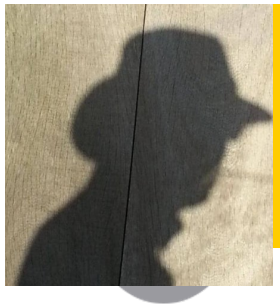
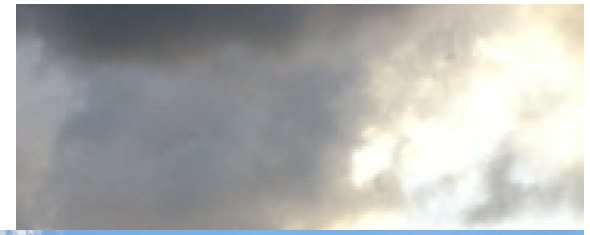


Piet de Visser - PDVBV

hroug
godišnja konferencija
13.-16.10.2020.
Rovinj, Croatia

# Partitioning

# Positives and Pitfalls…




**Piet de Visser**
**Simple Oracle DBA**


Commit Your ORACLE Knowledge

Favorite Quotes: “The Limitation shows the master” (Goethe), “Simplicity is not a luxury, it is a necessity. Unfortunately, “Complex’ solutions sell better. (EW Dijkstra).

# Logo Cloud

PDVBV

SOLUTIONS THAT MATTER


- Portbase
- (dutch gov)
- Shell
- Philips
- ING bank
- Nokia
- Insinger, BNP
- Etihad
- NHS
- BT
- Claritas, Nielse
- Unilever
- Exxon
- GE

Commit Your ORACLE Knowledge



Don't waste time on Self-Inflation… but Hey, this was such a cool Idea (from a marketing guy)…
Logos of my major customers over time. If you want your logo here: Hire me.

# What does it look like..

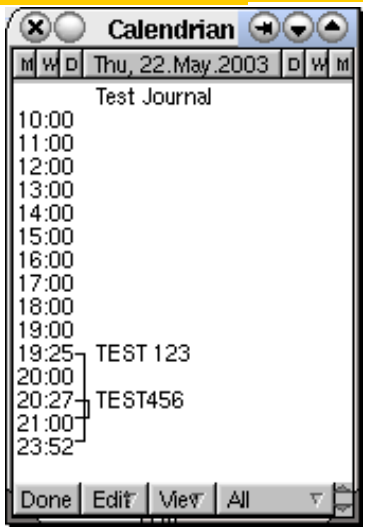
PDVBV

SOLUTIONS THAT MATTER


# Agenda (45min +/- my “Dev” preso.. )

PDVBV

SOLUTIONS THAT MATTER




**Partitioning…**

**Summary: Design !!**

**(see final slides. ;-) )**

**Top-Tip: Keep It Simple.**

**10+ min Discussion (Virtual ….???)**



Agenda. No longer allowed when presenting online (c.f. Connor…)
Oh, BTW: I am known for Typos.. Find a typo = get a drink..

# Basics; Why Paritioning ?



- Partitioning: Split 1 table into “Many”

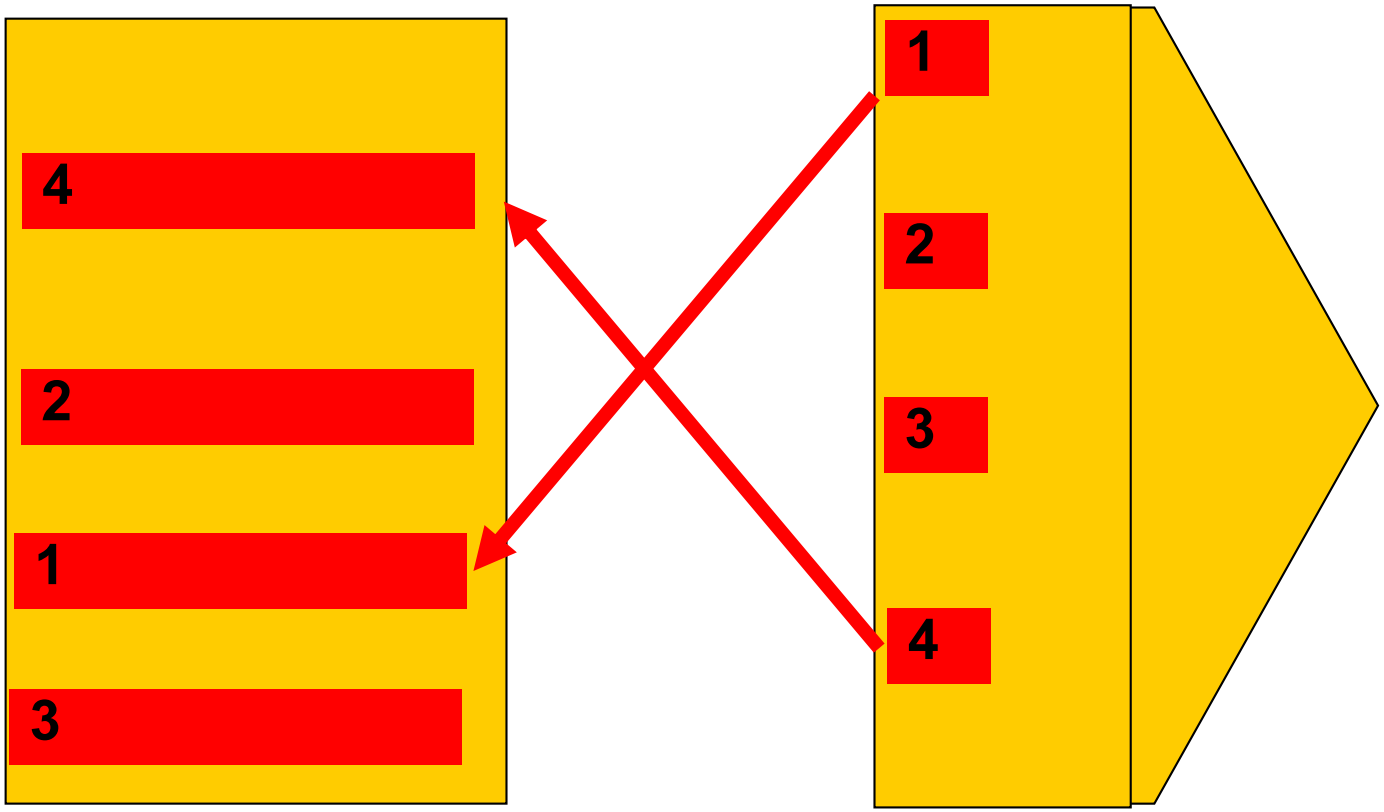
- Two Main Advantages:
- 1. Avoid Redo
- 2. Scan less data on Qrys.
- Many more… later.
  - Compress partitions..
  - Read-only, storage tiers
  - Partial indexing
  - Ref-partitions.
  - Hybrid Partitioned-tbls…. WOW!
  - Later (next month’s ppt...)




(paid-for-EE-option…) Two main Advantages, will try to illustrate both
Piffalls later… other advantages: later. (add coffee break…)

# Table and Index. Conventional.





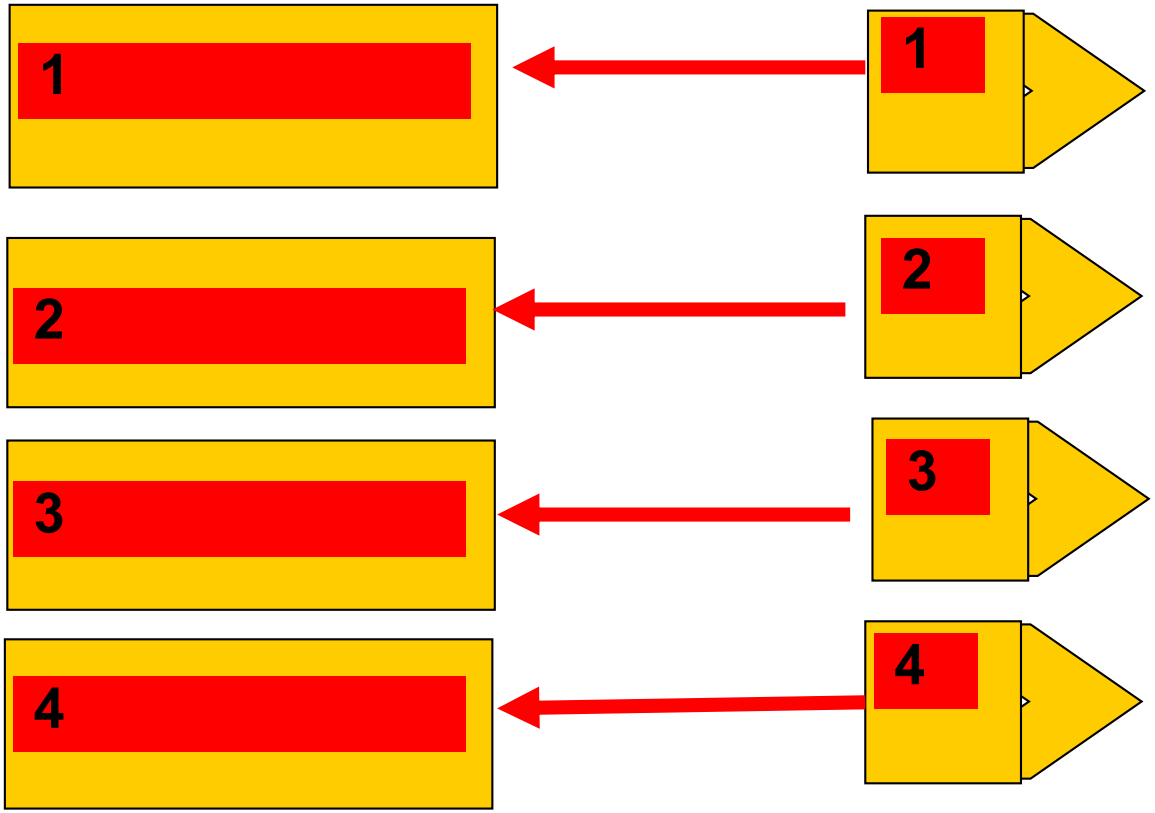

- Table
- (Global) Index on ID



A quick illustration of table and indexes…. Data in the tabler is randomly spread out, but the inxexes contain ordered lists and pointers to thre table-records.

# Partitioned tables… (and local index)






**Smaller pieces**

**“known” content**

**Still 1 Table.**

**Local index !!**



A quick illustration of (range) partitions and local indexes…. Partitions are just small tables with known (ranges) of data.. Oracle “Knows” those ranges.

# Partitioned tables; Global index; Active='Y'



## Table still Partitioned..

## GLOBAL index, Points to all Parts

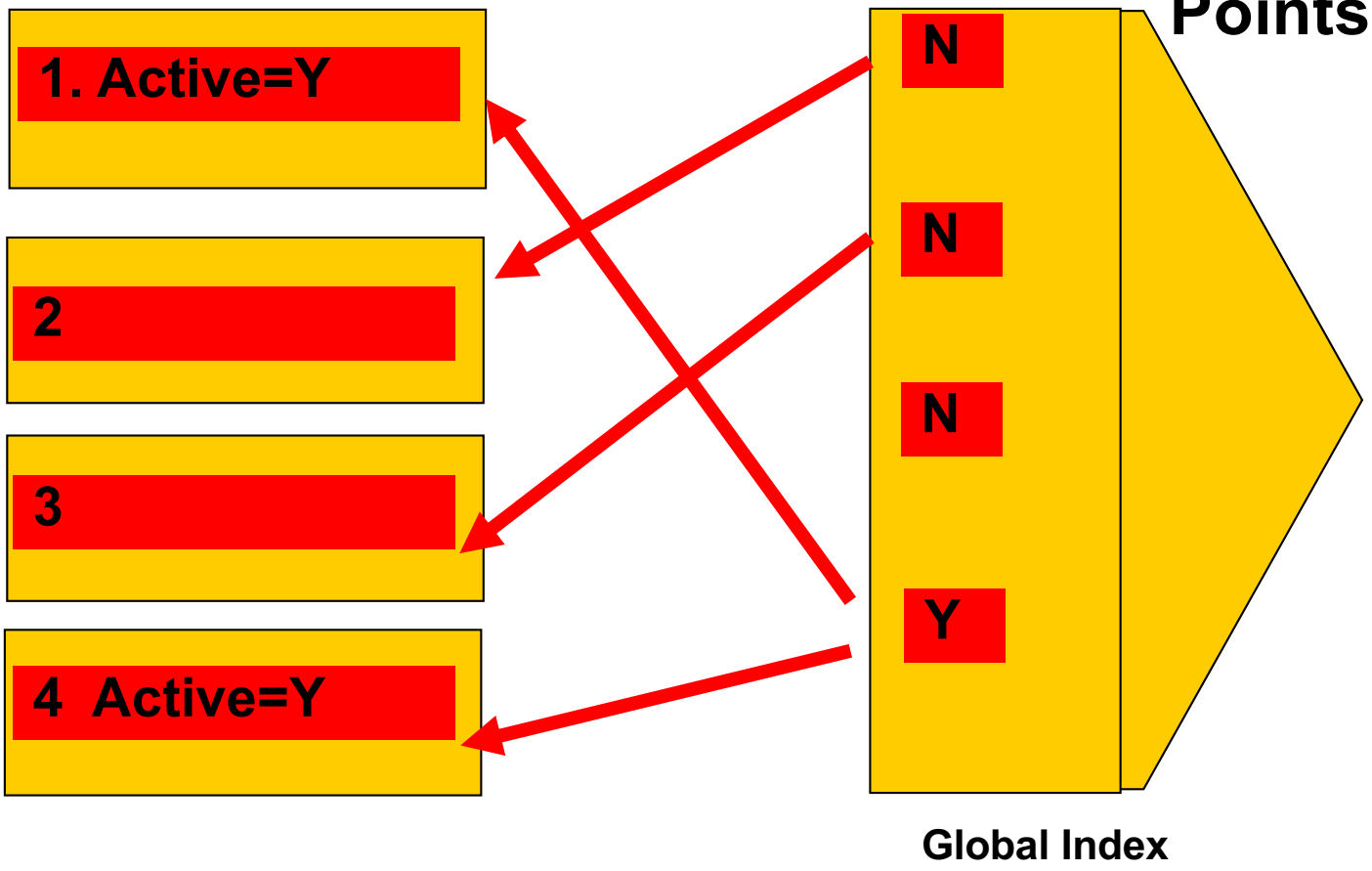





illustration of GLOBAL indexes… The index is now One Single object, Pointing to all partitions.
The impact of this will be noted later: on Redo and on Queries.

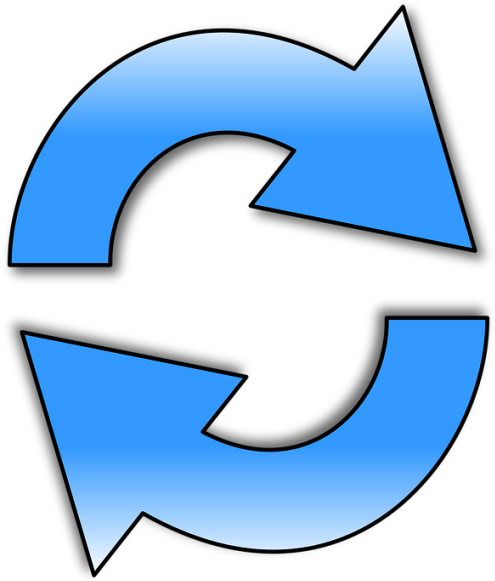
# 1st Advantage: Less Redo (on delete…)

PDVBV

SOLUTIONS THAT MATTER

- Ins / Upd / Del is “Work…”
  - Undo + Redo… ( ~ WAL)
  - Redo = Arch = Stndby…

- Delete?
  - Drop or Truncate is “Faster”



- You Can!  - Drop Partitions!
- But…
  - Only if your partitioning is suitable.
  - Only on “delete” (or exchange partition)

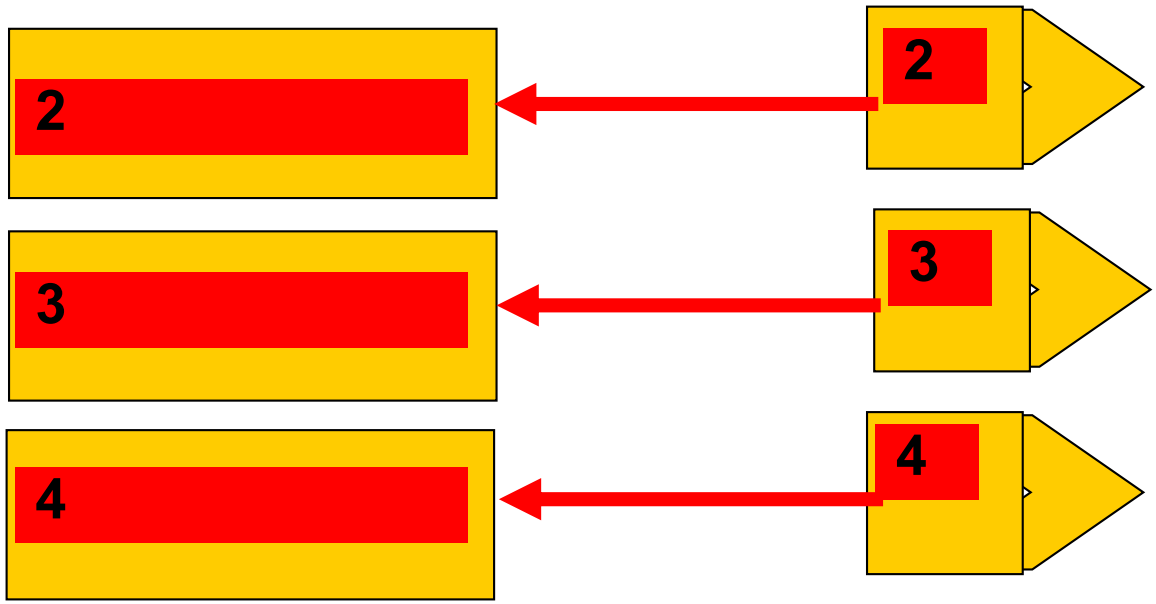


Explain deleting old data with dorp-partition.
Typical use-case: ingest + remove of data with limited lifetime in the DB.. You can save half the redo..

# Drop Partition… (Fast, no-redo)



**SQL> Alter table PT drop partition PT_1 ;**




Instantaeous Delete of the “range” inside a partition. Vry Little Effort.
Note: inserts and updates will still require redo… and Global indeses.. Well, just wait.

# Partitioned table; Global index;



## SQL> Alter table PT drop partition PT_1 ;

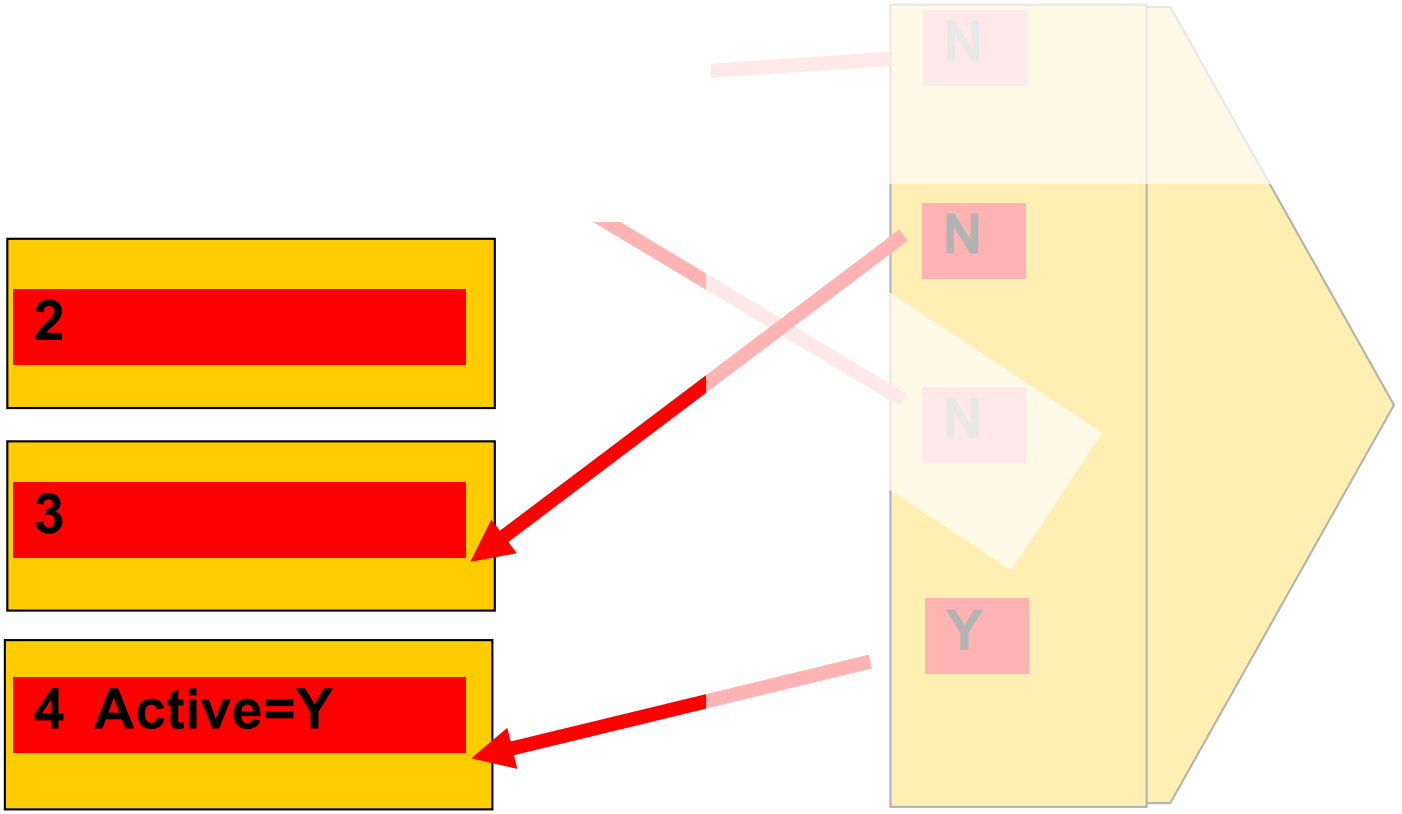





illustration of GLOBAL indexes… Can no longer “truncate” index, index points to whole range..
On “drop-partition, will need rebuild of index…

# Demo time..



- T = Table
- PT = Partitioned table

- Delete from T => redo
- Delete from PT => still redo..

- Drop partition =>  Much More Efficient..

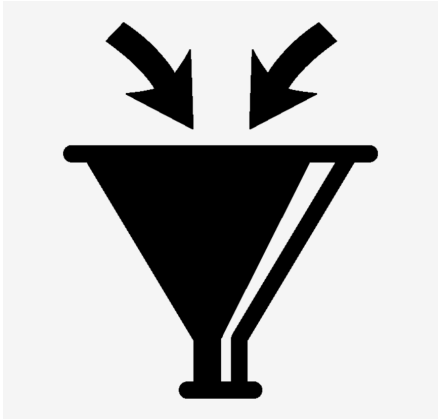
- SQL > @demo_part
- SQL> @demo_part_0
- SQL> @demo_part_0a (with global index…)

# 2nd Advantage: (some) Queries Go Faster…

PDVBV

SOLUTIONS THAT MATTER

- Scan Less Data
  - less blocks, less IO, less Cache
- Typical use-case:
  - Queries / Aggregates over 1 or few Partitions.
- Anti-pattern:
  - Loop over All Partitions… (later)
- Next slides: show me how..

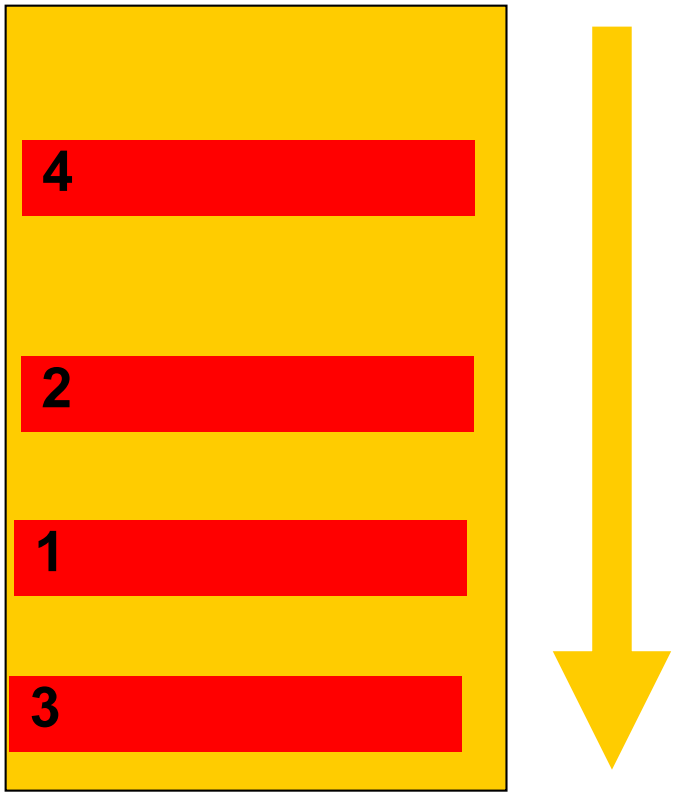




Ideally, queries scan as little data als possible to return results .. Fast
Redue the work…

# Aggregates, FTS over Conventional table






•Table

- **Data all over the Table..**

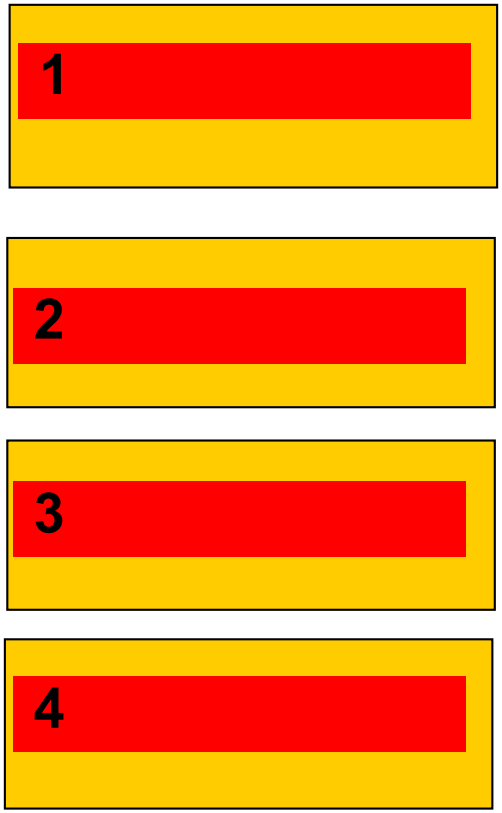
```
Select Sum (amt)
Where [range]
Group by ..
```

- **Probably FTS**

# Aggregates on Partitions: less data to scan?

PDVBV

SOLUTIONS THAT MATTER




- IF… we know where to look..
- Then… FTS on…
- just 1 Part. ?
- Design !
  - Know your data.
  - Control your SQL



Some (most) searches / scans can be limited to just the relevant partitions..
This Will Only Work if we can eliminate sufficient partitions. (Design!!). - note : No Indexes.

# Demo time..



- T (Table)
- PT (partitioned)

```
Select Range, SUM(amt)
From T/PT
Where range Between 10000 and 19999
Group by Range;
```

- SQL > @demo_part
- SQL> @demo_sum



This is what we will see. In demo.. -- What do we Expect ?
(don't forget to initiate the data)

# More Queries: Find Specific Records



- `Where ID = :n`

  Find 1 record; Easy, use (local) index.

- `Where Active = 'Y'`

  Find Multiple records, all over…

  Global index..? But … Redo?

  Local Index..? But … How many Partitions ?

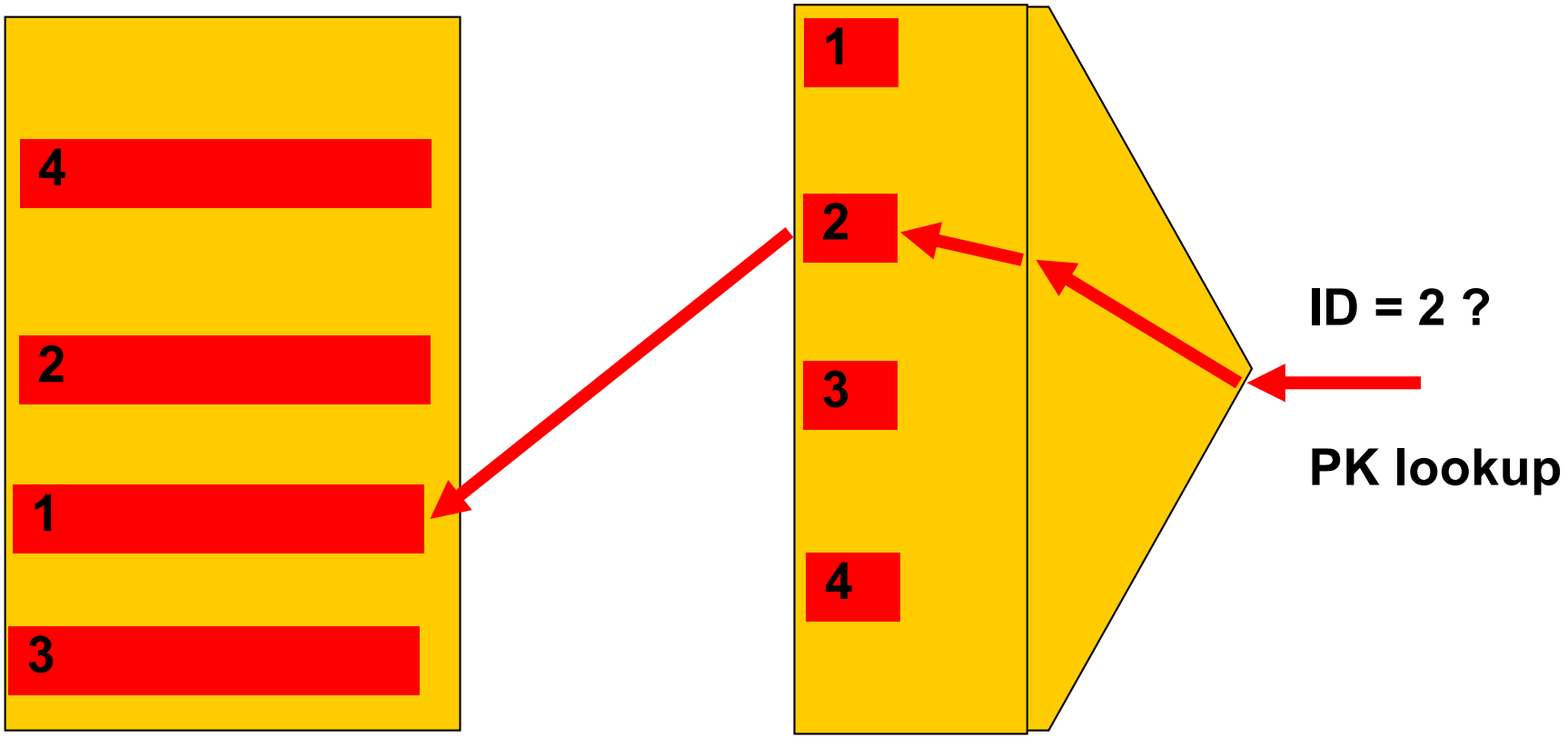
- Anti-pattern:
  - Loop over All Partitions…



When you need "Fast" return of a small set, you need an index… Global or Local
But avoid having to loop/scan many partitions…

# Conventional. QRY for 1 record; on PK/UK.






- Table
- (Global) Index on ID

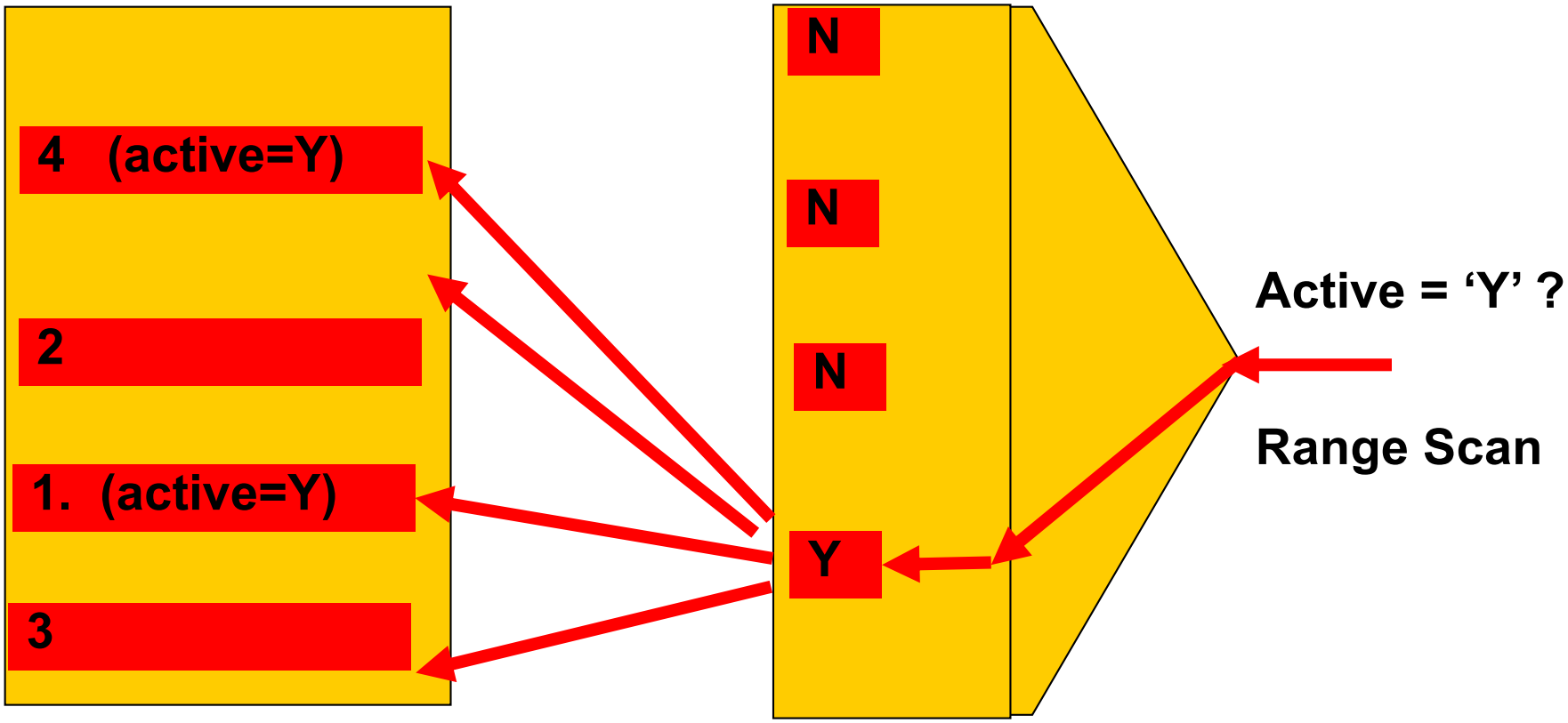


A quick illustration of table and indexes…. Data in the tabler is randomly spread out, but the indexes contain ordered lists and pointers to the table-records.

# Table, index… QRY for a set; Active=Y

PDVBV

SOLUTIONS THAT MATTER




•Table

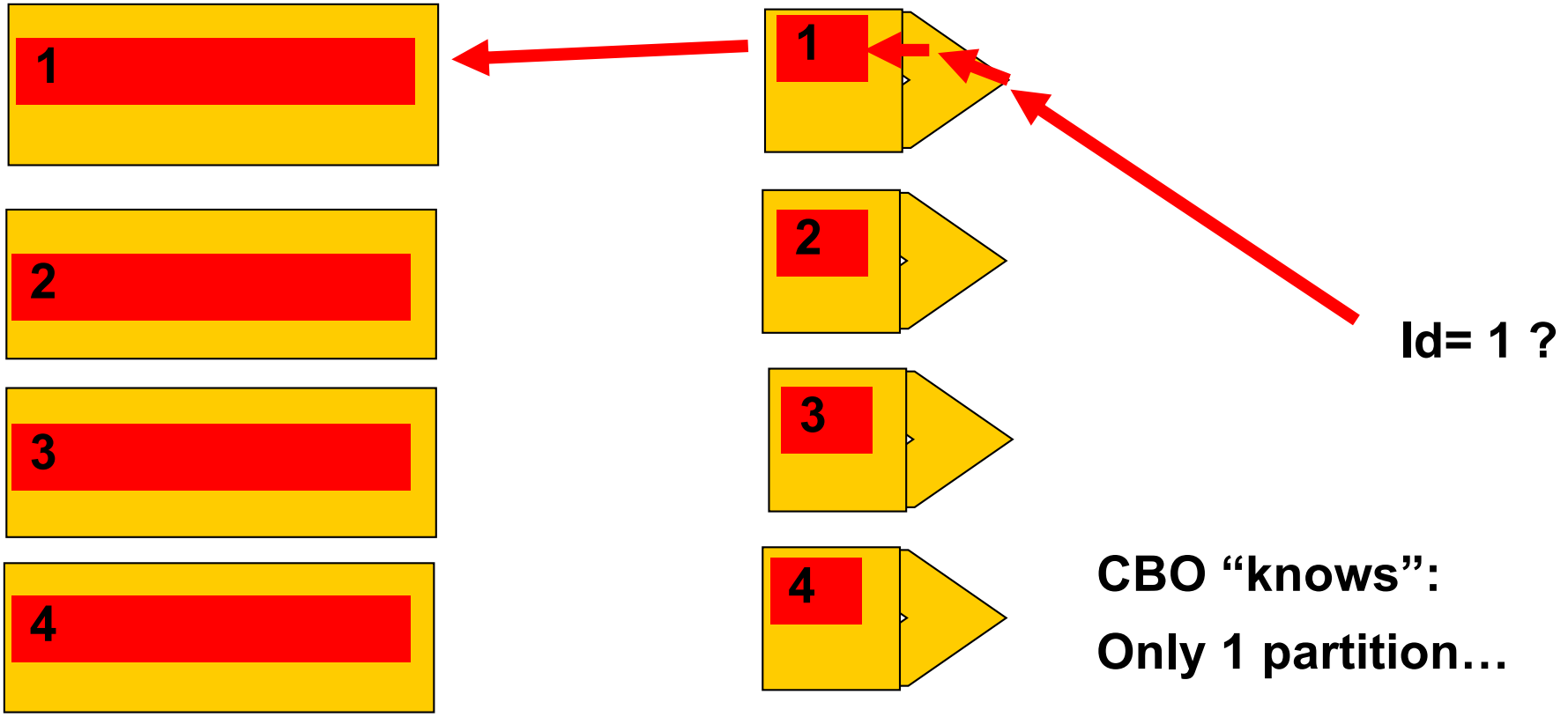
•(Global) Index on active..



Same situation, different index.
The few active=Y fields can be all over the table (and in all partitions..).

# Partitioned table + local index on PK








Searching for the PK or partition key is Easy… Visit 1 local index, and find the record.
CBO can see from the where-clause which (local) index-partition it needs…

# Global index; Active='Y'



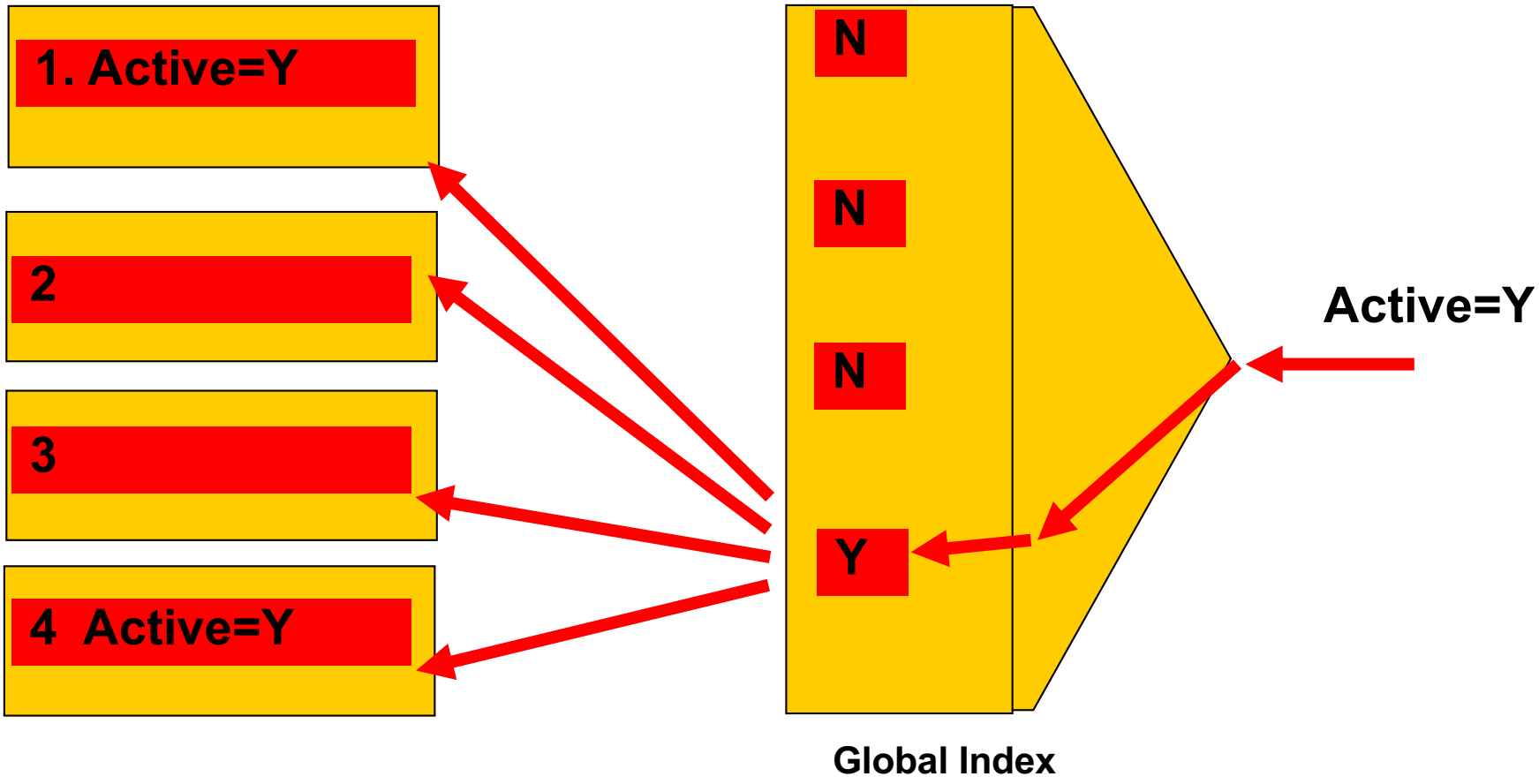
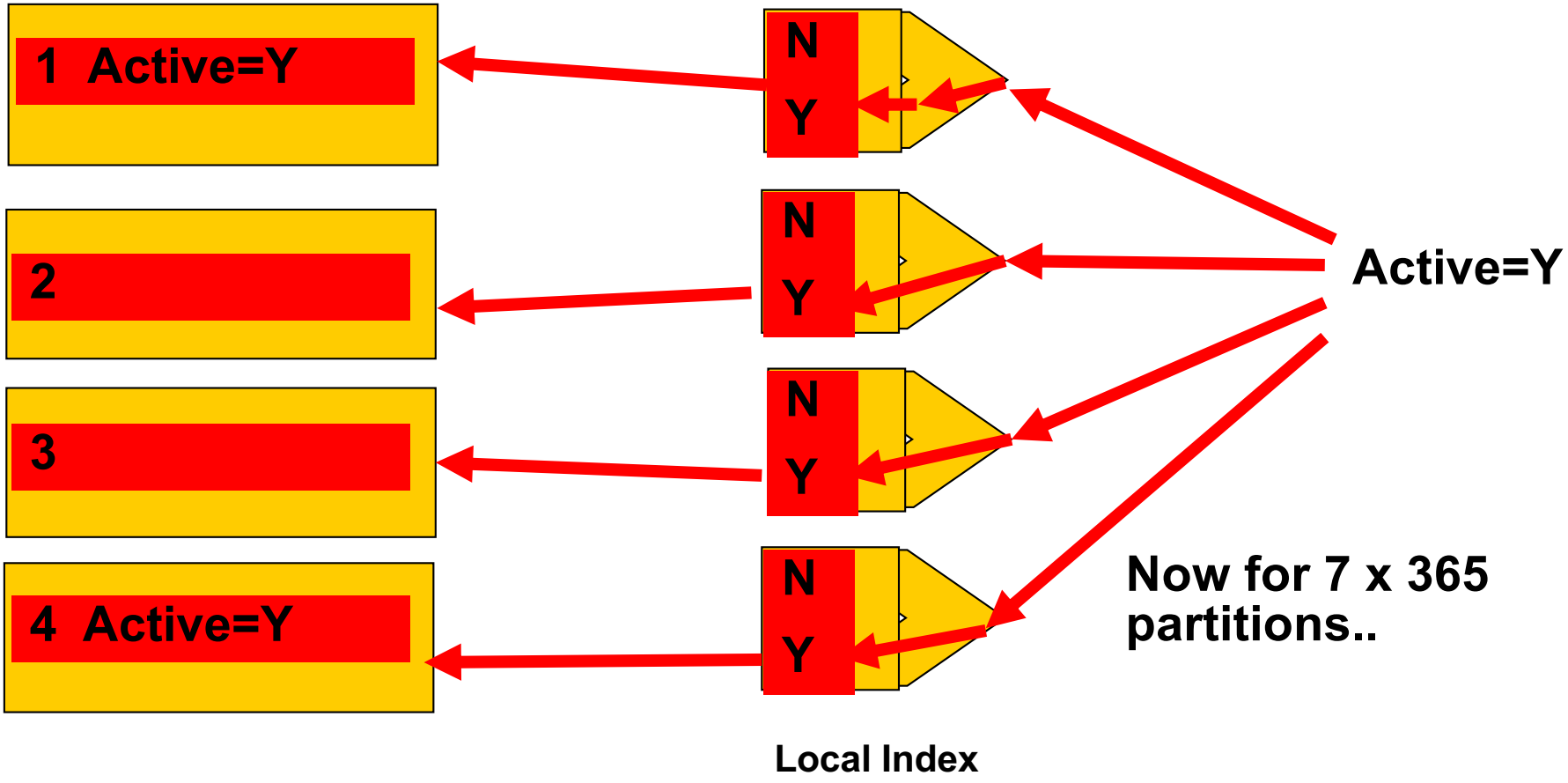



illustration of GLOBAL indexes…. And the ups and downs, SQL is equally efficient as on "Table"
But Point out the need for rebuild if you drop 1 partition: 25% of pointers is gone…

# LOCAL index, active=Y…








If the SQL does not gives us a clue for the Partittion,
We need to Search Through Every Local Index… (Parition-Range-All.. Looping)

# Demo time..



- **PT (partitioned)**

```
Select id, active
From PT
Where active = 'Y';
```

- **Compare GLOBAL and LOCAL index.**

- **SQL > @demo_part**
- **SQL > @demo_part_1**



This is what we will see. In demo.. -- What do we Expect ?
(don't forget to initiate the data)

# Pitfalls; What to Avoid…

PDVBV

SOLUTIONS THAT MATTER

- Avoid Global Indexes
  - Extra work on drop-partition
- Avoid “Partition Range All”
  - Looping, multiplies the work…
- Consequence:
  - All Qries Need “The Part-Key”
- Up Front Design!




Two main Advantages, will try to illustrate both
Piffalls later… other advantages: later.

# Bonus-Trick: a PK-Key for Partitioning. 1/2



(not saying this is a good idea… YMMV ! )

- Partitions = mostly a “date thing”
  - Not always: List-part on Cstmr-ID also happens.

- No Global Indexing
- Only 1 Unique Key
- Hence UK = PK = Partition key.

- (did I say: Up Front Design?)

# Bonus-Trick: a PK-Key for Partitioning. 2/2



- Two part key (64bit integer)
  - Date + Seq: YYYY DDD SSSS nnnnnn
  - Seq: nnnnnn, cycling at 999,999

- Id = “epoch” + seq (16 digits)
- Id = YYYY DDD SSSSS + seq (18 digits)
- Id = YYYYMMDD HH24MISS + seq (20 digits)

Also check : “GUID as PK” (@franckpachot)

Bonus demo: SQL> @demo_part_pk

# Summary (the watch of the cstmr)

PDVBV

SOLUTIONS THAT MATTER

- Partitioning: Only From Design.
- 1. Less Redo: No Global indexes (yet..?)
- 2. SQL: (fast) Queries need the Partition Key.


- Use(ful) Cases:
  - Limited ( it is not “cloud” …)
  - Time Series
  - Fast Moving data (batch-deletions…)
  - List partitioning = Sharding (discuss !)

- Know + Control your Database + App.

# Interesting Times Ahead…



- Many Improvements
  - (global indexes – are improving)
- Many New Features.
  - Partial indexing
  - Ref-partitions
  - Hybrid Partitioned-tbls…. Wow ??!
- Discuss
  - What should be in next month's ppt...



Watch this space… Lots of interesting new features + tricks.
Would love to test some of those for Real… But. Beware of over-engineering.

# Don’t Take my word for it…



**RTFM: start with concept-guides**

**Test.**

**@sdjh2000 (Hermann Baer @ Oracle)**

**Simplicity**

- **In case of doubt: Simplify!**

**SimpleOracleDba . Blogspot . com**

**@pdevisser (twitter)**

**Goethe _____..... (simplicity)**

# Quick Q & A (3 min ;-) 3 .. 2 .. 1 .. Zero

PDVBV

SOLUTIONS THAT MATTER

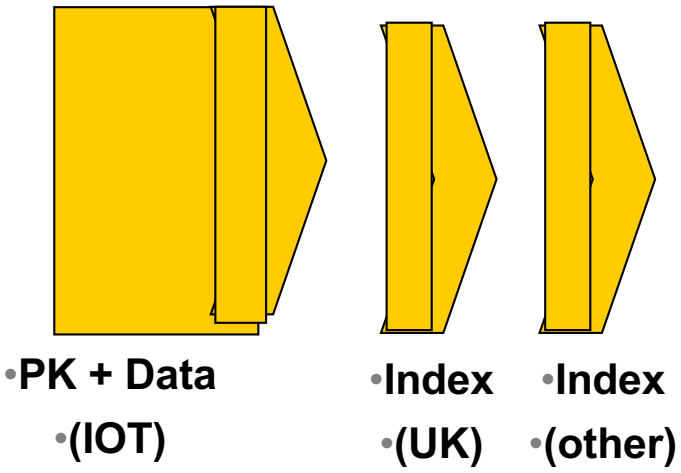
- Questions ?
- Reactions ?
- Experiences from the audience ?
- @pdevisser (twitter..)

# Other Paths into data: IOT 1/3



- Index Organized Tables
  - Overloading to the extreme: all data in the PK.
- Group and Order data by leading columns
  - Ideal for Parent-child tables: Children Forced together.
- Also Good for (small) Lookup-tables (TomK, RichardF)

- IOT : one less segment..:





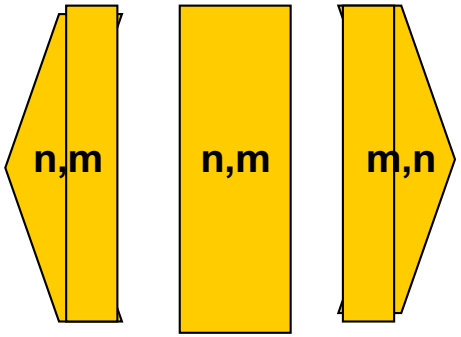

IOTs simply don't have "tables", all data is in the PK.
You have one less segment, there is no more table, no more un-ordered heap of records…

# IOTs only get Better… 2/3



- **Bonus-feature on IOTs: Fat Indexes**
  - 2ndary indexes are “overloaded”
  - contain the PK-values (as rowid) to allow Access to PK (+data)

- **Good for n:m relationships and join-only access**
  - Normally, you need TBL + PK + FK
  - (you can “overload” to get index-only-access)
  - The IOT does the overloading for you…
  - And removes the “table” segment altogether.




- **Show grouping + bonus-feature.** SQL > @cr_di2.sql

# IOTs

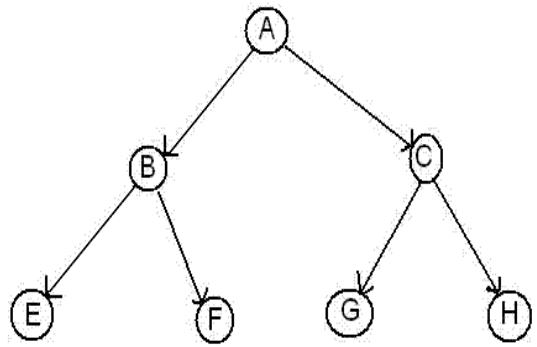
- Slow I
  - Esp
  - (pe
- Overfl
  - Pla
- 2ndar
- “Table
- Statis
- Bugs.


Knowledge

# Recap 1/2



- Any (Btree) index is: Data + pointers, stored in order
- Index + statement, (DBA and Dev) must work together:
  - Good: Leading columns in the Where-clause
  - Better: All where-conditions in the index (smallest slice)
  - Even Better: Order-by from Index, Prevent sort
  - Best: All data from Index, don't visit the Table
- Various books, but … Tapio Lahdenmäki !
  - All you need to know about "good" indexing.
- Demos: index, overloading, IOT.






Tapio : "the book" on indexes. 1, 2, 3 stars and Fat-indexes.
Demo: cr_di.sql (create demo index) – show explain, shnow nr-gets per sql.

# Recap 2/2 : INDEXING



- Verify Access Paths (especially on OLTP):
  - Explain, (auto)trace and check v$sql and v$sql_plan
- Good Indexing: good, better, best…
  - will help you more then anything
- Overloading is useful
- Index-Compression is useful (but test)
- Clusters and IOTs … If applicable (but test)
- Now for the C-B-O… (paracetamol…)

# Optional Intermezzo: Poll for audience



**Thank You !**

**Stay Tuned**

**(not done yet…)**

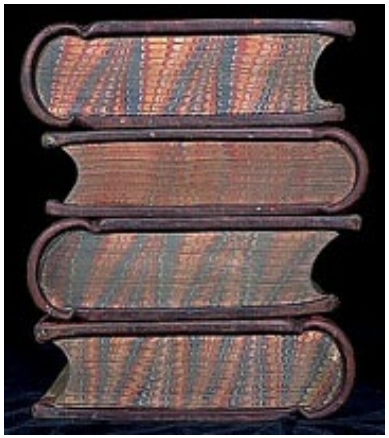


Find out from the audience.
Bonus question: who prefers init-files ?

# Eh, just to remind you … Simplicity



- **Leonardo da Vinci:**
  - **Simplicity is the ultimate sophistication.**
- **Goethe:**
  - **In der Beschränkung zeigt sich der Meister".**
- **EW Dijkstra:**
  - **Simplicity is a pre-requisite for reliability.**
  - **The sore truth is that Complexity sells better**

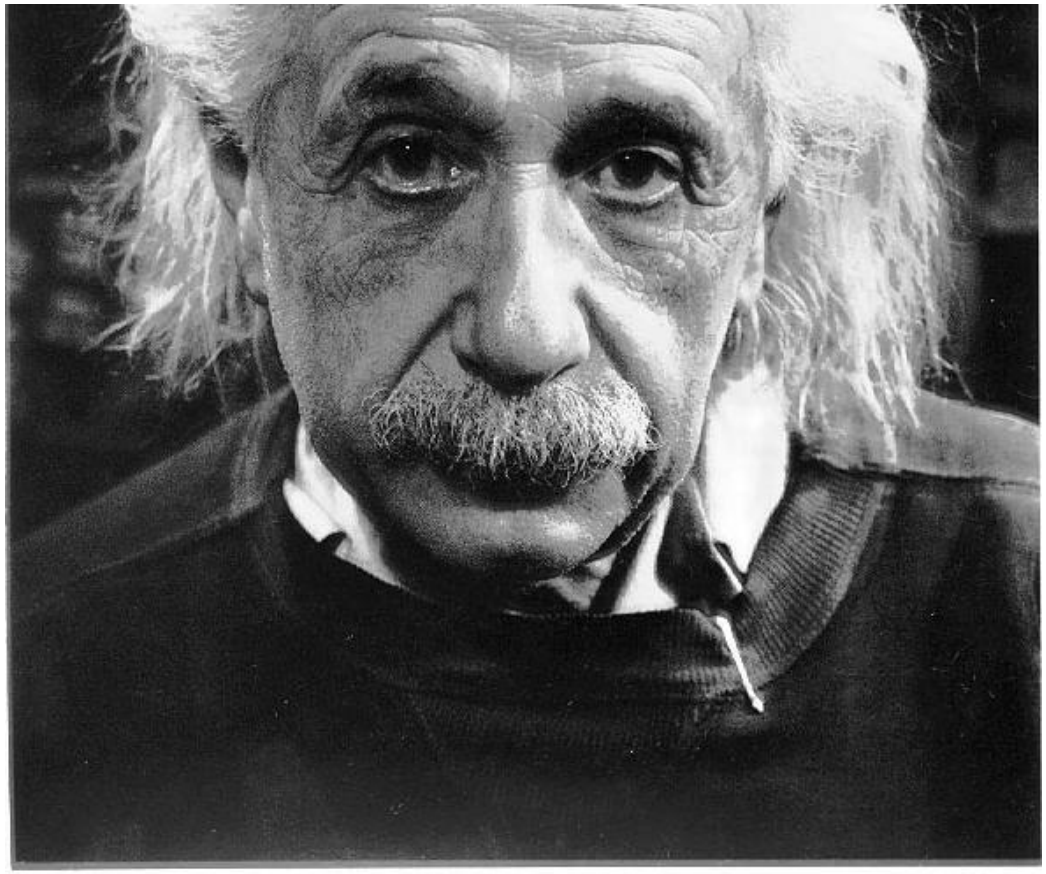




www.brainyquote.com,
Quotes from Leonardo da Vinci, Albert Einstein, and Edsgar W Dijkstra.

# He got it …





"If you can't explain it simply, you don't understand it well enough"



As Simple as Possible, but not too simple
Simplicity is a Requirement – but Comlexity just sells better (EWD).

# Tak

# BV

MATTER

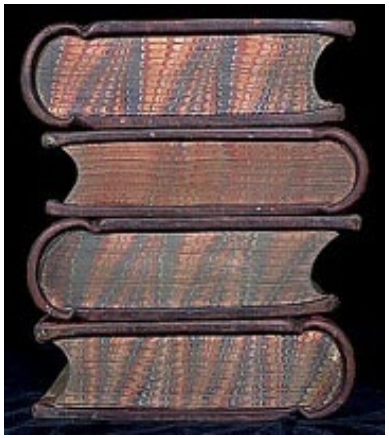
- De
- IND
- CB
- Sta
- Tw


Knowledge

# Notes

- Road + map are good metophores.

- Occams Razor
  - Least possible ASSumptions
  - Simplest Solution.





LoE: Layer of Expertise ( © to Moans for this abbrev.?)
Simplicity at each Layer of Expertise is essential for a robust system.

# CBO basics

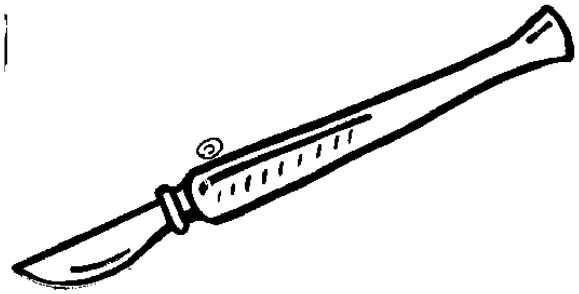
- CBO is like TomTom (= Garmin); Very Clever, but…
  - Do you ever mess with your Tomtom ?
- You need good Roads - hence my rant on Indexes.
- TomTom needs “the map”
- TomTom needs good “settings”
- Sometimes it needs common sense
  - Hints, sqlplans, SPM.
- And Somtimes it needs a Spanking.

# What do these have in common…

PDVBV

SOLUTIONS THAT MATTER

- SQL > COMMENT on table emp is ‘you are it’;
- SQL > GRANT select on emp to perfstat;
- SQL > ANALYZE table emp estimate statistics ;
- SQL > alter system flush shared_pool;
- SQL > .. You may have more of these …



- Pre-11 shooting of a cursor!
- Never quite “precise”, but they generally work.
- New: dbms_shared_pool.purge( cursor ) (c/o “Prutser”)

# CBO runs on Information





- Spfile-parameters
- System stats
- Object Stats
- Session-parameters
- Outlines or SQLPlans
- Hints (if you really have to…)
- (and … sometimes it “needs to get lucky” – c/o JL)


- Realize the hierarchy: Order
  - Troubleshoot from bottom to top!



There is a hierarchy in the information used by CBO. Troubleshooting should take this into account.
Start with changes that are “precise”, aimed at your problem-only, prevent wider impact of “tweaks”

# Impact on the CBO… 2/3



- Spfile: Simplify; get rid of ALL init.ora “history”.
  - Any change = system wide (Don’t Mess here!)
  - Optimizer_mode… (dflt Choose is fine)
  - Hash-/Sort-area-size (Session level, if at all)

- System Statistics (often overlooked):
  - Gather on your hardware (CPUs, disk-behaviour)
  - Tip: Collect + Plot over time, get a feel for your system.
  - Set system-stats manual ?…
  - (See book by Christian Antognini, but Need more Info)



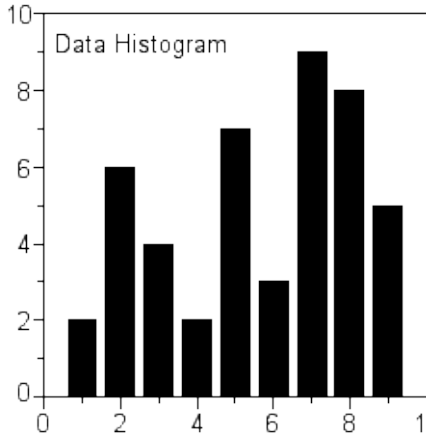
System stats tend to vary, notably if hardware changes (vmware, dynamic hardware provisioning) and SAN behaviour over the day – re-check the book and the docu! Funny: ML **368252.1 (for EBS users)**

# CBO needs More Information 3/3



- Session-parameters
  - Override Spfile-parameters – for duration of session.
  - (I don't mess with these, but you can…, optimizer_mode)

- Object Statistics
  - DBMS_STATS (But I "analyze" when in a hurry)
  - Can be "Set", copied, tweaked.
  - In doubt: use worst-case stats and LOCK.
    - if it works on 60M, it works on 120 records too.
  - New Month, New Partitions: Stale.



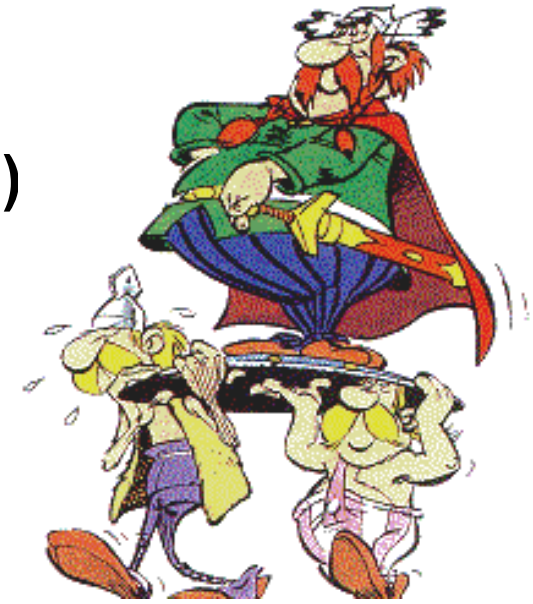



Session settings will (only) last for the session. Beware how long your session lasts.
Table/index_stats will auto-gather… do you want that ? What about new, "empty" objects ?

# Gathering Stats.. (old disucssion) 1/2



- **Simplify ( be Lazy): Use default gather_stats_job**
  - **Gather_database_stats_job ( internal use only … ?)**
- **DBMS_STATS = Heavy (and unpredictable)**
  - **Check the Maintenance Window**
  - **(and learn to use the scheduler )**
- **Save Stats you trust – for re-import (=effort?)**
- **10g: Restore-stats: Safety-net.**
  - **Retention of 31 days…**





Long and hard discussions on "when + how to analyze" Just do auto-gather + stale… but…
Beware of the Job, control the window. Nb: Restore is different from exp/imp stats !

# Gathering Stats.. 2/2



- Stale, 10%... : Lock stats you trust! (but how long…?)
  - Check for stale anyway (=work…).
  - Locking of stats: for any use of the segment
  - Locking of SQL (hint, outline, sqlplan) : per stmnt…)

- By Exception only: set or tweak stats.
  - Volatile tables, GTTs … maybe… (I hestiate, ... more work…)
- "Upgrade took two weeks to stabilize…" (Thx!)
  - You need an upgrade-strategy,
  - Whitepaper + outlines/SQLplans!



Complications around gathering stats… sigh… the "automation" is good, but beware of unpredictable changes. 10%-stale is a bit low. Lock the ones you trust. Upgrade was my intro into "outlines"

# Outlines: an “Emergency” strategy

PDVBV

SOLUTIONS THAT MATTER

- I never liked outlines: too much hassle, but..
  - I Discovered their use on upgrades from 9 to 10.


- IF you can afford to do this (1 hr work):
  - Get outlines of all major queries when “Good”
  - Then either lock m in place.
  - Or keep m for use and reference when needed.

- This is SQL-Plan management by another name..
  - But I’m not on 11g yet…

# Hints… Necessary Evil



- Hints are EVIL
  - Gremlins, time-bombs (job security?)
- Maybe: on GTTs
  - Dynamic sampling (tt, 1)  (c/o JLewis)
- Possibly on “The-Cast-table” in PL/SQL
  - Tell CBO what is in your array

- You can “catch” a hint from dbms_xplan…
  - (demo_outline_hint.sql – how to get in trouble…)

# Outlines (and plans)



- There is a lot to outlines and “plan management”
- It it becomes (complicated) tweaking….
- It is probably too … complicated.

•Image ?

- Think of an easier way !
  - Good indexing – good INDEXING…
  - Shoot qrys (but cant do that forever)
  - Try New, better stats
  - Outlines – just in case



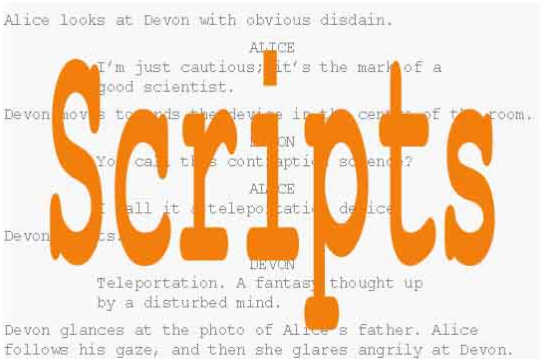
Time.. You cant tweak CBO forever, no matter how much Oracle wants you to…
Did I mention “good Indexing” ?

# Keep the clipart

PDVBV

SOLUTIONS THAT MATTER

- **Frozen Plans.**



# Keep the clipart



- Thu, 18Feb, 11:15 (one one one five...)
- Hall …,
- the SIMPLE approach to Indexing and CBO

- Same time: many real celebrities presenting



User Managed: Flexible, but "You run it" Dataguard: covers almost any-outage, user-errors, machine-failure, site-disasters, and great for "other usages"