ORACLE

Oracle Partitioning
It's getting even better!

Hermann Bär
Database Product Management

Oracle Partitioning

Over a decade of development

	Core functionality	Performance	Manageability
Oracle 8.0	Range partitioning Global Range indexes	Static partition pruning	Basic maintenance: ADD, DROP, EXCHANGE
Oracle 8*i*	Hash partitioning Range-Hash partitioning	Partition-wise joins Dynamic partition pruning	Expanded maintenance: MERGE
Oracle 9*i*	List partitioning		Global index maintenance
Oracle 9*i* R2	Range-List partitioning	Fast partition SPLIT	
Oracle 10*g*	Global Hash indexes		Local Index maintenance
Oracle 10*g* R2	1M partitions per table	Multi-dimensional pruning	Fast DROP TABLE
Oracle 11g	Virtual column based partitioning More composite choices REF partitioning		- Interval partitioning - Partition Advisor - Incremental stats mgmt
Oracle 11g R2	Hash-Hash partitioning Expanded REF partitioning	"AND" pruning	Multi-branch execution

Make a robust and successful feature even better

- Improved business modeling
- More efficient data maintenance

Improved Business Modeling

Improved Business Modeling

- Align Partitioning with business requirements
- Simplify application development through advanced partition maintenance

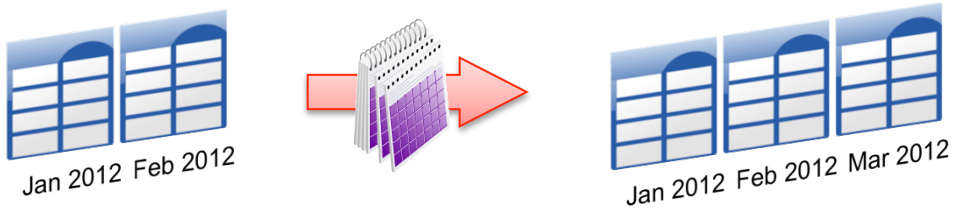
Improved Business Modeling

Interval-Reference Partitioning

New partitions are automatically created when new data arrives

All child tables will be automatically maintained

Combination of two successful partitioning strategies for better business modeling




```
INSERT INTO orders
VALUES ('01-MARCH-2012', ... );
```

Improved Business Modeling

Interval-Reference Partitioning

```
SQL> REM create some interval-referenced tables ..
SQL> create table intRef_p (pkcol number not null, col2 varchar2(200),
  2                         constraint pk_intref primary key (pkcol))
  3  partition by range (pkcol) interval (10)
  4  (partition p1 values less than (10));

Table created.

SQL>
SQL> create table intRef_c1 (pkcol number not null, col2 varchar2(200), fkcol number not null,
  2                          constraint pk_c1 primary key (pkcol),
  3                          constraint fk_c1 foreign key (fkcol) references intRef_p(pkcol) ON DELETE CASCADE)
  4  partition by reference (fk_c1);

Table created.

SQL>
SQL> create table intRef_c2 (pkcol number primary key not null, col2 varchar2(200), fkcol number not null,
  2                          constraint fk_c2 foreign key (fkcol) references intRef_p(pkcol) ON DELETE CASCADE)
  3  partition by reference (fk_c2);

Table created.
```

Improved Business Modeling

Interval-Reference Partitioning

- New partitions only created when data arrives
  - No automatic partition instantiation for complete reference tree
  - Optimized for sparsely populated reference partitioned tables
- Partition names inherited from already existent partitions
  - Name inheritance from direct relative
  - Parent partition p100 will result in child partition p100
  - Parent partition p100 and child partition c100 will result in grandchild partition c100

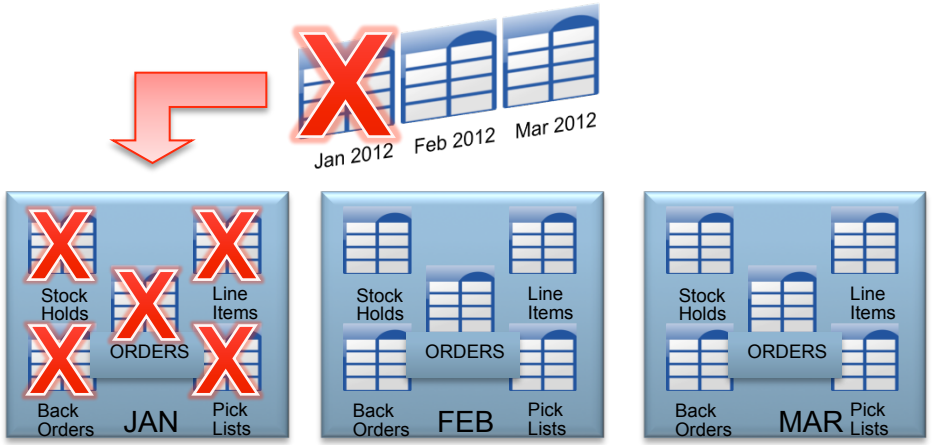
Improved Business Modeling

Cascading TRUNCATE and EXCHANGE PARTITION

Cascading TRUNCATE and EXCHANGE for improved business continuity

Single atomic transaction preserves data integrity

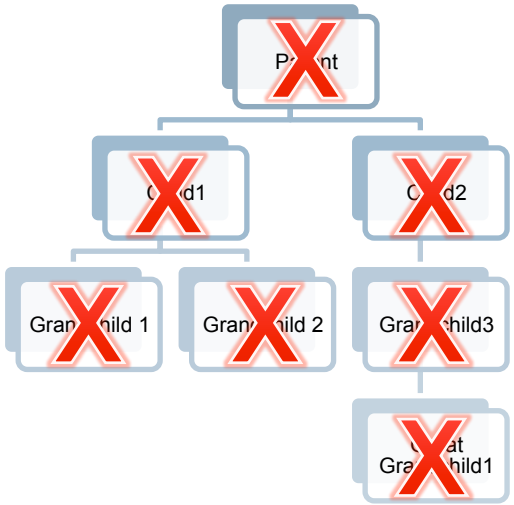
Simplified and less error prone code development



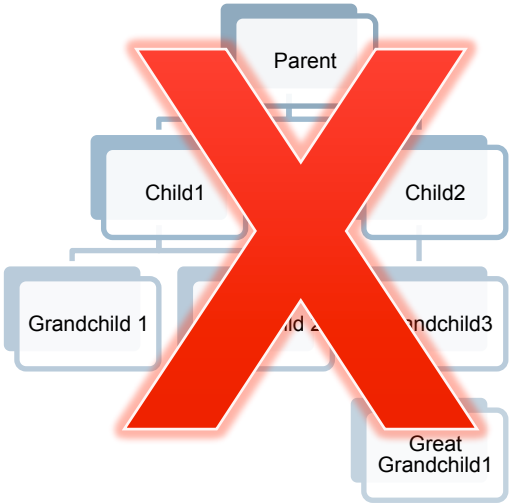

```
ALTER TABLE orders TRUNCATE PARTITION Jan2012 CASCADE;
```

Improved Business Modeling

Cascading TRUNCATE PARTITION



- Proper bottom-up processing required
- Seven individual truncate operations



- One truncate operation

Improved Business Modeling

Cascading TRUNCATE PARTITION

```
SQL> create table intRef_p (pkcol number not nu
  2                         constraint pk_intre
  3  partition by range (pkcol) interval (10)
  4  (partition p1 values less than (10));

Table created.

SQL>
SQL> create table intRef_c1 (pkcol number not n
  2                          constraint pk_c1 p
  3                          constraint fk_c1
  4  partition by reference (fk_c1);

Table created.
```

```
SQL> select * from intRef_p;

     PKCOL COL2
---------- ----------------------------------------
       333  data for truncate - p
       999  data for truncate - p

SQL> select * from intRef_c1;

     PKCOL COL2                                               FKCOL
---------- ---------------------------------------- ----------
      1333  data for truncate - c1                           333
      1999  data for truncate - c1                           999

SQL> alter table intRef_p truncate partition for (999) cascade update indexes;

Table truncated.

SQL> select * from intRef_p;

     PKCOL COL2
---------- ----------------------------------------
       333  data for truncate - p

SQL> select * from intRef_c1;

     PKCOL COL2                                               FKCOL
---------- ---------------------------------------- ----------
      1333  data for truncate - c1                           333
```

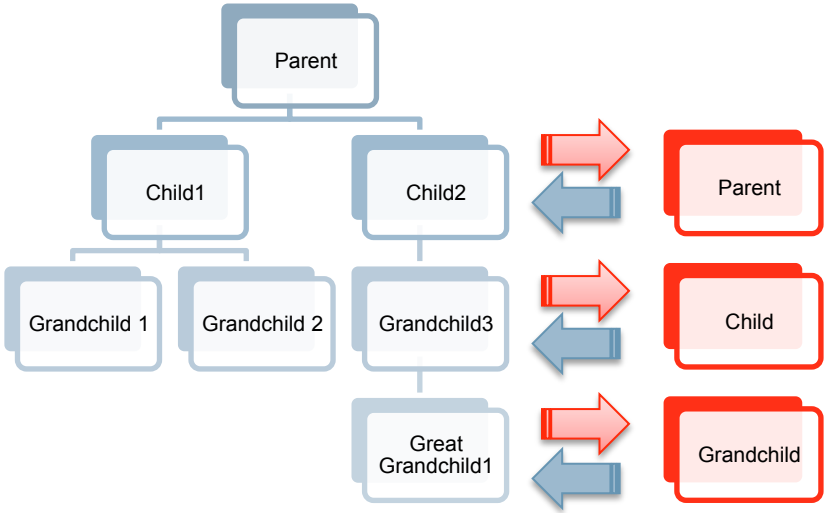
Improved Business Modeling

Cascading TRUNCATE PARTITION

- CASCADE applies for whole reference tree
  - Single atomic transaction, all or nothing
  - Bushy, deep, does not matter
  - Can be specified on any level of a reference-partitioned table
- ON DELETE CASCADE for all foreign keys required
- Cascading TRUNCATE available for non-partitioned tables as well
  - Dependency tree for non-partitioned tables can be interrupted with disabled foreign key constraints

Improved Business Modeling

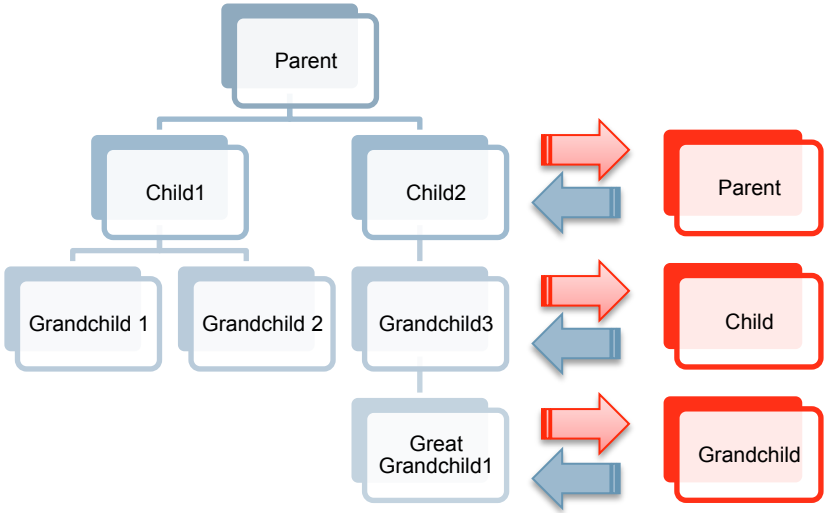
Cascading EXCHANGE PARTITION

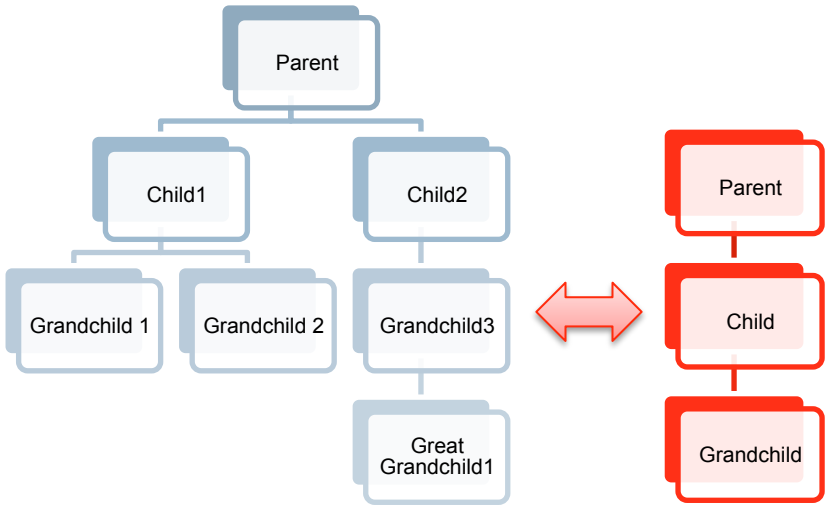


- Exchange (clear) out of target bottom-up
- Exchange (populate) into target top-down

Improved Business Modeling

Cascading EXCHANGE PARTITION





- Exchange (clear) out of target bottom-up
- Exchange (populate) into target top-down

- Exchange complete hierarchy tree
- One exchange operation

Improved Business Modeling

Cascading EXCHANGE PARTITION

```
SQL> select * from intRef_p;

     PKCOL COL2
---------- ------------------------------------
       333  p333 - data BEFORE exchange - p
       999  p999 - data BEFORE exchange - p

SQL> select * from intRef_c1;

     PKCOL COL2                                      FKCOL
---------- ------------------------------------ ----------
      1333  p333 - data BEFORE exchange - c1          333
      1999  p999 - data BEFORE exchange - c1          999

SQL> select * from intRef_gc1;

      COL1 COL2                                      FKCOL
---------- ------------------------------------ ----------
      1333  p333 - data BEFORE exchange - gc1         1333
      1999  p999 - data BEFORE exchange - gc1         1999
```

```
SQL> select * from XintRef_p;

     PKCOL COL2
---------- ------------------------------------
       333  p333 - data AFTER exchange - p

SQL> select * from XintRef_c1;

     PKCOL COL2                                      FKCOL
---------- ------------------------------------ ----------
      1333  p333 - data AFTER exchange - c1           333

SQL> select * from XintRef_gc1;

      COL1 COL2                                      FKCOL
---------- ------------------------------------ ----------
      1333  p333 - data AFTER exchange - gc1          1333
```

Improved Business Modeling

Cascading EXCHANGE PARTITION

```
SQL> select * from intRef_p;

     PKCOL COL2
---------- ------------------------------------
       333  p333 - data AFTER exchange - p
       999  p999 - data BEFORE exchange - p

SQL> select * from intRef_c1;

     PKCOL COL2                                      FKCOL
---------- ------------------------------------ ----------
      1333  p333 - data AFTER exchange - c1            333
      1999  p999 - data BEFORE exchange - c1          999

SQL> select * from intRef_gc1;

      COL1 COL2                                      FKCOL
---------- ------------------------------------ ----------
      1333  p333 - data AFTER exchange - gc1          1333
      1999  p999 - data BEFORE exchange - gc1        1999
```

```
SQL> select * from XintRef_p;

     PKCOL COL2
---------- ------------------------------------
       333  p333 - data BEFORE exchange - p

SQL> select * from XintRef_c1;

     PKCOL COL2                                      FKCOL
---------- ------------------------------------ ----------
      1333  p333 - data BEFORE exchange - c1          333

SQL> select * from XintRef_gc1;

      COL1 COL2                                      FKCOL
---------- ------------------------------------ ----------
      1333  p333 - data BEFORE exchange - gc1        1333
```

Improved Business Modeling

Cascading EXCHANGE PARTITION

- CASCADE applies for whole reference tree
  - Single atomic transaction, all or nothing
  - Bushy, deep, does not matter
  - Can be specified on any level of a reference-partitioned table
- Reference-partitioned hierarchy must match for target and table to-be-exchanged
- For bushy trees with multiple children on the same level, each child on a given level must reference to a different key in the parent table
  - Required to unambiguously pair tables in the hierarchy tree

More Efficient Data Maintenance

More Efficient Data Management

- Size of database systems and individual tables constantly growing
  - Multi-Terabyte single tables common in large enterprise systems
- Maintenance windows are shrinking or even non-existent
  - 24x7 availability requirement
- Requirement: data maintenance operations must
  - Operate transparently without impact on DML and queries
  - Scale with the size of data maintained
  - Touch only relevant data to begin with

More Efficient Data Management

Partitioning Enhancements

- Enhanced Partition Maintenance operations
  - Online partition move
  - Partition maintenance operations on multiple partitions
  - Asynchronous global index maintenance for DROP and TRUNCATE
- Partial global and local indexes

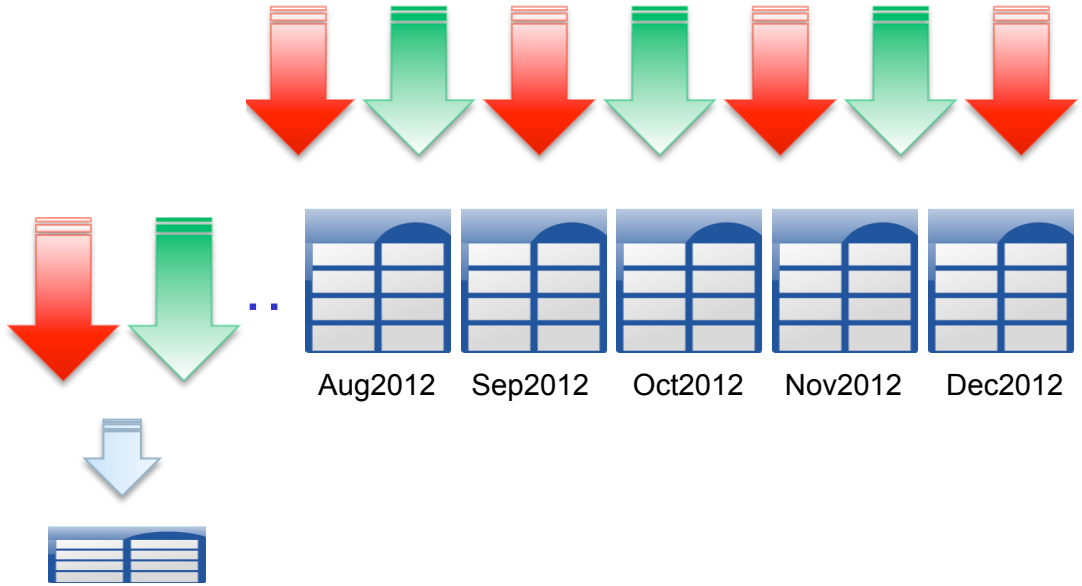
Enhanced Partition Maintenance Operations

Online Partition Move

Transparent MOVE PARTITION ONLINE operation

Concurrent DML and Query

Index maintenance for local and global indexes



Enhanced Partition Maintenance Operations

Online Partition Move – Best Practices

- Minimize concurrent DML operations if possible
  - Require additional disk space and resources for journaling
  - Journal will be applied recursively after initial bulk move
  - The larger the journal, the longer the runtime
- Concurrent DML has impact on compression efficiency
  - Best compression ratio with initial bulk move

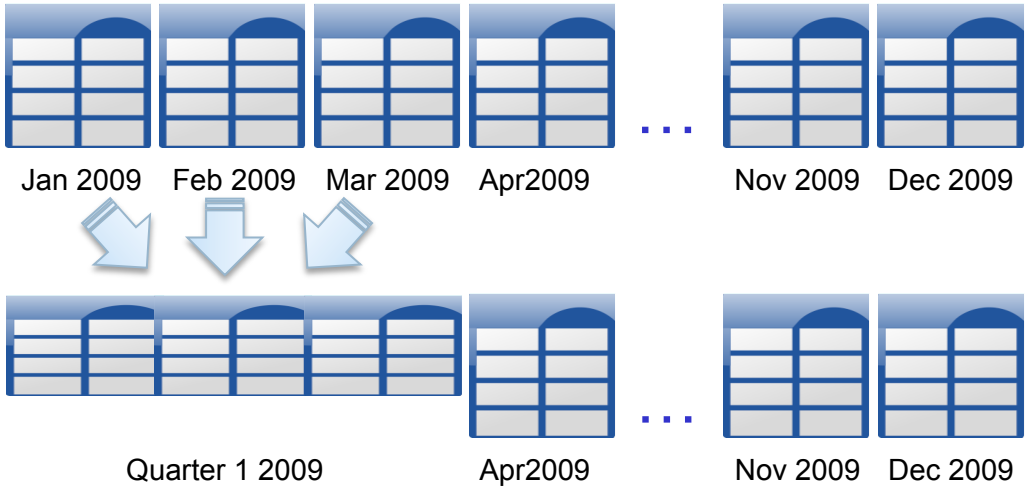
Enhanced Partition Maintenance Operations

Operate on multiple partitions

Partition Maintenance on multiple partitions in a single operation

Full parallelism

Transparent maintenance of local and global indexes




```
ALTER TABLE orders
MERGE PARTITIONS Jan2009, Feb2009, Mar2009
INTO PARTITION Quarter1_2009 COMPRESS FOR ARCHIVE HIGH;
```

Enhanced Partition Maintenance Operations

Operate on multiple partitions

- Specify multiple partitions in order

```
SQL> alter table pt merge partitions for (5), for (15), for (25) into partition p30;

Table altered.
```

- Specify a range of partitions

```
SQL> alter table pt merge partitions part10 to part30 into partition part30;

Table altered.
```

```
SQL> alter table pt split partition p30 into
  2    (partition p10 values less than (10),
  3     partition p20 values less than (20),
  4     partition p30);

Table altered.
```

- Works for all PMOPS
  - Supports optimizations like fast split

Enhanced Partition Maintenance Operations

Asynchronous Global Index Maintenance

- Usable global indexes after DROP and TRUNCATE PARTITION without index maintenance
  - Affected partitions are known internally and filtered out at data access time
- DROP and TRUNCATE become fast, metadata-only operations
  - Significant speedup and reduced initial resource consumption
- Delayed global index maintenance
  - Deferred maintenance through ALTER INDEX REBUILD|COALESCE
  - Automatic cleanup using a scheduled job

Enhanced Partition Maintenance Operations

Asynchronous Global Index Maintenance

- Before

```
SQL> select count(*) from pt partition for (9999);

  COUNT(*)
----------
  25341440

Elapsed: 00:00:01.00
SQL> select index_name, status, orphaned_entries from user_indexes;

INDEX_NAME                     STATUS   ORPHANED_ENTRIES
------------------------------ -------- --------------------
I1_PT                          VALID    NO

Elapsed: 00:00:01.04
SQL>
SQL> alter table pt drop partition for (9999) update indexes;

Table altered.

Elapsed: 00:02:04.52
SQL>
SQL> select index_name, status, orphaned_entries from user_indexes;

INDEX_NAME                     STATUS   ORPHANED_ENTRIES
------------------------------ -------- --------------------
I1_PT                          VALID    NO

Elapsed: 00:00:00.10
```

- After

```
SQL> select count(*) from pt partition for (9999);

  COUNT(*)
----------
  25341440

Elapsed: 00:00:00.98
SQL> select index_name, status, orphaned_entries from user_indexes;

INDEX_NAME                     STATUS   ORPHANED_ENTRIES
------------------------------ -------- --------------------
I1_PT                          VALID    NO

Elapsed: 00:00:00.33
SQL>
SQL> alter table pt drop partition for (9999) update indexes;

Table altered.

Elapsed: 00:00:00.04
SQL>
SQL> select index_name, status, orphaned_entries from user_indexes;

INDEX_NAME                     STATUS   ORPHANED_ENTRIES
------------------------------ -------- --------------------
I1_PT                          VALID    YES

Elapsed: 00:00:00.05
```

Enhanced Indexing with Oracle Partitioning

Indexing prior to Oracle Database 12c

- Local indexes
- Non-partitioned or partitioned global indexes
- Usable or unusable index segments
  - Non-persistent status of index, no relation to table

Enhanced Indexing with Oracle Partitioning

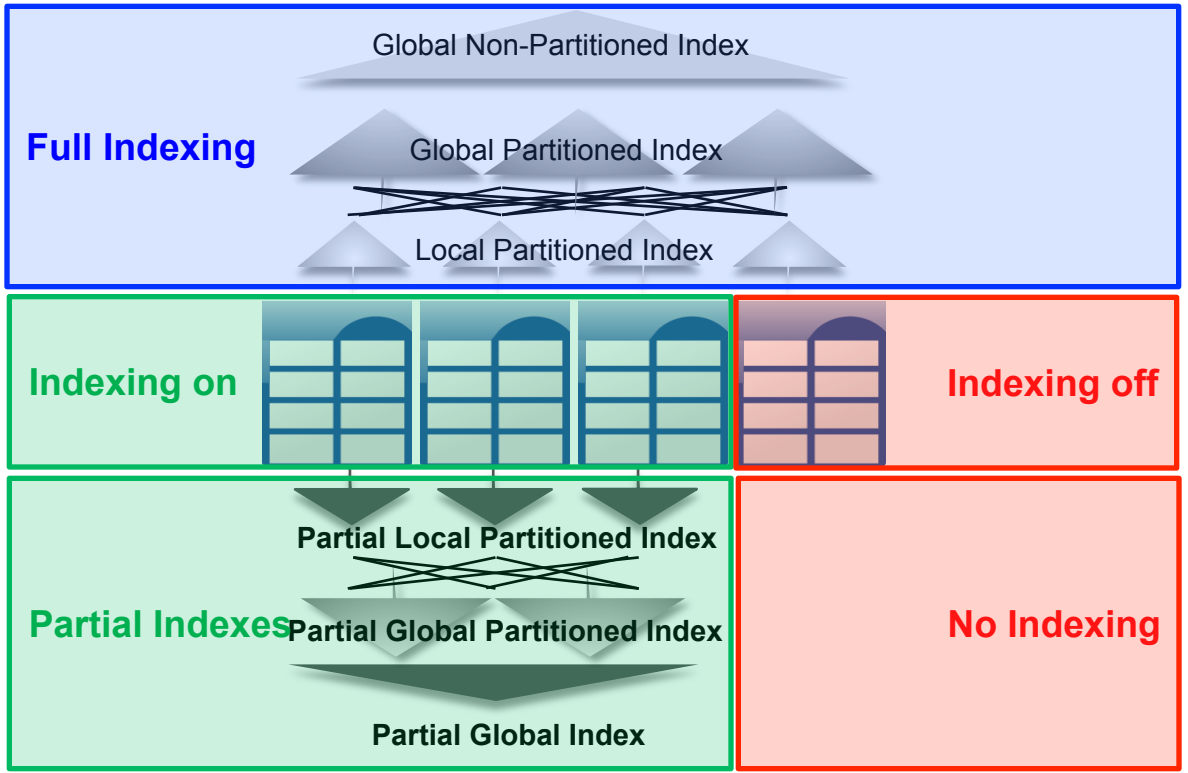
Indexing with Oracle Database 12c

- Local indexes
- Non-partitioned or partitioned global indexes
- Usable or unusable index segments
  - Non-persistent status of index, no relation to table
- Partial local and global indexes
  - Partial indexing introduces table and [sub]partition level metadata
  - Leverages usable/unusable state for local partitioned indexes
  - Policy for partial indexing can be overwritten

Enhanced Indexing with Oracle Partitioning

Partial Local and Global Indexes

- Partial indexes span only some partitions
- Applicable to local and global indexes
- Complementary to full indexing
- Enhanced business modeling



Enhanced Indexing with Oracle Partitioning

Partial Local and Global Indexes

```
SQL> create table pt (col1, col2, col3, col4)
  2  indexing off
  3  partition by range (col1)
  4  interval (1000)
  5  (partition p100 values less than (101) indexing on,
  6   partition p200 values less than (201) indexing on,
  7   partition p300 values less than (301) indexing on);

Table created.

SQL> REM partitions and its indexing status
SQL> select partition_name, high_value, indexing
  2  from user_tab_partitions where table_name='PT';

PARTITION_NAME                 HIGH_VALUE                     INDEXING
------------------------------ ------------------------------ --------
P100                           101                            ON
P200                           201                            ON
P300                           301                            ON
SYS_P1256                      1301                           OFF
```

```
SQL> REM local indexes
SQL> create index i_l_partpt on pt(col1) local indexing partial;
SQL> create index i_l_pt on pt(col4) local;

SQL> REM global indexes
SQL> create index i_g_partpt on pt(col2) indexing partial;
SQL> create index i_g_pt on pt(col3);


SQL> REM index status
SQL> select index_name, partition_name, status, null
  2  from user_ind_partitions where index_name in ('I_L_PARTPT','I_L_PT')
  3  union all
  4  select index_name, indexing, status, orphaned_entries
  5  from user_indexes where index_name in ('I_G_PARTPT','I_G_PT');

INDEX_NAME                     PARTITION_NAME                 STATUS     ORPHAN
------------------------------ ------------------------------ ---------- ------
I_L_PARTPT                     P100                           USABLE
I_L_PARTPT                     P200                           USABLE
I_L_PARTPT                     P300                           USABLE
I_L_PARTPT                     SYS_P1257                      UNUSABLE
I_L_PT                         P200                           USABLE
I_L_PT                         P300                           USABLE
I_L_PT                         SYS_P1258                      USABLE
I_L_PT                         P100                           USABLE
I_G_PT                         FULL                           VALID      NO
I_G_PARTPT                     PARTIAL                        VALID      NO

10 rows selected.
```

Enhanced Indexing with Oracle Partitioning

Partial Local and Global Indexes

- Partial global index excluding partition 4

```
SQL> explain plan for select count(*) from pt where col2 = 3;

Explained.

SQL> select * from table(dbms_xplan.display);

-------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name       | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |            |     1 |    22 |    54  (12)| 00:00:01 |       |       |
|   1 |  SORT AGGREGATE                          |            |     1 |    22 |            |          |       |       |
|   2 |   VIEW                                   | VW_TE_2    |     2 |       |    54  (12)| 00:00:01 |       |       |
|   3 |    UNION-ALL                             |            |       |       |            |          |       |       |
|*  4 |     TABLE ACCESS BY GLOBAL INDEX ROWID BATCHED| PT         |     1 |    26 |     2   (0)| 00:00:01 | ROWID | ROWID |
|*  5 |      INDEX RANGE SCAN                    | I_G_PARTPT |     1 |       |     1   (0)| 00:00:01 |       |       |
|   6 |     PARTITION RANGE SINGLE               |            |     1 |    26 |    52  (12)| 00:00:01 |     4 |     4 |
|*  7 |      TABLE ACCESS FULL                   | PT         |     1 |    26 |    52  (12)| 00:00:01 |     4 |     4 |
-------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - filter("PT"."COL1"<301)
   5 - access("COL2"=3)
   7 - filter("COL2"=3)
```

Oracle Partitioning in Oracle Database 12c

Make a robust and successful feature even better

- Improved business modeling
  - Interval-Reference Partitioning
  - Advanced partition maintenance for Interval-Reference Partitioning
- More efficient data maintenance
  - Enhanced partition maintenance operations
  - Asynchronous global index maintenance
  - Partial indexing

Oracle Partitioning in Oracle Database 12c

Over a decade of development **and better than ever before**

	Core functionality	Performance	Manageability
Oracle 8.0	Range partitioning Global Range indexes	Static partition pruning	Basic maintenance: ADD, DROP, EXCHANGE
Oracle 8*i*	Hash partitioning Range-Hash partitioning	Partition-wise joins Dynamic partition pruning	Expanded maintenance: MERGE
Oracle 9*i*	List partitioning		Global index maintenance
Oracle 9*i* R2	Range-List partitioning	Fast partition SPLIT	
Oracle 10*g*	Global Hash indexes		Local Index maintenance
Oracle 10*g* R2	1M partitions per table	Multi-dimensional pruning	Fast DROP TABLE
Oracle 11g	Virtual column based partitioning More composite choices REF partitioning		- Interval partitioning - Partition Advisor - Incremental stats mgmt
Oracle 11g R2	Hash-Hash partitioning Expanded REF partitioning	"AND" pruning	Multi-branch execution
Oracle 12c R1	Interval-REF partitioning	- Partition Maintenance on multiple partitions - Partial local and global indexes	- Asynchronous global index maintenance for DROP/TRUNCATE - Online partition MOVE - Cascading TRUNCATE/EXCHANGE


ORACLE®