Beginners' Guide
to
Partitioning

Jonathan Lewis
www.jlcomp.demon.co.uk
jonathanlewis.wordpress.com

Who am I ?

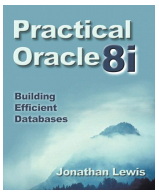
Independent Consultant.

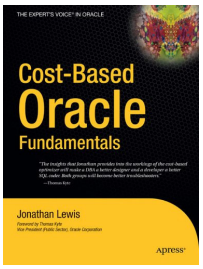
23+ years in IT
20+ using Oracle

Strategy, Design, Review
Briefings, Seminars
Trouble-shooting

www.jlcomp.demon.co.uk
jonathanlewis.wordpress.com

One of the directors of the UKOUG
Member of the Oak Table Network.
Oracle Author of the year 2006
"Select" Editor's choice 2007





Highlights

Partitioning possibilities

Potential Benefits

Possible Problems

Basic Options:

Range - continuous measures

- Typically time-based for aging data

Hash - random distribution

- Typically for reducing contention
- Large number of distinct values
- Powers of 2 for number of partitions

List - explicit distribution

- Short list of interesting values
- Based only on single column
- Histogram on partition key with literal values in queries

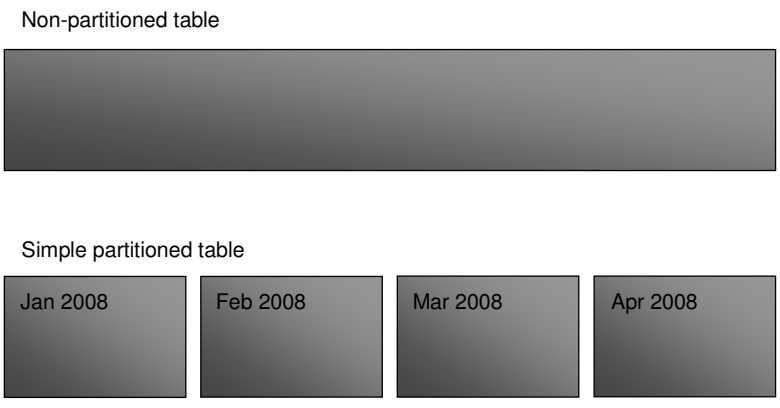
Composite - range / (list or hash) – until 11g

- But CBO doesn't use subpartition stats until 10.2.0.4+

Benefits

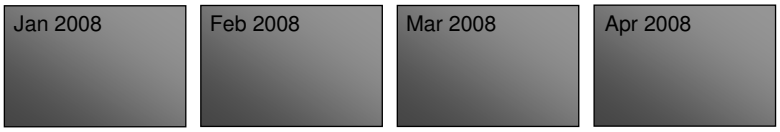
- Partition elimination on queries
  - Effectively “free indexing” effect
- Partition-wise joins
  - Splits one big join into several small joins
- Faster data loading
  - Less contention on concurrent activity
  - Exchange partition tricks for bulk loads
- Dropping old data (ILM – information lifecycle management)
  - Range partition by time

Simple Partitioning

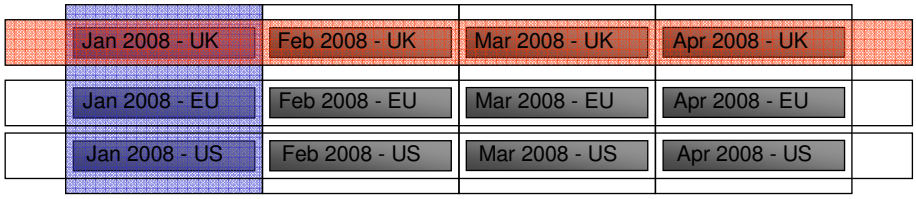


Composite Partitioning

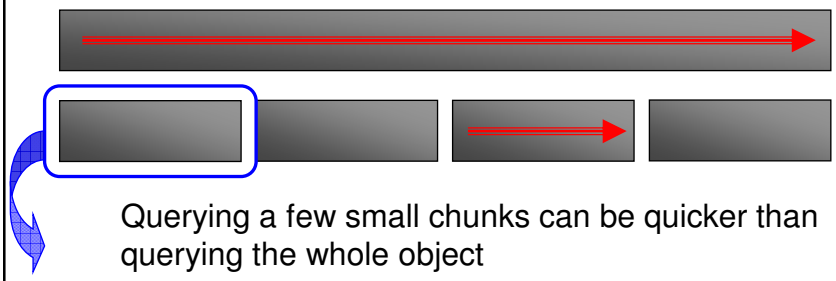
Simple partitioned table



Composite partitioned table



Theoretical Advantages - 1

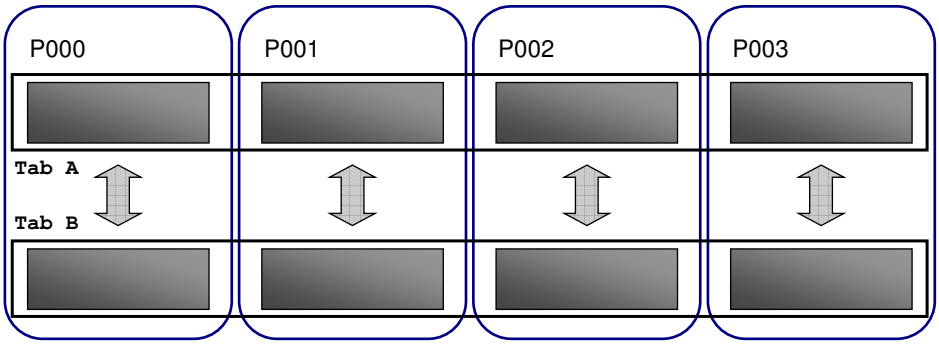


Querying a few small chunks can be quicker than querying the whole object

You can add or drop an entire partition without generating lots of undo and redo.

Rebuild, compress, recycle cache and read-only tablespaces for aging, stable, data.

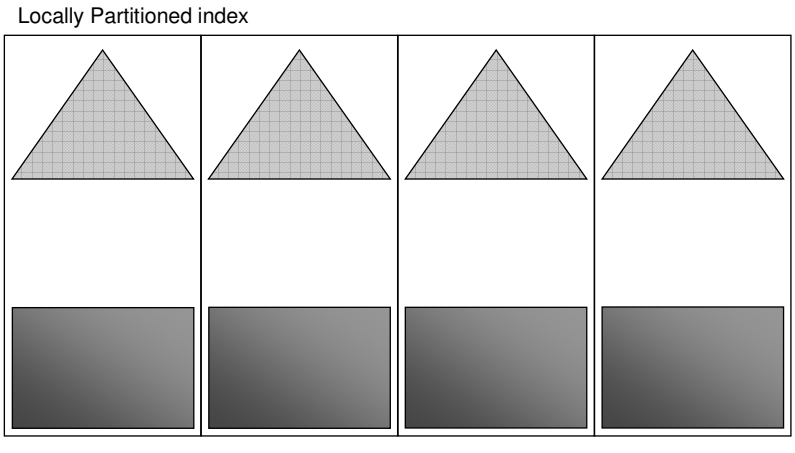
Theoretical Advantages - 2



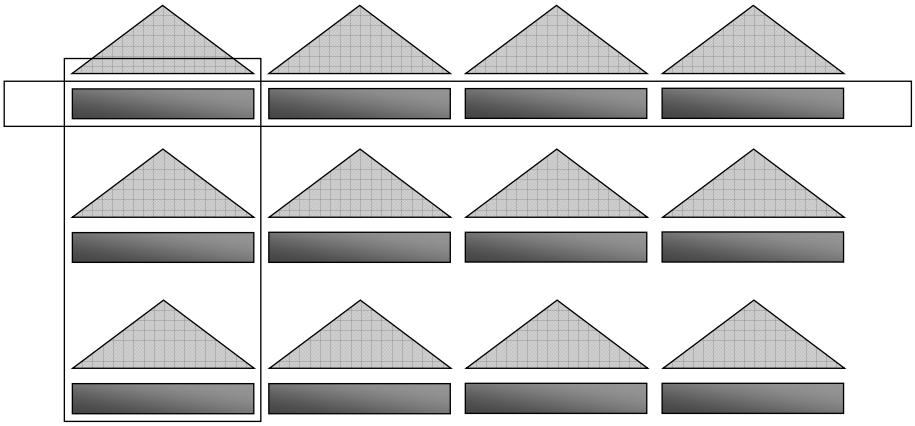
Partition-wise joins:

- Turn one big job into several little ones
- Can minimise inter-process messaging when running parallel

Indexing - 1

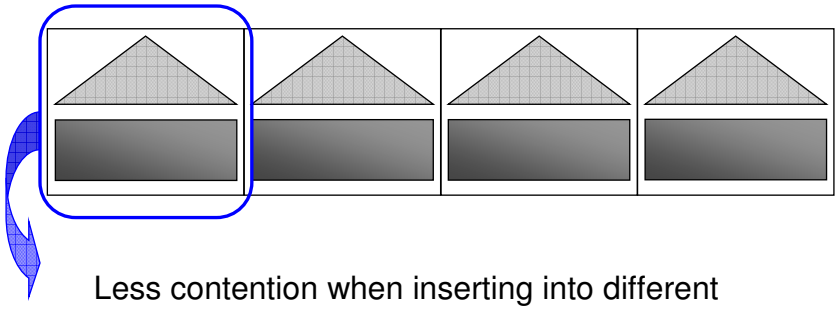


Indexing - 2



Composite partitioned table with local index

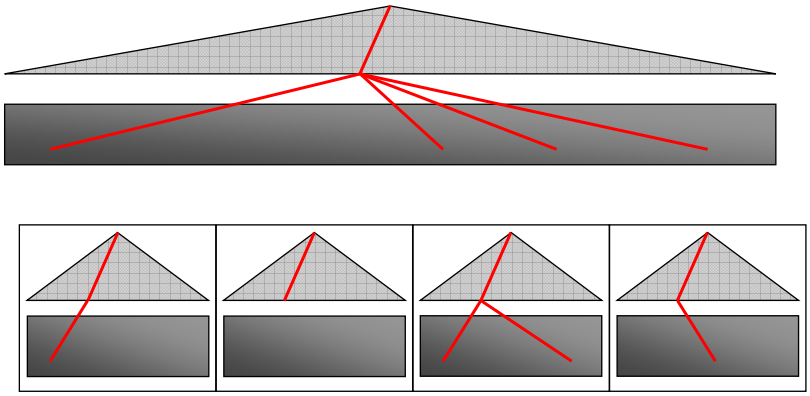
Plus Points



Less contention when inserting into different partitions – especially relevant to indexes, and potentially very useful with hash partitioning

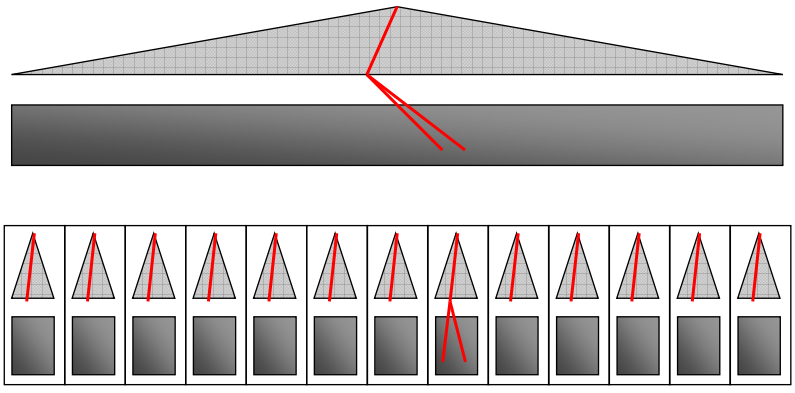
Can still drop a single table partition cheaply – you only take out the matching indexes.

Minus Point – 1a



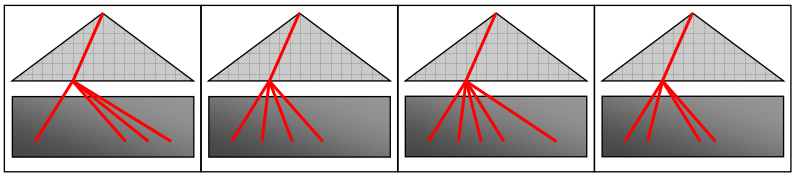
If your queries don't allow partition elimination, the workload and the contention ***get worse***.

Minus Point – 1b



With larger numbers of partitions and a small result set the overhead can be dramatic

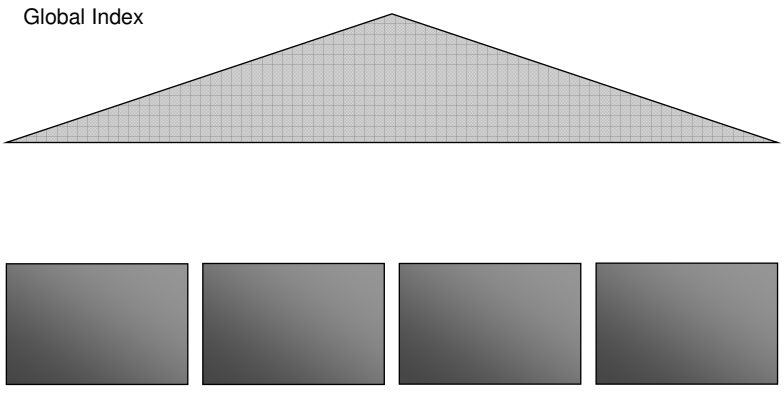
Minus Point - 2



Plans can change:

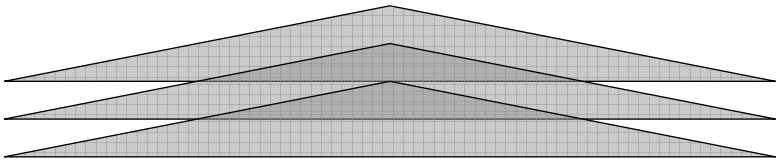
e.g. until this table was hash partitioned the query could use a *"sort order by (nosort)"* operation based on a range scan. Now it has to do a sort.

Indexing - 3



Indexing - 4

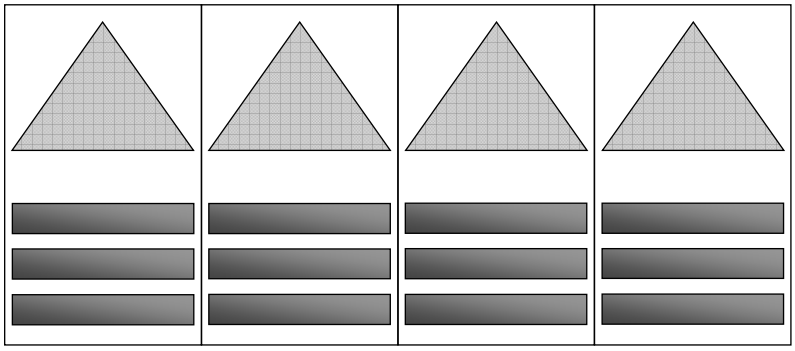
Globally Partitioned index



Simple partitioned table

Indexing - 5

"Semi-local" global index

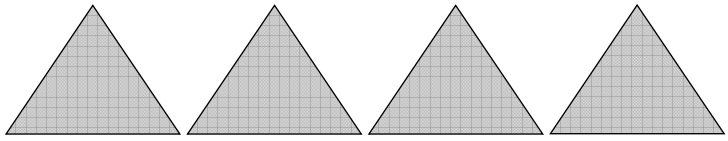


Composite partitioned table

Careful ***manual*** control allows you to index along one dimension of a composite table.

Indexing - 6

Partitioned index



Non-partitioned table

A hash-partitioned index (10g) may be particularly useful for reducing index contention.

Benefits - reprise

Partition elimination on queries

- Effectively “free indexing” effect

Partition-wise joins

- Splits one big join into several small joins

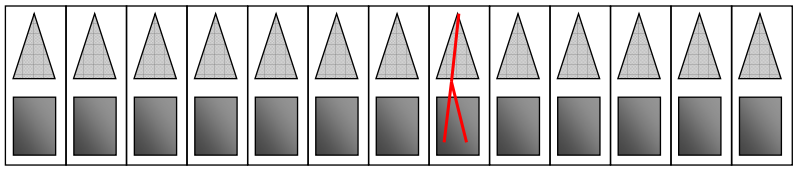
Faster data loading

- Less contention on small concurrent DML
- Exchange partition tricks for bulk loads

Dropping old data (ILM – information lifecycle management)

- Range partition by time

Partition Elimination - 1



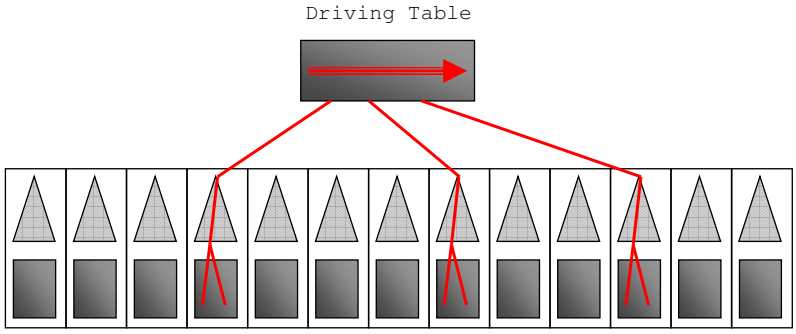
Needs high visibility of predicate in partition key

Does not need prefixed local indexes

Bind values and dates may cause CBO oddities

- be particularly careful to use 4-digit years

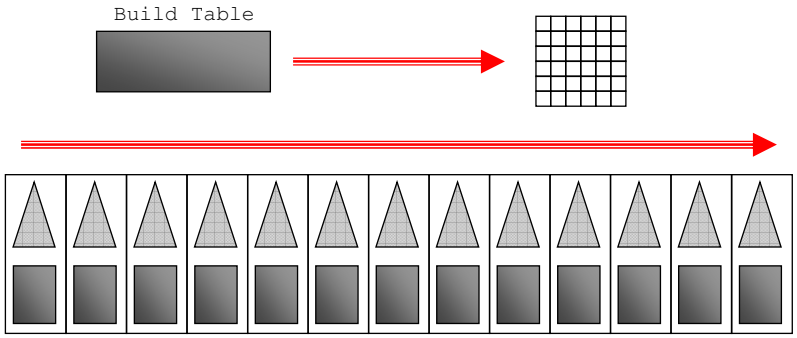
Partition Elimination - 2



Will appear on nested loop join to partition key column.

pstart / pstop in execution plan show as (Key)

Partition Elimination - 3

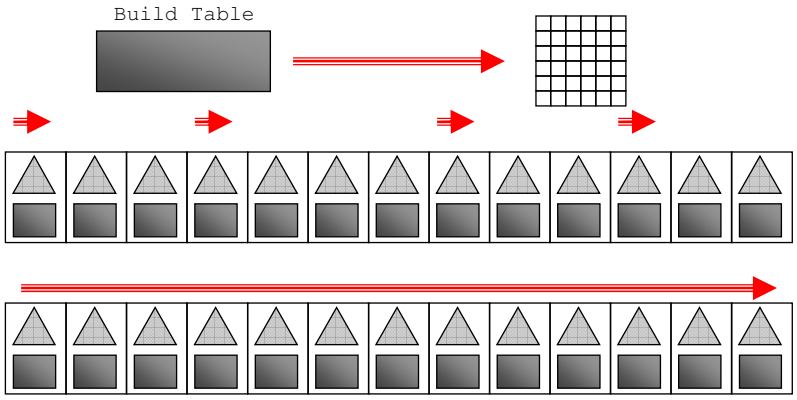


Won't appear for hash joins to partition keys unless ***subquery pruning*** occurs. (The "Bloom filter" appears in 10g)

pstart / pstop displays as Key(SQ) for subquery pruning

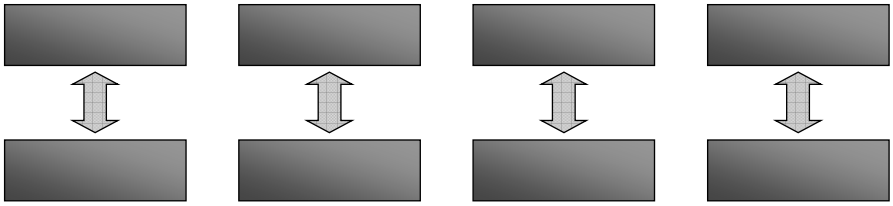
pstart / pstop displays as :BF0000 for bloom filtering

Partition Elimination - 4



If you have two partitioned tables after the driver, it may be impossible to eliminate on the second partitioned table unless you get ***Bloom filters*** working (may be 11g only)

Partition-wise joins - 1



Joining on the partition key.

Needs exact matches on the partitions

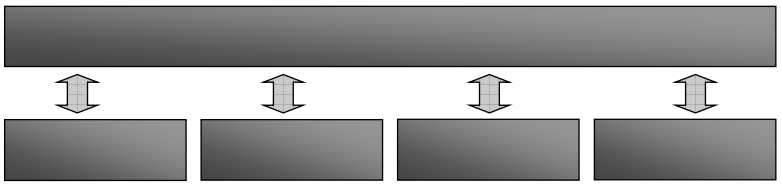
same number, same high values

Be careful with list partitioning ('CA','TX') != ('TX','CA')

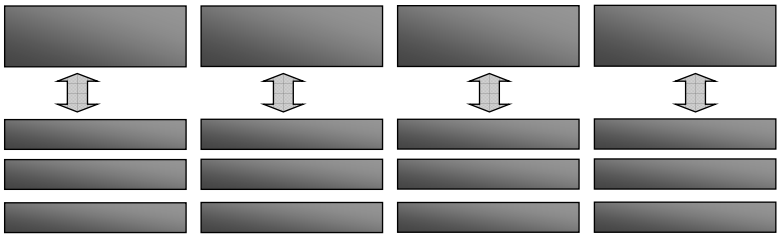
Can be "broken" by high degrees of parallelism.

Oracle may do "broadcast" distribution in error.

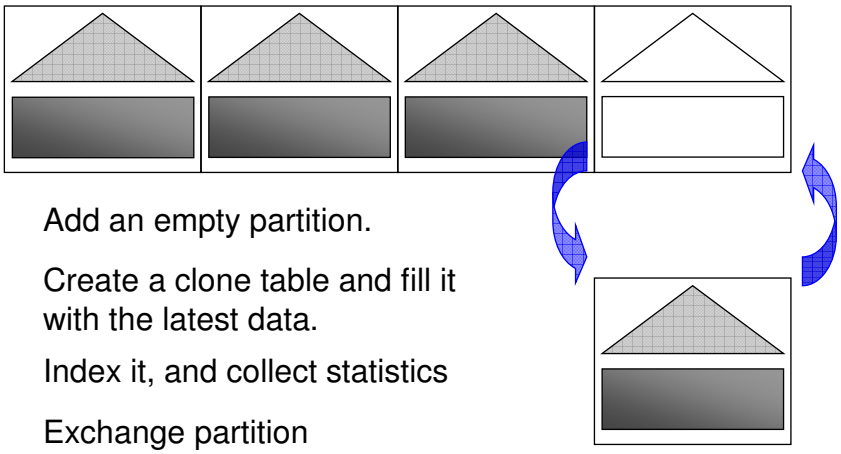
Partition-wise joins - 2



Can be "synthesised" (especially relevant to PX)



Bulk Loading - 1



Add an empty partition.

Create a clone table and fill it with the latest data.

Index it, and collect statistics

Exchange partition

Bulk loading - 2

Plus points.

The data doesn't move, it's a dictionary update.

Avoids contention and read consistency issues

But …

Global indexes still have a bulk update

Table level statistics need correction

Integrity constraints need special handling

Enable SQL trace and test everything (with some data) to see the SQL that happens under the covers.

Referential Integrity – 1

```
alter table child  drop partition p1000;
Table altered.

alter table parent drop partition p1000;

alter table parent drop partition p1000
*
ERROR at line 1:
ORA-02266: unique/primary keys in table referenced
  by enabled foreign keys
```

There may be child rows in the next partition up when you make the call to drop the "matching" parent.

Referential Integrity – 2

To drop the parent end of a ***R***eferential ***I***ntegrity constraint
You need to ***disable*** the foreign key constraint.

You can still exchange a child partition with another table,
But the foreign key constraint has to be set as ***novalidate***.
(The error message is a little misleading if you don't do this:
ORA-02266: unique/primary keys in table referenced by enabled foreign keys)

11g allows for "ref partitioning".
This doesn't apply if your parent key is part of the child key
(i.e. if your database design is correct).

Uniqueness – 1

```
alter table parent
      exchange partition p3000 with table parent_ex
--    including indexes
--    without validation
--    update [global] indexes
;
```

The resources used in the operation depend on:
- *the options chosen (indexes, validation)*
- *the constraint state (validate/novalidate, enable/disable)*

Always test with some data, and SQL trace enabled.

Uniqueness – 2

*Primary Key in **validate** state, Exchange **without** validation*

```
select /*+ first_rows(1) ordered */ 1
from
      parent_ex     a ,
      parent        b
where  a.id = b.id
and    (   tbl$or$idx$part$num(parent,0,0,0,b.rowid)  <  2
       or tbl$or$idx$part$num(parent,0,0,0,b.rowid)  >  2
       )
and    tbl$or$idx$part$num(parent,0,0,0 ,a.id)  <>  2
and    rownum < 2
;
```

Oracle checks the data integrity for all OTHER partitions.

Uniqueness – 3

*Primary Key in either state, Exchange **with** validation*

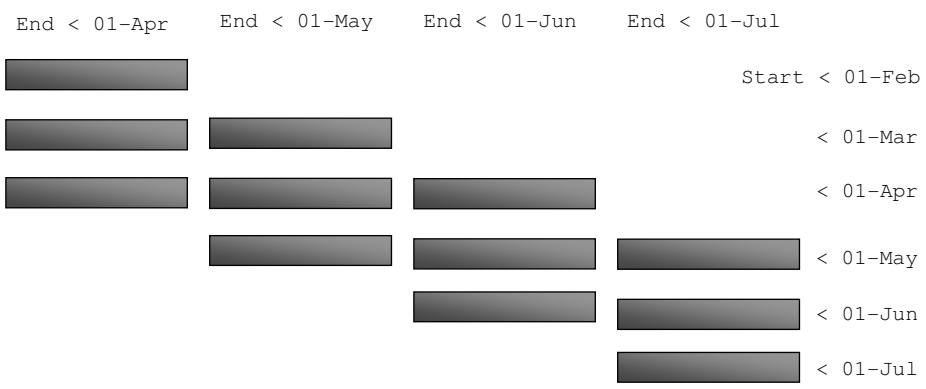
```
select 1
from parent_ex
where tbl$or$idx$part$num(parent, 0, 3,1048576,id) != :1
```

Oracle checks every new row belongs in THIS partition.

There is no under-cover check when the PK is in the novalidate state, and you exchange without validation.

Left as exercise

where :b1 between dt_start and dt_end ?



11g offers more options in partitioning. Some (e.g. interval partitioning) are administrative – but range/range composites may be very useful in carefully constructed special cases.

Summary

Partition for:

- Housekeeping (partition maintenance)
- Reducing contention
- Partition elimination / “free indexing”
- Partition-wise joins

Threat points when partitioning.

- Impact when elimination does not occur
- Issues with unique and referential integrity
- Effects of local or global indexes