GoToDBA

From 4 Minutes to 8 Seconds in an Hour


Liron Amitzi


liron@gotodba.com


@amitzil


https://gotodba.com

About Me

- Liron Amitzi
- Oracle DBA since 1998 (and Oracle 7)
- Database consultant since 2002
- Oracle ACE
- An independent senior DB consultant
- Recently moved to Vancouver, BC
- BCOUG president

GoToDBA

500+ Technical Experts Helping Peers Globally

ORACLE ACE Program

ORACLE ACE Director

ORACLE ACE

ORACLE ACE Associate

---

3 Membership Tiers

- Oracle ACE Director
- Oracle ACE
- Oracle ACE Associate

bit.ly/OracleACEProgram

Connect:

oracle-ace_ww@oracle.com

Facebook.com/oracleaces

@oracleace

Oracle Groundbreakers

Nominate yourself or someone you know: acenomination.oracle.com

GoToDBA

This session is based on a true story

table and column names have been changed to protect the innocent

What are we Talking About?

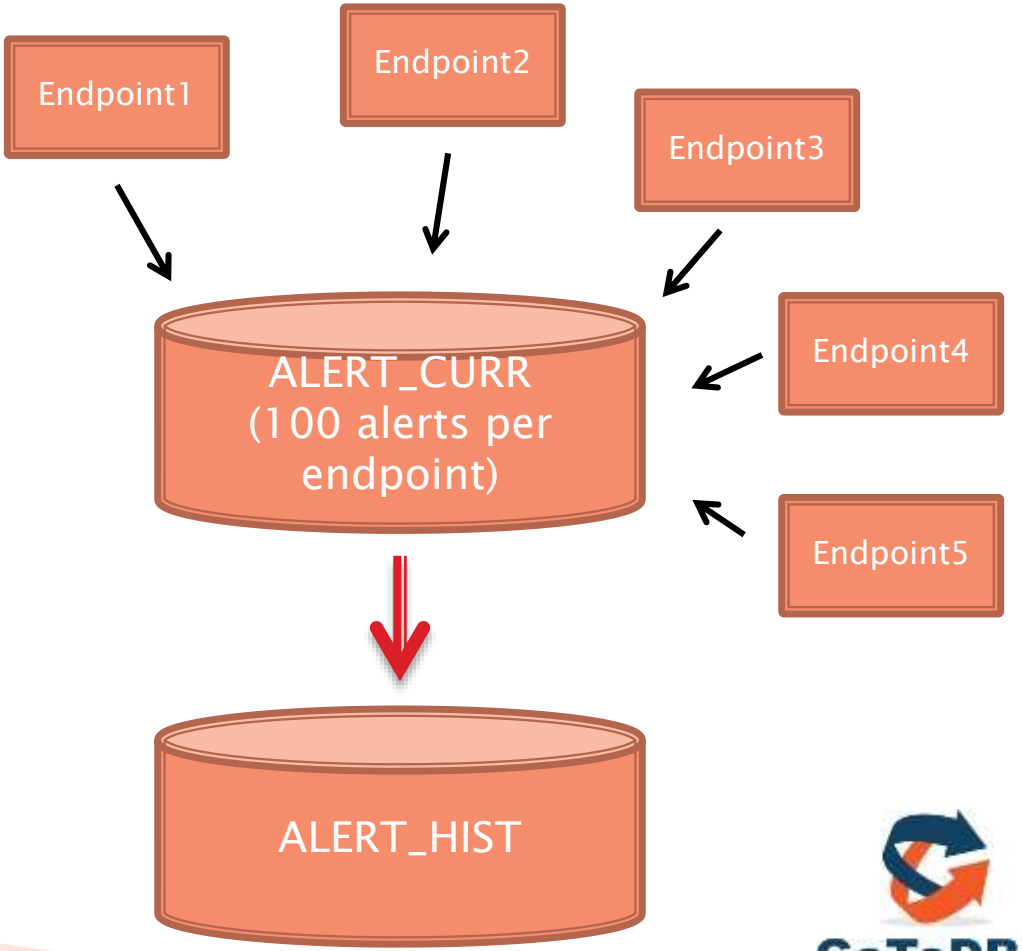
- This happened more than 10 years ago, but is still relevant
- A dashboard query took about 4 minutes and timed out
- Spoiler alert - at the end of the process the query took 8 seconds

What does SQL Tuning Include?

- Understanding the application design - but why?
- Understanding the query logic - but why?
- Understanding the query code - OK I get that one
- Understanding what Oracle does - sounds reasonable
- Trying to help Oracle do something better - how?

The Design

- Monitoring system
- Endpoints are sending many alerts
- After 100 alerts per endpoint old alerts are moved to ALERT_HIST



GoToDBA

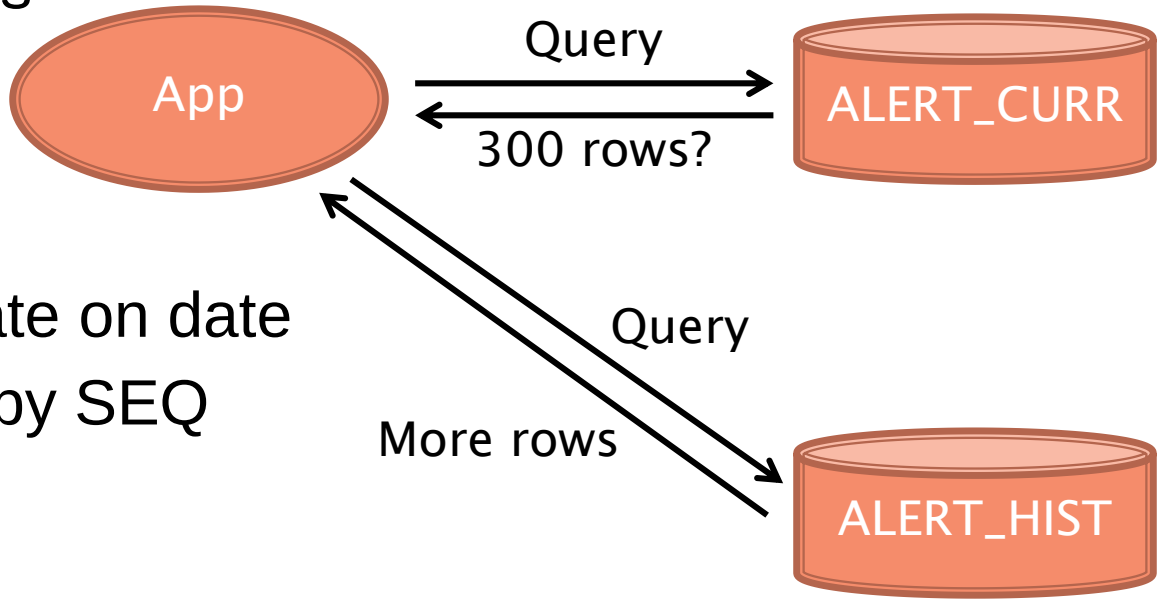
More About the Design

- Alerts were coming quickly, so they added a sequence to ensure order
- PK was SEQ, date and endpoint_id
- We couldn't change the base design (history structure) but were allowed to change anything else
- Partitions could be great here, but this was SE...

SEQ	Date	Endpoint	...
1	10-OCT-18 2:00:00	1	
2	10-OCT-18 2:00:00	1	
3	10-OCT-18 2:00:00	2	
4	10-OCT-18 2:00:02	2	
5	10-OCT-18 2:00:02	2	

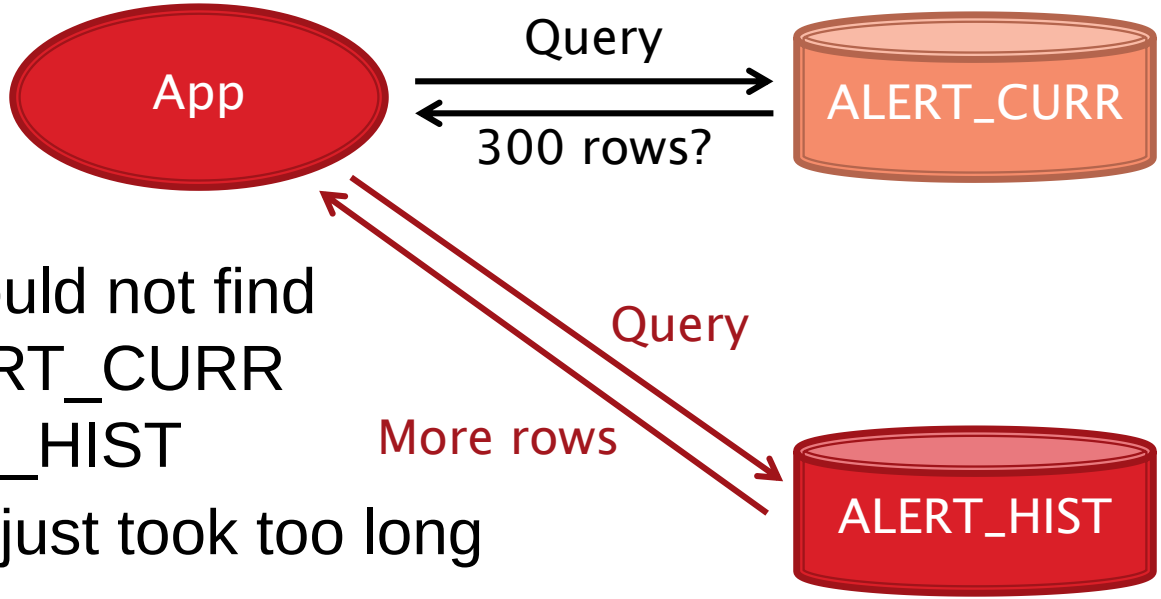
The Query Logic

- Dashboard shows 300 rows
- Users could add predicates
- Common predicate on date
- Results ordered by SEQ



GoToDBA

The Problem



- When the app could not find 300 rows in ALERT_CURR it queried ALERT_HIST
- In many cases it just took too long

GoToDBA

The Query Code (Original)

```
select * from
     (select *
      from <tab>
      where <filter>
      order by seq desc
    )
where rownum<=300;
```

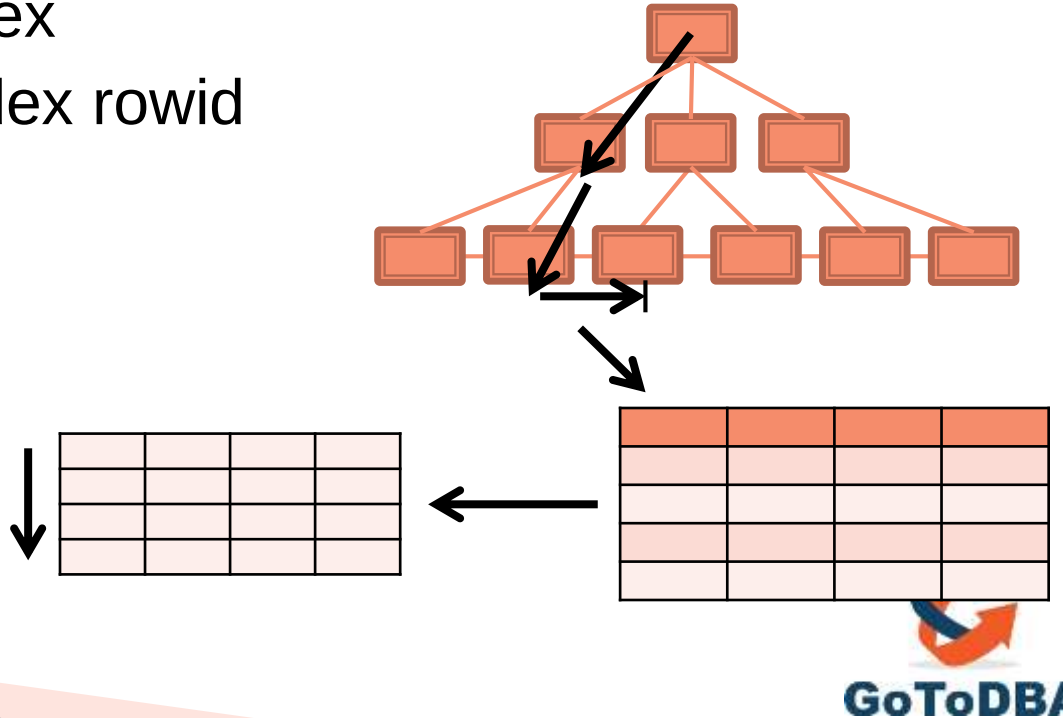
The SEQ Column

- Was created only to make sure the order is preserved
- Today I would use timestamp
- Query often had a predicate on the date and order by SEQ
- We had PK (SEQ, date, endpoint_id) and a regular index (date)
- When we use predicate on date, what will Oracle do?

SEQ Column - Index Usage #1

Range index scan on the date index:

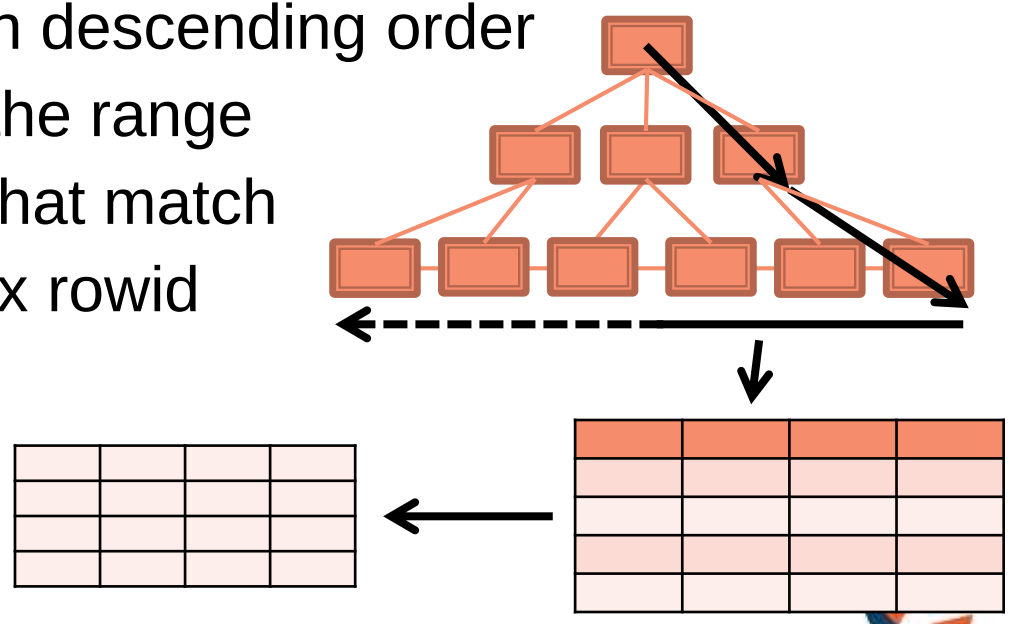
1. Filter rows by the index
2. Fetch the rows by index rowid
3. Sort the result set
4. Return first 300



SEQ Column - Index Usage #2

Full index scan on the primary key:

1. Scan the entire index in descending order
2. Check if the date is in the range
3. Get the first 300 rows that match
4. Fetch the rows by index rowid



GoToDBA

Changing the PK

- Oracle decided to use the PK to scan the SEQ column ordered
- Since SEQ value is not important, we changed the PK:
  - The old PK was SEQ, date, endpoint_id
  - The new PK was date, SEQ, endpoint_id
- We also added the date to the order by
- That way Oracle used the index for both predicate and sort

The Query Code (New PK)

```
select * from
     (select *
      from <tab>
      where <filter>
      order by date desc, seq desc
    )
where rownum<=300;
```

A Bug in the Logic!

- Wait a second!
- The design is based on numbers per endpoint, while the dashboard queries the latest
- There is a bug if one endpoint send many alerts while the others don't
- Dashboard might show wrong data as new data is already in ALERT_HIST

Fixing the Bug

- ALERT_HIST can contain alerts that are newer than some alerts in ALERT_CURR
- We had to query ALERT_HIST every time, which made the problem even worse!

The Query Code (Step 1)

```
select * from
    (select * from
        (select * from alert_curr
         union all
         select * from alert_hist
        )
     where <filter>
     order by date desc, seq desc
    )
where rownum<=300;
```

Amount of Rows

- After fixing all of this, the query was still slow...
- The indexes were not being used optimally
- The union and order by resulted in a lot of work on Oracle's side
- Returning 300 rows after sort requires a full sort operation
- Any ideas?

Amount of Rows - Solution

- We realized that we need 300 rows in the end
- That's 300 from the first table, or 300 from the second, or any combination of the two
- Let's limit each table to 300 rows efficiently and then take the top 300
- Makes sense?

The Query Code (Step 2)

```
select * from
 (select * from
    ((select * from
      (select * from alert_curr where <filter> order by date*,seq*)
      where rownum<=300)
     union all
     (select * from
      (select * from alert_hist where <filter> order by date*,seq*)
      where rownum<=300)
    )
    where <filter> order by date*, seq*)
   where rownum<=300;
```

* - desc order

GoToDBA

Results

- The query that took 4 minutes at the beginning now took about 8 seconds
- Index range scan was very efficient (used for both date predicate and order by)
- There is a single order by operation of only 600 rows

Summary

- A successful project and a very satisfied customer
- We do need to understand the logic
- We do need cooperation from the developers, we are not magicians
- Without understanding the system we could not:
  - Find and fix the bug in the logic
  - Change the PK (what if there was a reason for SEQ to be first?)

Q&A

Liron Amitzi

liron@gotodba.com

@amitzil

https://gotodba.com