Are Indexes Unnecessary in Exadata?

Arup Nanda
Longtime Oracle DBA
(and now DMA)
Disclaimer

If you downloaded this slide deck, please note:

These slides are designed merely as props to help my presentation. They are not intended to stand as independent sources of information. Therefore the contents of the slide deck are not meant to be exhaustive in any way of the content delivered at the session.
Quotes

“ You don’t need indexes on Exadata.

“ Drop all the indexes and reclaim space.

“ Why? Because there is a storage index.
3 Questions for “Best Practices”

1. Why it is better than the rest?
2. What happens if it is not followed?
3. When are they not applicable?
Storage Index
SELECT NAME
FROM CUSTOMERS
WHERE STATUS = 'ANGRY'

Instance
Combination of
• Memory Areas
• Background Processes

Are Indexes Unnecessary in Exadata
Query Processing

SELECT NAME FROM CUSTOMERS WHERE STATUS = 'ANGRY'

Are Indexes Unnecessary in Exadata
Are Indexes Unnecessary in Exadata
Components for Performance

- CPU
- Memory
- Network
- I/O Controller
- Disk

**Less I/O = better performance**
The Solution

• A typical query may:
  – Select 10% of the entire storage
  – Use only 1% of the data it gets
• To gain performance, the DB needs to shed weight
• It has to get less from the storage
  → Filtering at the storage level
  → The storage must be cognizant of the data

```
SELECT NAME
FROM CUSTOMERS
WHERE STATUS = 'ANGRY'
```

Filtering should be Applied Here
Are Indexes Unnecessary in Exadata
Are Indexes Unnecessary in Exadata

Cell Server

ESS

Disk1
Disk2
Disk3

iDB
SELECT ... 
FROM TABLE 
WHERE COL1 = 2
Storage Indexes

• Do not point to the database blocks
• Merely stores for a Storage “Unit”
  – Max/Min Values
  – Whether nulls are present
  – For some columns
• Is on Memory of Cells; not disk
  – Disappears when the cell is down
Checking Storage Index Use

```sql
select name, value/1024/1024 as stat_value
from v$mystat s, v$statname n
where s.statistic# = n.statistic#
and n.name in ('cell physical IO bytes saved by storage index',
'cell physical IO interconnect bytes returned by smart scan')
```

Output

<table>
<thead>
<tr>
<th>STAT_NAME</th>
<th>STAT_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI Savings</td>
<td>5120.45</td>
</tr>
<tr>
<td>Smart Scan</td>
<td>1034.00</td>
</tr>
</tbody>
</table>
Offloading and Smart Scan

Offloading
Processing to storage cells

Smart Scan
Reduction in I/O
Offloading

- Column Projection
  ```sql
  select cust_id, sale_amt
  from sales
  ```
- Predicate Filtering
  ```sql
  where status = 'ANGRY'
  ```
- Function Offloading
  ```sql
  select min(sale_amt)
  ```
- Virtual Columns
Smart Scan Benefits

• Less I/O means
  – Faster disk access time
  – Less data from storage to DB
  – Less buffers
  – Less CPU
  – Less data between compute nodes
Why Not?

• Pre-requisite for Smart Scan
  – Direct Path
  – Full Table or Full Index Scan
  – > 0 Predicates
  – Simple Comparison Operators

• Other Reasons
  – Cell is not offload capable
    • The diskgroup attribute cell.smart_scan_capable set to FALSE;
  – Not on clustered tables, IOTs, etc.
Impact of Data Distribution

SELECT ...
FROM TABLE
WHERE COL1 = 2
8 Columns

Table T1

C1
C2
C3
C4
C5
C6
C7
C8
C9
C10

Table

Arup Nanda

Are Indexes Unnecessary in Exadata
No Predicate

- Aggregations
  ```sql
  select sum(sale_amt) from sales
  ```

- Sorting
  ```sql
  select ... from sales order by sale_amt;
  ```

Index on
SALE_AMT
Function Based Indexes

• Traditional Indexes can’t work
  select ...  
  from sales  
  where to_char(sale_dt,'YY') = '13'

• Function Based Indexes help

• SI indexes will not be useful
IOTs

- Index Organized Tables
- PK-based rows
- Secondary Indexes built on the other columns
Clustered Tables

Are Indexes Unnecessary in Exadata
Exclusion for SIs

• Not for non-equality
  
  select sale_amt  
  from sales  
  where status != 'SHIPPED'

• No Wildcards
  
  select sale_amt  
  from sales  
  where city like 'NEW YORK%'

Arup Nanda
Virtual Columns

• Example
  alter table EMP add (  
    tot_sal number(13) generated always as sal+comm)  
 )

• Implication
  – Do not actually exists in the table
  – Computed at runtime
Indexes on Small Tables

• Small table
  – Parameter `_small_table_threshold`

• Indexes still help small table
  – Less latching
Summary of SI Limitations

• Direct Path not used
• No Predicate ➤ No SI
• No Inequality (!=)
• ≤ 8 columns
• No Virtual Columns
• No wildcard match (LIKE ‘..%’)
• No IOT, Clustered Table
• Latching on small tables
• First-timer Penalty
  — Only subsequent queries benefit
These are flash cards presented as disks; not memory to the Storage Cells. They are similar to SAN cache; but Oracle controls what goes on there and how long it stays.

```
alter table person storage (cell_flash_cache keep)
```
Flash Trick for Indexes

• Pin Oft-Used Objects in Flash
  SQL> alter index in_t2 storage (cell_flash_cache keep);

• Check flash
  CellCLI> list flashcachecontent attributes –
    cachedKeepSize, cachedSize, hitCount, –
    hoursToExpiration, missCount –
    where objectnumber = 382380;

• Or, partitions
Drop the Index?

• Make the indexes invisible
  SQL> alter index i1 invisible;
  – Maintains the index; but optimizer ignores it

• See the performance impact.

• Selectively see the impact
  SQL> alter session set optimizer_use_invisible_indexes = true;

• See the performance impact.
Disable

• Two parameters
  – Could be session level
• To disable offloading
  `cell_offload_processing = false;`
• To disable storage indexes alone
  `_kcfis_storageidx_disabled = true;`
In Conclusion

• Full table scans in Exadata
  – may be faster compared to non-Exadata
  – may not be faster than index scans in Exadata
  – may benefit from Storage Indexes

• Storage Indexes are not same as DB Indexes

• No DB Indexes helps in some cases
  – But not all

• Test by making DB Indexes invisible

• Force FTS in those cases where index hurts
Thank You!

Blog: arup.blogspot.com | Twitter: arupnanda