE SUBPARTITION USERS
SALES_TABLE     P_2010_S_NORTH       TABLE SUBPARTITION USERS
SALES_TABLE     P_2010_S_SOUTH       TABLE SUBPARTITION USERS
SALES_TABLE     P_2010_S_WEST        TABLE SUBPARTITION USERS
SALES_TABLE     P_2011_S_EAST        TABLE SUBPARTITION USERS
SALES_TABLE     P_2011_S_NORTH       TABLE SUBPARTITION USERS
SALES_TABLE     P_2011_S_SOUTH       TABLE SUBPARTITION USERS
SALES_TABLE     P_2011_S_WEST        TABLE SUBPARTITION USERS

8 rows selected.

SQL>
```

Elements – Example (4)

```
SQL> select partition_name, partition_position, high_value
  2  from user_tab_partitions
  3  where table_name = 'SALES_TABLE'
  4  order by partition_position
  5  /
 PARTITION_NAME       PARTITION_POSITION
-------------------- ------------------
HIGH_VALUE
--------------------------------------------------------------------------------
P_2010                                1
TO_DATE(' 2011-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIA
 P_2011                                2
TO_DATE(' 2012-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIA

SQL>
```

Elements – Example (5)

```
SQL> select partition_name,subpartition_name, subpartition_position, high_value
  2  from user_tab_subpartitions
  3  where table_name = 'SALES_TABLE'
  4  order by partition_name, subpartition_position
  5  /
PARTITION_NAME       SUBPARTITION_NAME    SUBPARTITION_POSITION
-------------------- -------------------- ---------------------
HIGH_VALUE
--------------------------------------------------------------------------------
P_2010               P_2010_S_EAST                            1
'EAST’
P_2010               P_2010_S_NORTH                           2
'NORTH’
P_2010               P_2010_S_SOUTH                           3
'SOUTH’
P_2010               P_2010_S_WEST                            4
'WEST’
P_2011               P_2011_S_EAST                            1
'EAST’
P_2011               P_2011_S_NORTH                           2
'NORTH’
P_2011               P_2011_S_SOUTH                           3
'SOUTH’
P_2011               P_2011_S_WEST                            4
'WEST’

SQL>
```

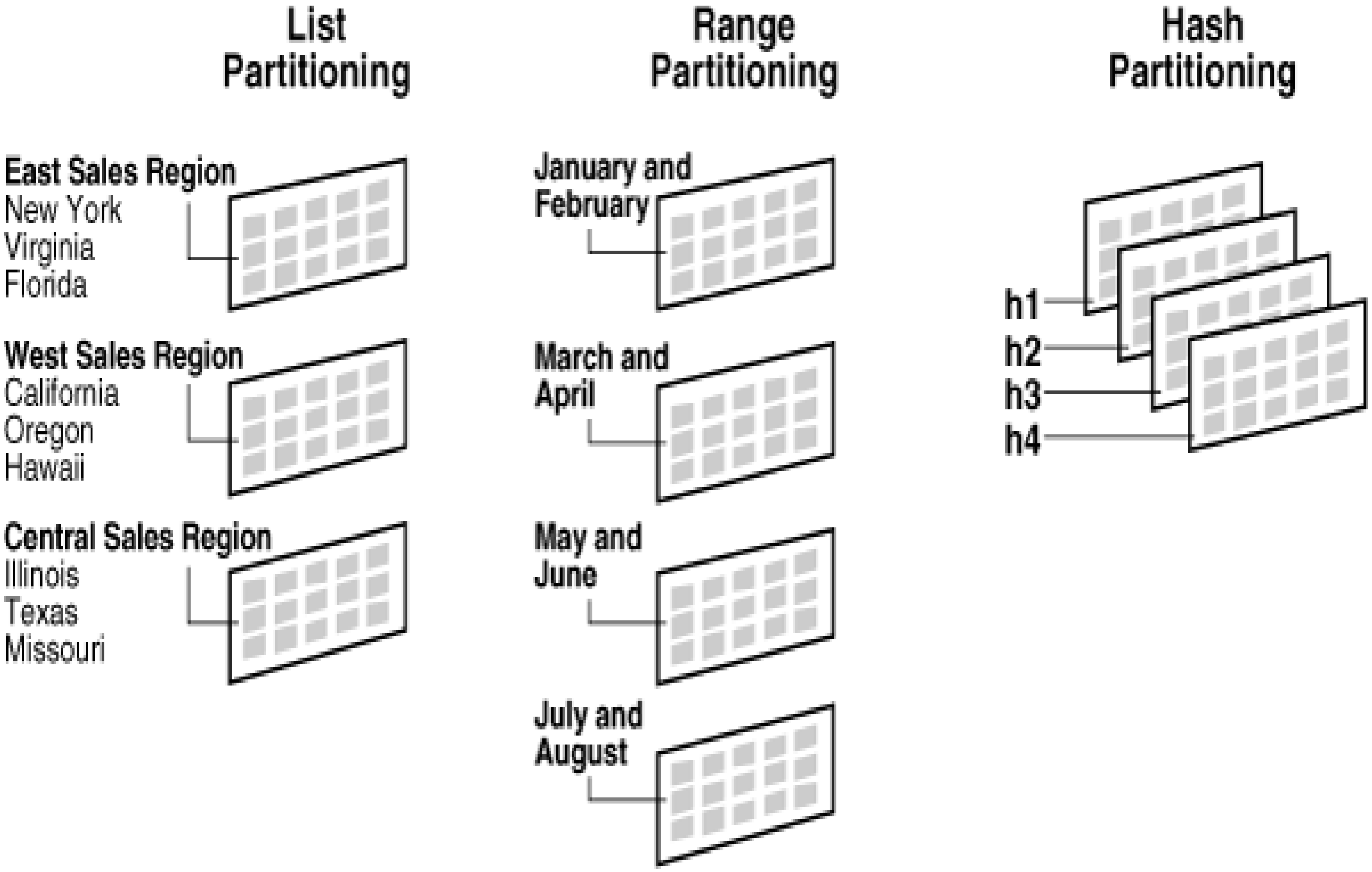
Common Types

- Range Partitioning *(introduced in V8 and still the most popular*)
- Hash Partitioning *(introduced in 8i but much less used than Range and List)*
- List Partitioning *(introduced in 9i)*
- Composite (Range-Hash) (Range-List) (List-Range) (List-Hash) etc Partitioning

Complex Definitions (11g)

- Virtual Column Partitioning
- Reference Partitioning
- Interval Partitioning (as an extension to Range Partitioning)
- *** not in scope ***

Partitioning Types



From the 11gR2 VLDB and Partitioning Guide : Figure 2-2

Range Partitioning

- Most frequently used with Date Ranges

```
partition by range (sale_date)
    (partition p_2010 values less than
                    (to_date('01-JAN-2011','DD-MON-YYYY'))
```

- Useful when you need to be able to Archive/Purge by Date (simply TRUNCATE/DROP the oldest Partition(s))
- Supports multi-column Partition Key
- Each Partition stores values within it’s Upper Bound
- Use a MAXVALUE partition for values above known max
- The Optimizer can use the min and max values for each Partition

Range Partitioning – Examaple (1)

```
SQL> create table ACCOUNTING
  2  (biz_country varchar2(10) not null, acctg_year number not
   null, data_1 varchar2(20))
  3  partition by range (biz_country, acctg_year)
  4  (
  5  partition p_in_2006 values less than ('IN',2007),
  6  partition p_in_2007 values less than ('IN',2008),
  7  partition p_in_2008 values less than ('IN',2009),
  8  partition p_sg_2006 values less than ('SG',2007),
  9  partition p_sg_2007 values less than ('SG',2008),
 10  partition p_sg_2008 values less than ('SG',2009),
 11  partition p_max values less than (MAXVALUE, MAXVALUE)
 12  )
 13  /

Table created.

SQL>
```

Range Partitioning – Example (2)

```
SQL> insert into ACCOUNTING values ('IN',2007,'Row 1');

1 row created.

SQL> insert into ACCOUNTING values ('IN',2008,'Row 2');

1 row created.

SQL> insert into ACCOUNTING values ('JP',2007,'Row 3');

1 row created.

SQL> insert into ACCOUNTING values ('JP',2015,'Row 4');

1 row created.

SQL> insert into ACCOUNTING values ('US',2006,'Row 5');

1 row created.

SQL> insert into ACCOUNTING values ('US',2009,'Row 6');

1 row created.

SQL>
```

Range Partitioning – Example (3)

```
SQL> select * from ACCOUNTING partition (p_in_2006);
no rows selected

SQL> select * from ACCOUNTING partition (p_in_2007);
BIZ_COUNTR ACCTG_YEAR DATA_1
---------- ---------- --------------------
IN               2007 Row 1

SQL> select * from ACCOUNTING partition (p_in_2008);
BIZ_COUNTR ACCTG_YEAR DATA_1
---------- ---------- --------------------
IN               2008 Row 2

SQL> select * from ACCOUNTING partition (p_sg_2006);
BIZ_COUNTR ACCTG_YEAR DATA_1
---------- ---------- --------------------
JP               2007 Row 3
JP               2015 Row 4

SQL> select * from ACCOUNTING partition (p_max);
BIZ_COUNTR ACCTG_YEAR DATA_1
---------- ---------- --------------------
US               2006 Row 5
US               2009 Row 6

SQL>
```

Hash Partitioning

- Define the Partition Key
- Oracle dynamically uses the PK Value to allocate a row to a Partition – you cannot associate them in advance
- A Hashing algorithm is used
- Useful when you have a large number of values but cannot determine allocation
- Define 2^N Partitions else allocation is unbalanced

List Partitioning

- Well-defined list of values for the Partition Key
- Single Column only
- (11g allows List-List composite partitioning)
- Use a DEFAULT Partition for unknown values
- The Optimizer *knows* that every row has the same value for the Partition Key

List Partitioning – Example (1)

- Badly defined Range Partitioning :

```
SQL> create table MONTH_END_BALANCES
  2  (Partition_Key     varchar2(8) not null, account_number number, balance number)
  3  partition by Range (Partition_Key)
  4  (partition P_2011_JAN values less than ('20110132'),
  5  partition P_2011_FEB values less than ('20110229'),
  6  partition P_2011_MAR values less than ('20110332'),
  7  partition P_2011_APR values less than ('20110431'),
  8  partition P_2011_MAY values less than ('20110532'),
  9  partition P_2011_JUN values less than ('20110631'),
 10  partition P_2011_JUL values less than ('20110732'),
 11  partition P_2011_AUG values less than ('20110832'),
 12  partition P_2011_SEP values less than ('20110931'),
 13  partition P_2011_OCT values less than ('20111032'),
 14  partition P_2011_NOV values less than ('20111131'),
 15  partition P_2011_DEC values less than ('20111232')
 16  )
 17  /

Table created.

SQL>
```

List Partitioning – Example (2)

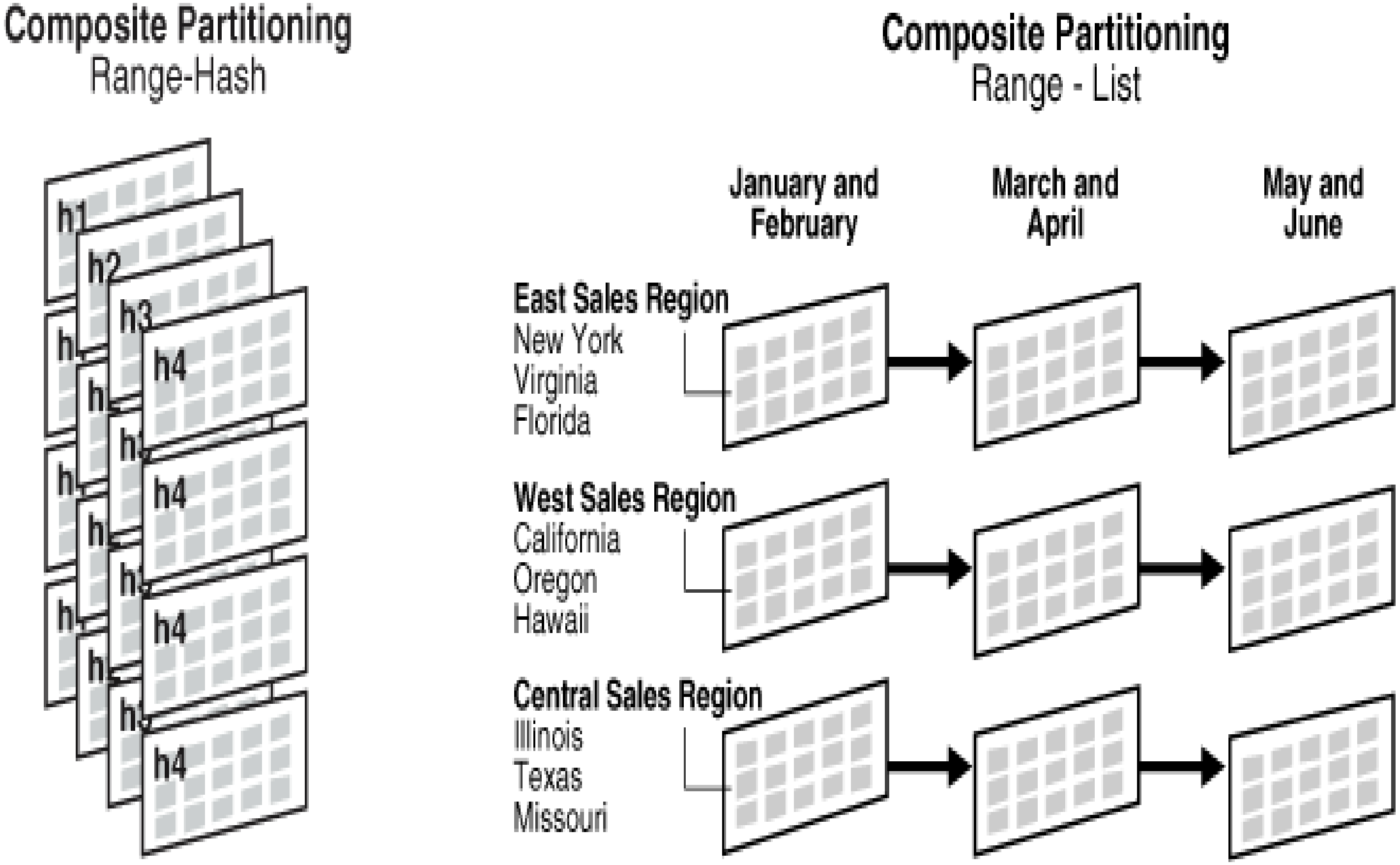
- Better Definition :

```
SQL> create table MONTH_END_BALANCES
  2  (Partition_Key     varchar2(6) not null, account_number number, balance number)
  3  partition by List (Partition_Key)
  4  (partition P_2011_JAN values ('201101'),
  5  partition P_2011_FEB values ('201102'),
  6  partition P_2011_MAR values ('201103'),
  7  partition P_2011_APR values ('201104'),
  8  partition P_2011_MAY values ('201105'),
  9  partition P_2011_JUN values ('201106'),
 10  partition P_2011_JUL values ('201107'),
 11  partition P_2011_AUG values ('201108'),
 12  partition P_2011_SEP values ('201109'),
 13  partition P_2011_OCT values ('201110'),
 14  partition P_2011_NOV values ('201111'),
 15  partition P_2011_DEC values ('201112')
 16  )
 17  /

Table created.

SQL>
```

Composite Partitioning



From the 11gR2 VLDB and Partitioning Guide Figure 2-3

Choosing a Partitioning Method

- Need to Archive / Purge Data : Range
- Querying only for specific range (month in Range Partition, column value in Hash or List)
- Distribution of large data set : Hash
- Discrete, small, set of values : List

Partitioning Method – Examples

- Historical data of Transactions that needs to be purged after 7years : DATE Range
- Employee / Contractor information by State : STATE List
- Statistical Information with a large range of values : VALUE Hash

Adding Data

- You do not need to specify the target Partition name when running an INSERT (serial, parallel, direct).
- If you do name the target Partition, Oracle will still check if the data belongs to the Partition.
- Bulk Insert (Direct Path) into a single (named) Partition will lock only that Partition.
- Use EXCHANGE Partition to switch a non-Partitoned Table with a Partition

Maintaining Partitioned Tables

- Operations : ADD, DROP/TRUNCATE, COALESCE, SPLIT, MERGE, EXCHANGE, DBMS_REDEFINITION, MOVE
- However there are caveats and restrictions
- Maintenance has impact on Indexes as Maintenance Operations are DDLs

ADD Partition

- ADD is to create a new Partition.
- In Range Partitioning you cannot “add” an intermediate partition, you have to SPLIT a Partition (as also if you have a MAXVALUE Partition)
- In Hash Partitioning, adding a new Partition results in Oracle actually splitting an existing Partition – ending up with unbalanced Partitions

DROP or TRUNCATE

- DROP is to drop a Partition
- In Range Partitioning, new rows will go into the “next” Partition
- In List Partitioning, reinserting the dropped values requires a DEFAULT Partition
- TRUNCATE truncates a Partition (TRUNCATE Table truncates all the Partitions) but retains the definition

COALESCE, SPLIT and MERGE

- MERGE allows you to merge adjacent Partitions (use COALESCE in Hash Partitioning to reduce one of the partitions)
- Rows in the “removed” Partition are “moved” into the “new” Partition
- MAXVALUE and DEFAULT Partitions can be SPLITted to create “new” Partitions
- Existing Range Partitions can be SPLITted for granularity

EXCHANGE

- EXCHANGE allows you to interchange a non-Partitioned Table with a Partition
- The empty object is replaced – so be careful as to which of the two is empty !
- EXCHANGE is used to load data from a staging table without another INSERT operation
- EXCHANGE is useful to “move out” an older Partition as a separate Table that can be Exported / Archived

DBMS_REDEFINITION and MOVE

- A Partition can be moved to another Tablespace via DBMS_REDEFINITION
- A Partition can be MOVEd in the same manner as an ALTER TABLE … MOVE
- DBMS_REDEFINITION can also be used “convert” a non-Partitioned Table to a Partitioned Table by copying data across online

Maintaining Indexes

- Types of Indexes
  - Global
  - Global Partitioned
  - Local
- Local Indexes are Equi-Partitioned with the Table
- Global Indexes become UNUSABLE with Table Partition Maintence unless they are UPDATEd

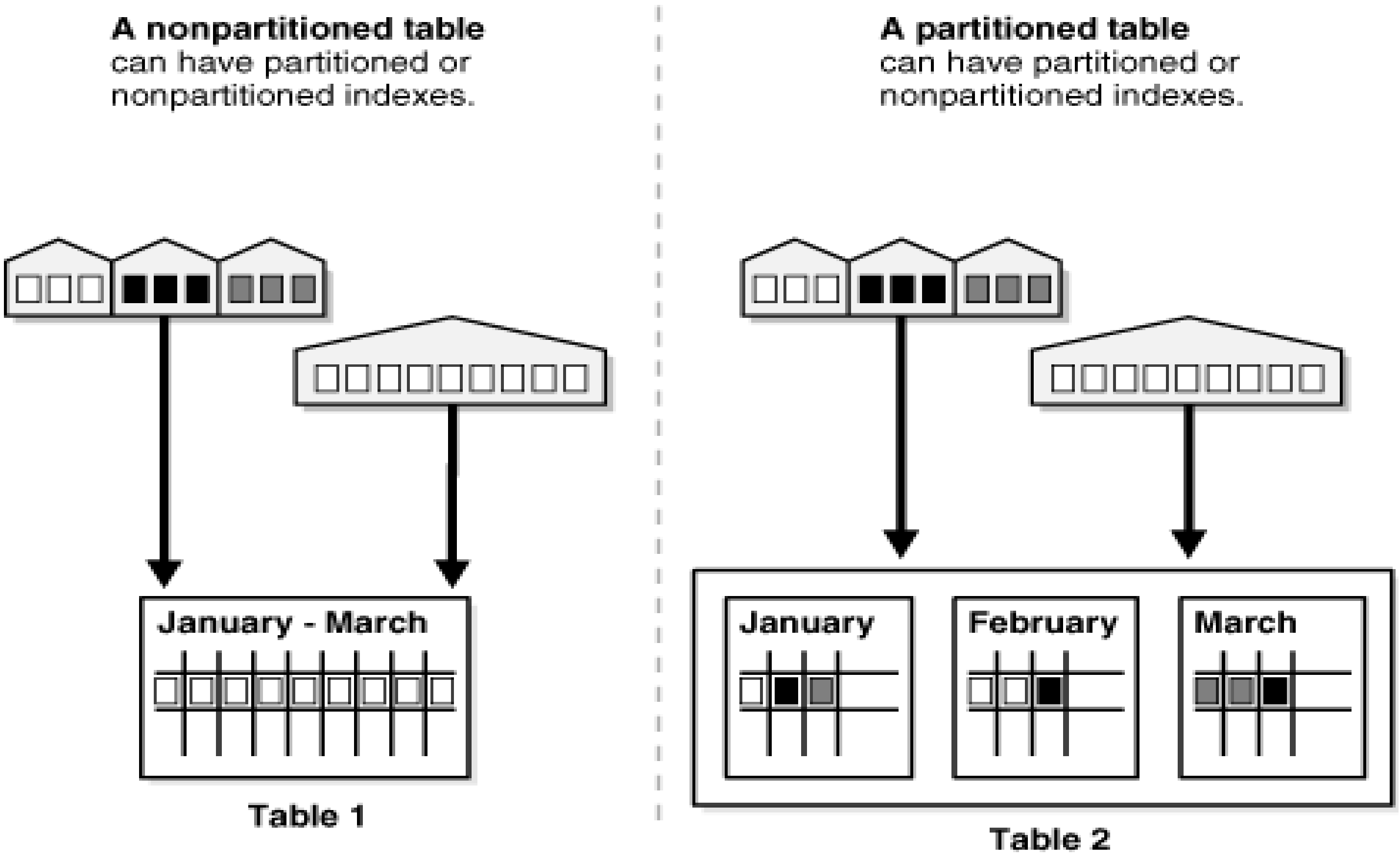
Global Indexes

- Generally recommended in “OLTP” environments --- the real rationale is that queries do not do Partition Pruning
- No different from Local Indexes when retrieving a single row but much more useful when retrieving rows across Partitions
- Global Partitioned Indexes are Partitioned on a separate Partition Key. Useful for queries that are Index-only

Local Indexes

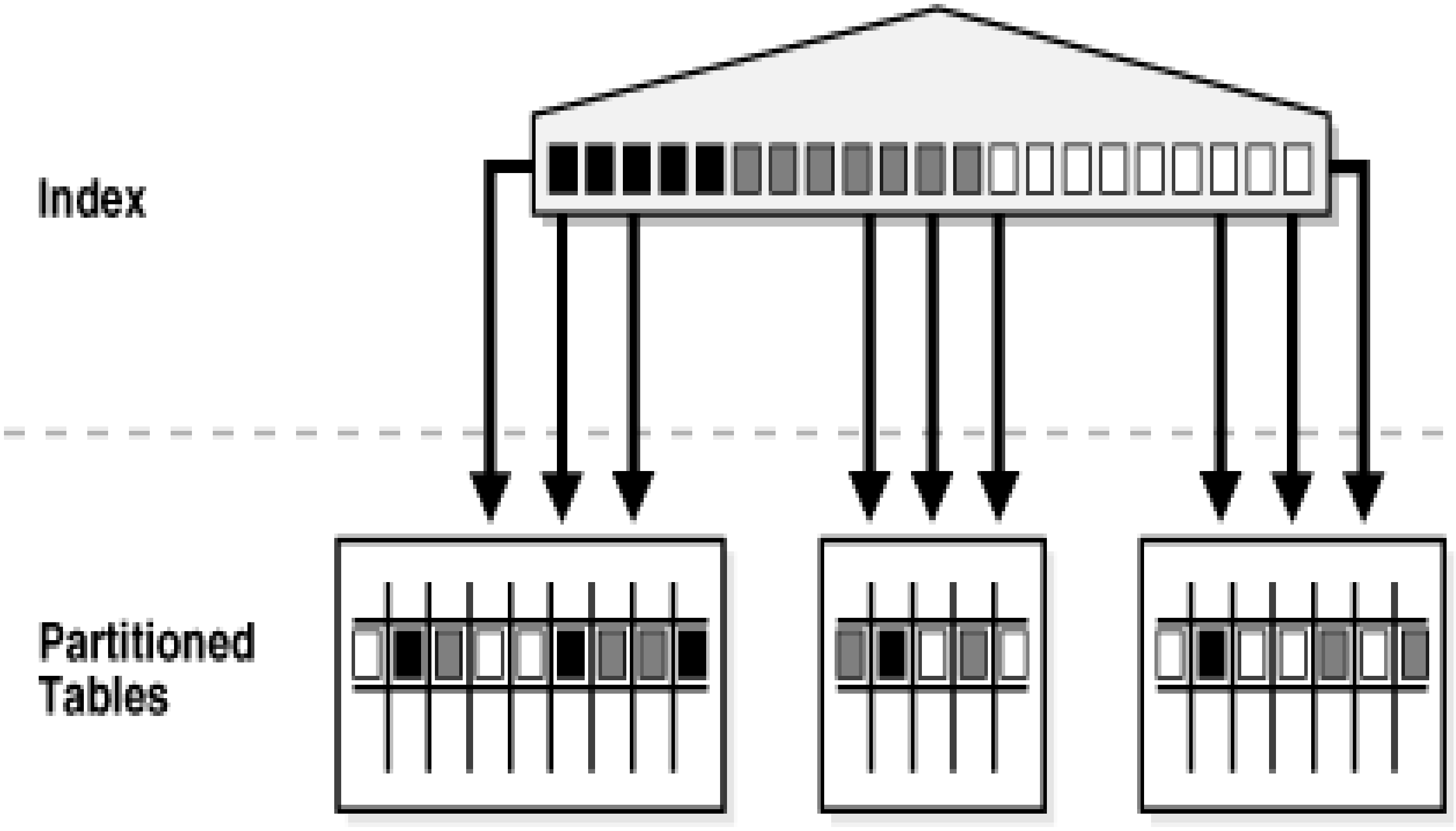
- Generally used for DWH/DSS ; queries that target Partitions (Partition Pruning)
- Must contain Partition Key if defined as UNIQUE
- Most Partition Maintenance operations can also maintain Index Partitions
  - Adding a Table Partition adds Index Partition
  - Dropping a Table Partition drops Index Partition

Partitioned Indexes



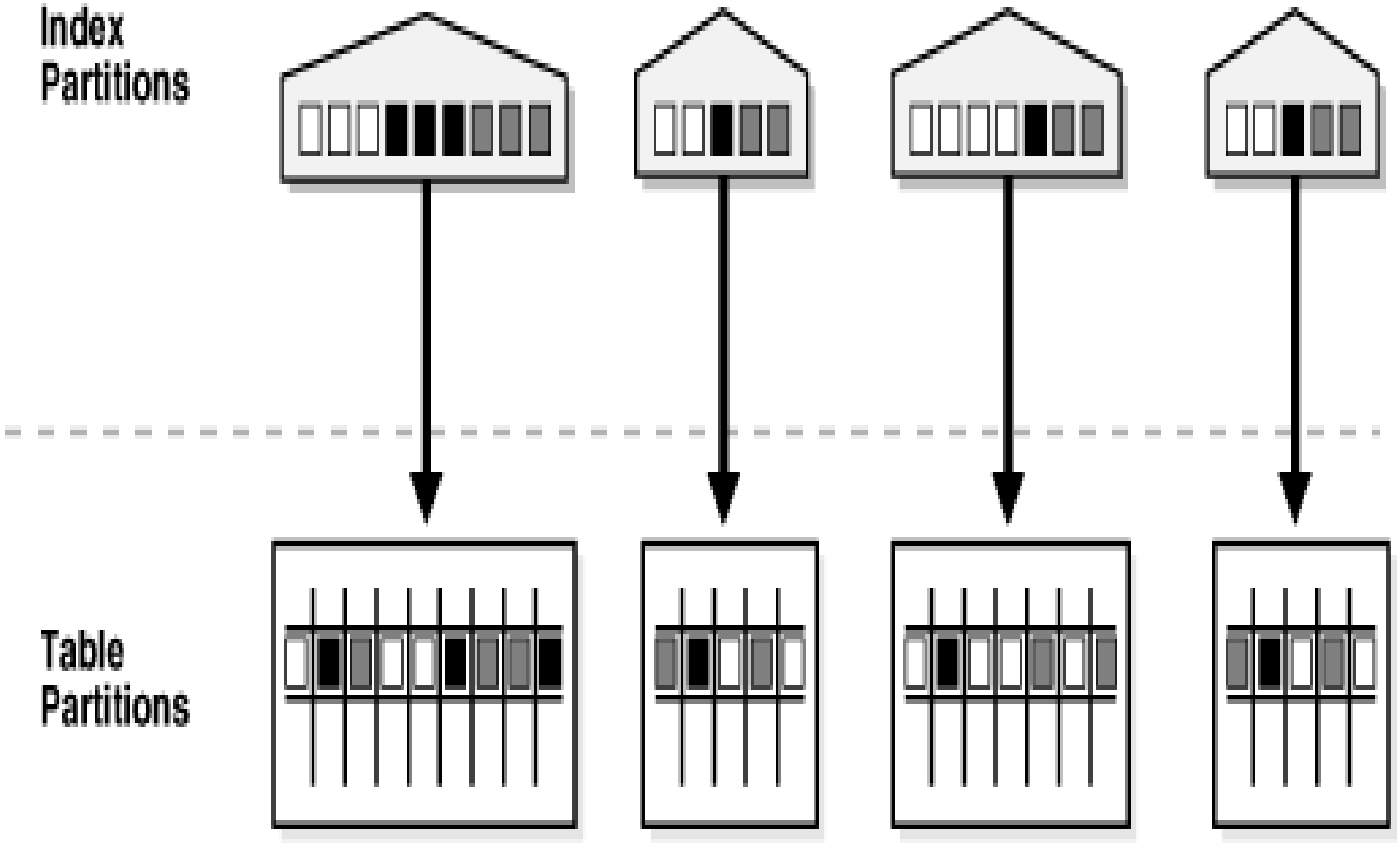
From the 11gR2 VLDB and Partitioning Guide : Figure 2-1

Global Indexes



From the 11gR2 VLDB and Partitioning Guide Figure 2-8

Local Indexes



From the 11gR2 VLDB and Partitioning Guide Figure 2-7

Performance Strategies

- Partition Pruning for subset(s) of Table
- Bulk Insert with Parallel (across Partitions)
- Global Indexes for non-partition Pruning
- Partition-Wise Join with Equi-Partitioning on the same Partition Key
- Hash Partitions for Partition-Wise Joins
- Verify Index definitions when attempting EXCHANGE

Archiving Data

- Data Partitioned on Date key can be Archived/Purged with EXCHANGE and/or DROP
- “Older” Partitions can be moved to slower storage with MOVE
- Partitions can be SPLITted or MERGEd as required

Common Mistakes

- Using the wrong Partitioning Method (and using the wrong datatype for the Partition Key)
- Incorrect Ranges, data going into “other” Partitions (caution : TRUNCATE or DELETE … PARTITION can result in the wrong data being purged !)
- Unequal Partitions (Range or List or Hash)

Common Mistakes - 2

- Updating the Partition Key
- Incorrect LOCAL indexes --- which do not get used or suffer overuse --- where GLOBAL indexes would have helped
- Partition Maintenance with GLOBAL indexes present but not UPDATEing them

Conclusion

- *DON'T* rush into Partition
- Carefully consider :
  - Manner in which data is being or is planned to be inserted
  - Queries
  - Need or plans to purge / archive data
  - Queries
  - Index definitions
  - Queries

Thank you !

- Visit my Oracle Blog
- http://hemantoracledba.blogspot.com