Avoid „Horror Queries“

Keep Your SQL Simple and Fast

Dani Schnider

@dani_schnider

danischnider.wordpress.com

Dani Schnider

- Senior Principal Consultant at Trivadis AG in Glattbrugg/Zurich
- Trainer of several Trivadis courses
- Co-Author of Books “Data Warehousing mit Oracle” and “Data Warehouse Blueprints”
- Oracle ACE

@dani_schnider

danischnider.wordpress.com

ORACLE ACE

BASEL | BERN | BRUGG | BUCHAREST | COPENHAGEN | DÜSSELDORF | FRANKFURT A.M. | FREIBURG I. BR.
GENEVA | HAMBURG | LAUSANNE | MANNHEIM | MUNICH | STUTTGART | VIENNA | ZURICH

trivadis

Note on this presentation

trivadis

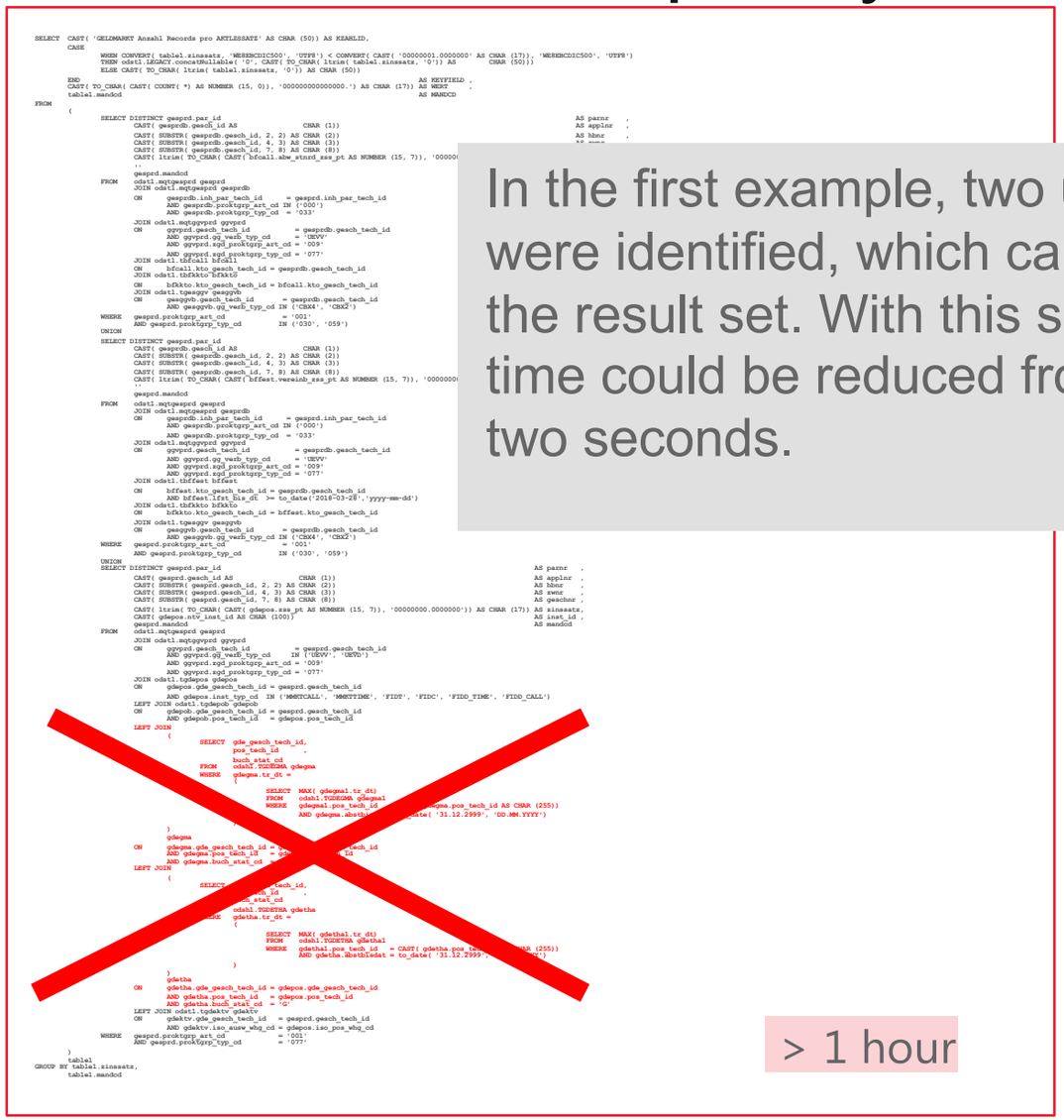
- The session presents five examples of complex SQL statements that are simplified step-by-step to improve performance. The slides consist of numerous individual steps that are difficult to understand without a "soundtrack".
- For this reason, many slides have been omitted from this download version. Instead, a brief explanation is given for each example of the actions used to improve query performance.

trivadis

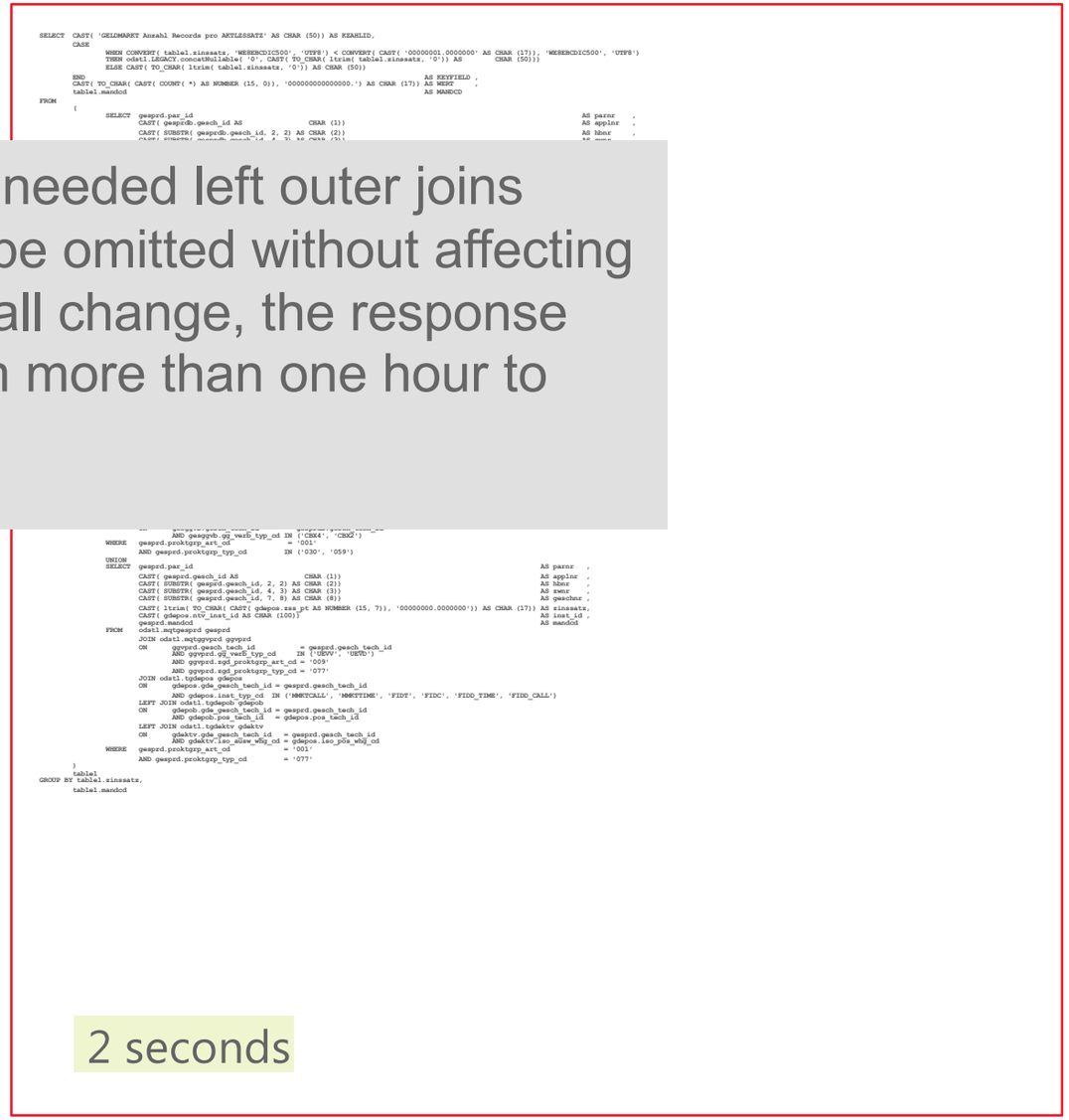
Reduction of Complexity

Reduction of Complexity

trivadis



> 1 hour



2 seconds

In the first example, two unneeded left outer joins were identified, which can be omitted without affecting the result set. With this small change, the response time could be reduced from more than one hour to two seconds.

trivadis

Performance Tuning without Database Access

```
    SELECT S1.DEPTNO,
           S1.JOB,
           MIN (S1.EMPNO) AS EMPNO,
           LTRIM (MAX (SYS_CONNECT_BY_PATH (S1.EMPNO, ';'))
                  KEEP (DENSE_RANK LAST ORDER BY S1.curr), ';') AS EMPFAENGERLISTE
      FROM (SELECT DEPTNO,
                   JOB,
                   EMPNO,
                   ROW_NUMBER () OVER (PARTITION BY DEPTNO, JOB
                                       ORDER BY EMPNO) AS curr,
                   ROW_NUMBER () OVER (PARTITION BY DEPTNO, JOB
                                       ORDER BY EMPNO) - 1 AS prev
              FROM EMP
             WHERE DEPTNO > 0) S1
  GROUP BY S1.DEPTNO, S1.JOB
CONNECT BY     S1.prev = PRIOR S1.curr
           AND S1.DEPTNO = PRIOR S1.DEPTNO
START WITH curr = 1

SELECT DEPTNO,
       JOB,
       MIN (EMPNO) AS EMPNO,
       LISTAGG(EMPNO, ';')
             WITHIN GROUP (ORDER BY EMPNO)
             AS EMPFAENGERLISTE
  FROM EMP
 WHERE DEPTNO > 0
GROUP BY DEPTNO, JOB
```

In the second example, a complex query was replaced by a simpler statement using LISTAGG, which runs much faster and is easier to understand. This example is described in detail here: https://danischnider.wordpress.com/2018/04/03/keep-your-sql-simple-and-fast/

trivadis

An Expensive Attribute

Simple Query on a View

trivadis

```
SELECT state_filing_number,
       legal_entity_name,
       toi_desc,
       sub_toi_desc,
       state_full_desc,
       project_lead,
       state_lead,
       pr_state_status,
       response_date,
       form_type_desc,
       state_tracking_num,
       serff_track_nbr,
       circular_number_txt,
       request_effective_date,
       effective_date,
       status_date,
       state_req_uid
  FROM rah.vw_state_tab_data
WHERE legal_entity_id IN (6)
ORDER BY state_desc;
```

The calculation of an attribute STATUS_DATE in a view became very inefficient due to several nested subselects. Using a subquery factoring clause (WITH statement) and a CASE statement, the response time has been reduced from several minutes to a few seconds.

trivadis

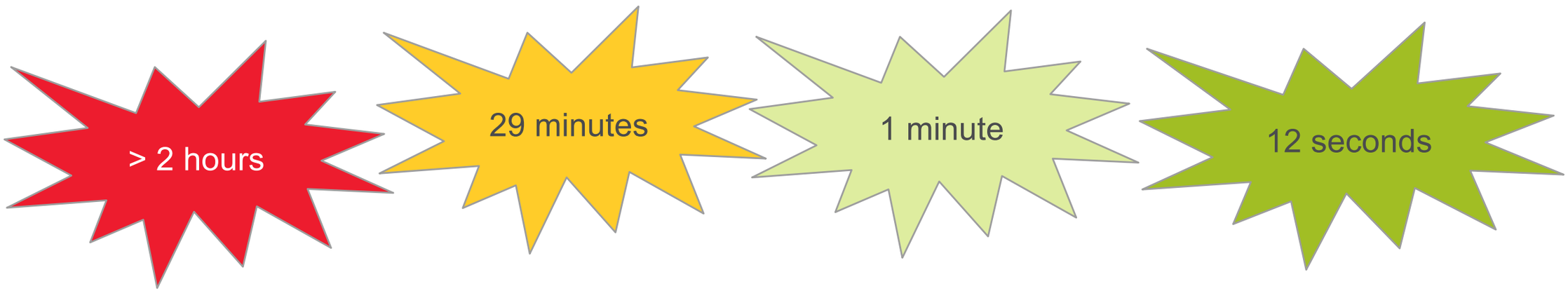
Make it Faster Step by Step

Query on daily accounting data too slow

trivadis

By using several performance features in two nested views, it was possible to speed up the frequent query for a single date step by step.

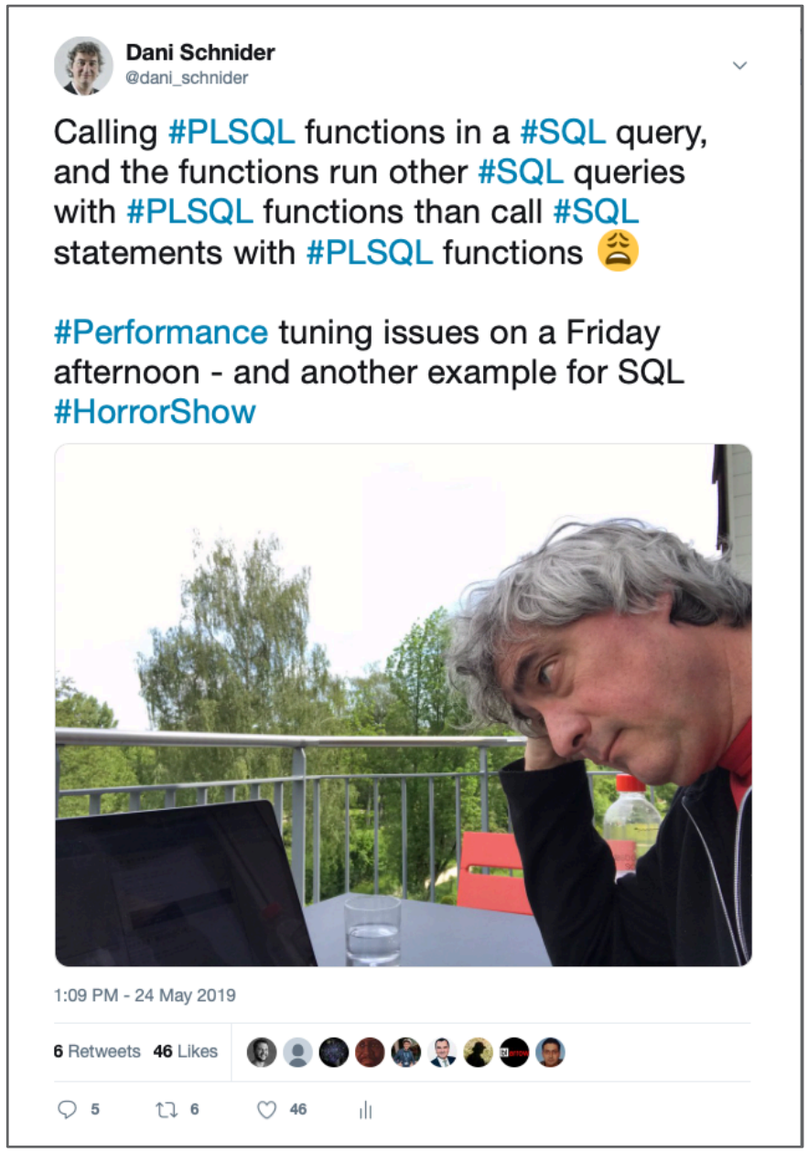
```
SELECT SECURITY_CODE, RATING_AGENCY, RATING, RATING_DATE
  FROM V_IF_SRAM_INSTR_RATINGS
 WHERE ACCOUNTING_DATE = TO_DATE('20180513','yyyymmdd');
```



trivadis

A PL/SQL Journey

trivadis



It is a very bad idea to call PL/SQL functions containing SQL statements in the SELECT or WHERE part of a SQL statement because the locally called SQL statement is executed for every row of the query. If this local SQL statement calls other PL/SQL functions, the performance of the main query is getting worse and worse.

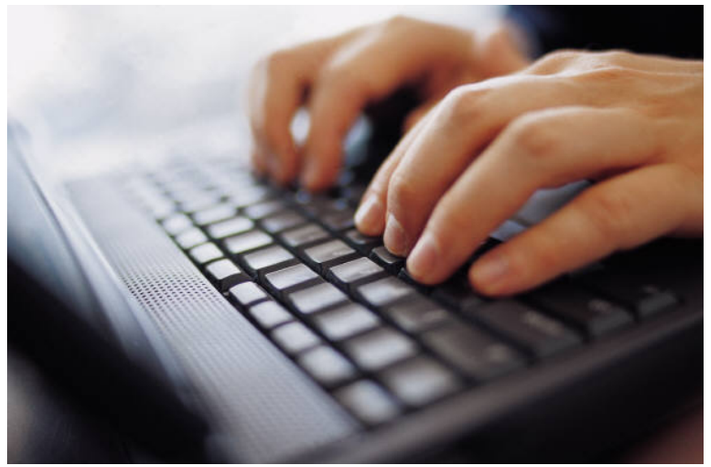
trivadis

Summary

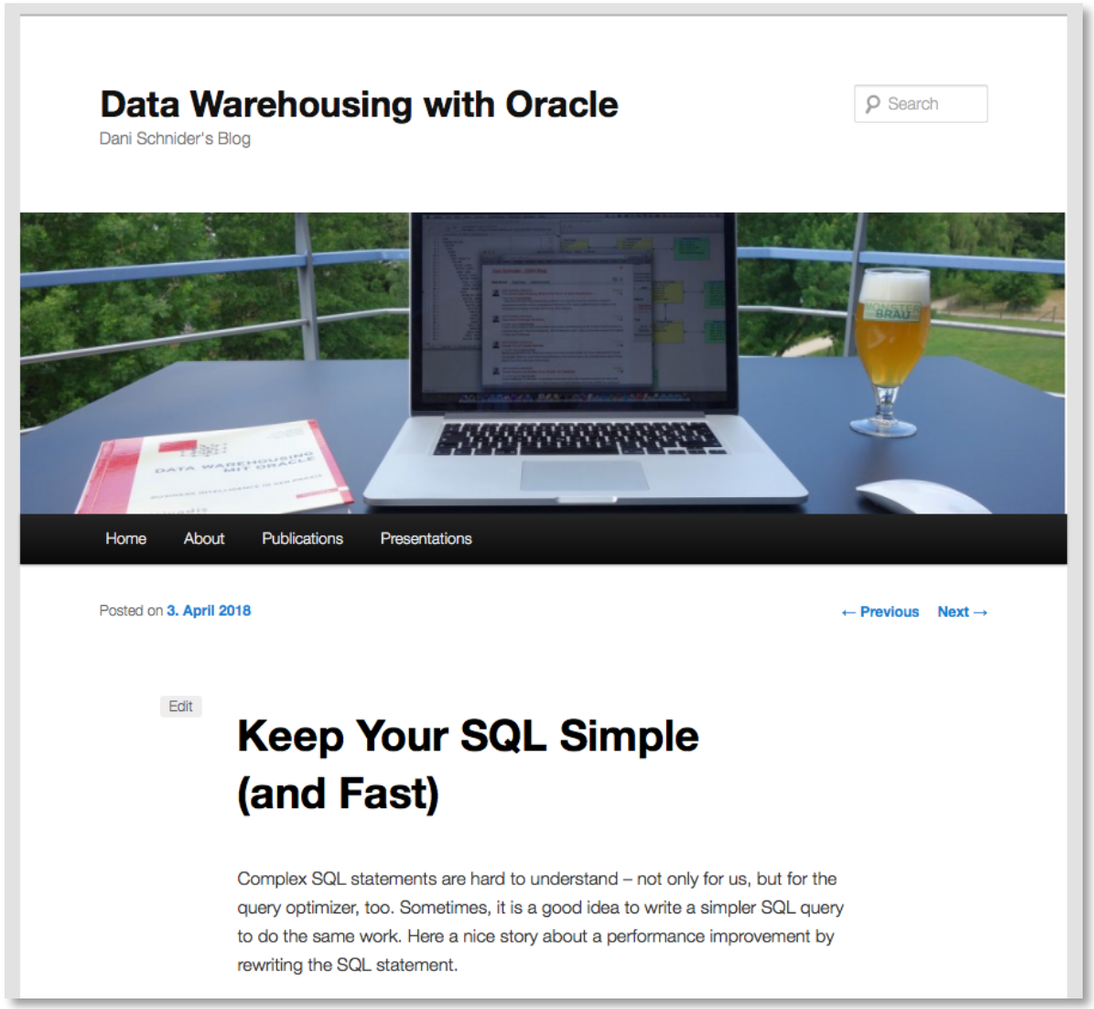
Summary

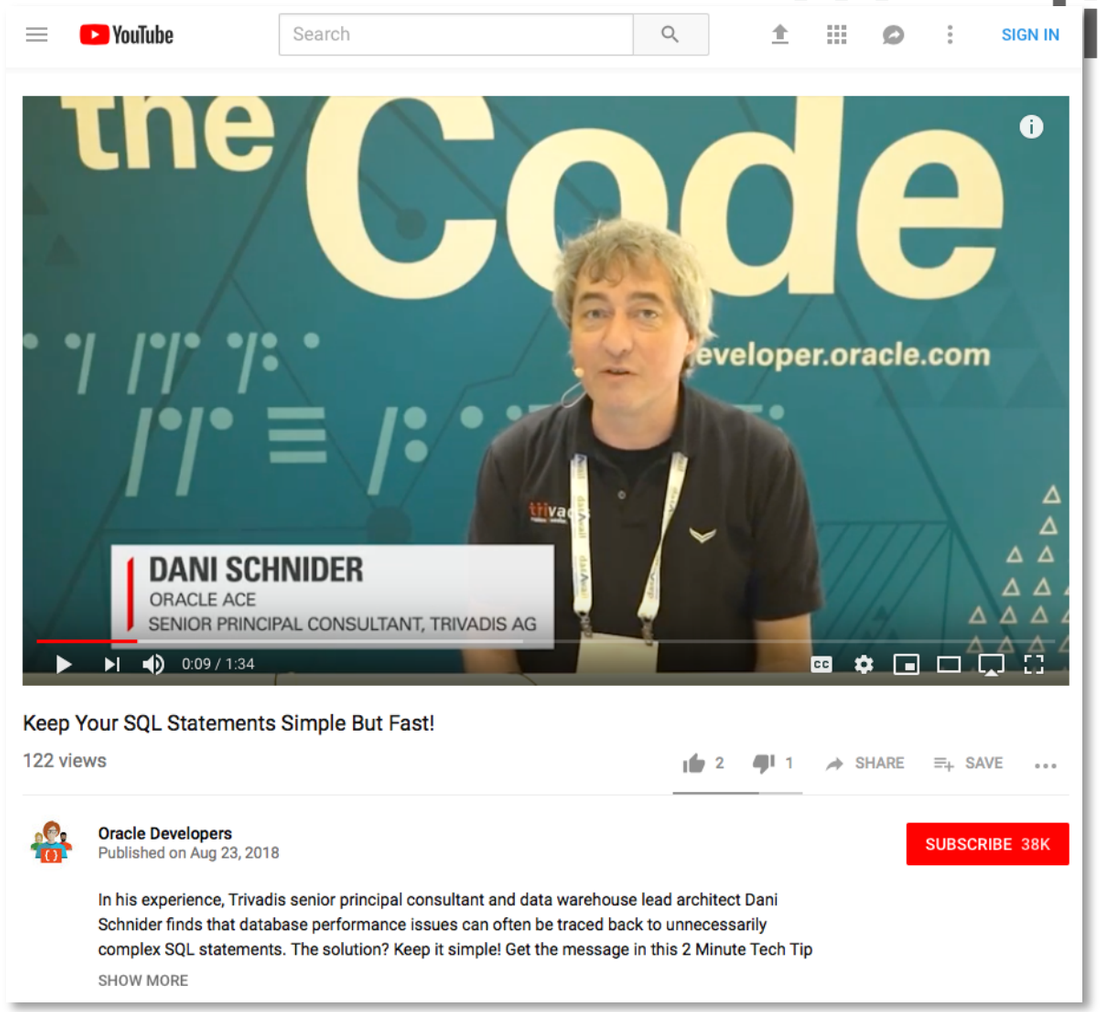
trivadis

- SQL Performance Tuning is Hard Work
  - … but effective
  - Massive performance improvements are possible
- Reduce complexitity of SQL statements
  - Filtering the data set as soon a possible
  - Avoid multiple executions
  - The easier, the better – for us and for the optimizer



Keep Your SQL Simple and Fast





https://danischnider.wordpress.com/2018/04/03/keep-your-sql-simple-and-fast/
https://youtu.be/CrJYqHtulGg