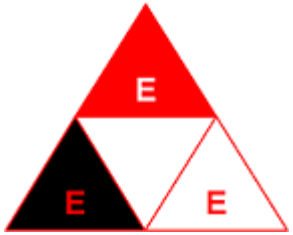
Materialized Views
&
Partition Change Tracking



Uwe Hesse

Oracle Certified Master

Senior Principal Instructor at Oracle University

Twitter handle: `@UweHesse`

Blog: `uhesse.com`

Agenda

- Brief introduction: Materialized Views & Rewrite, Refresh
- Partition Change Tracking (PCT) benefits for
  - Query Rewrite
  - Fast Refresh

The Playground

- Creation of Demo User & Base Table

```
SQL> grant dba to adam identified by adam;
SQL> connect adam/adam
SQL> alter session set workarea_size_policy=manual;
SQL> alter session set sort_area_size=1000000000;
SQL> create table sales as select rownum as id,
        mod(rownum,5) as channel_id,
        mod(rownum,1000) as cust_id,
        5000 as amount_sold,
        sysdate as time_id
        from dual connect by level<=1e7;
```

Without Materialized Views

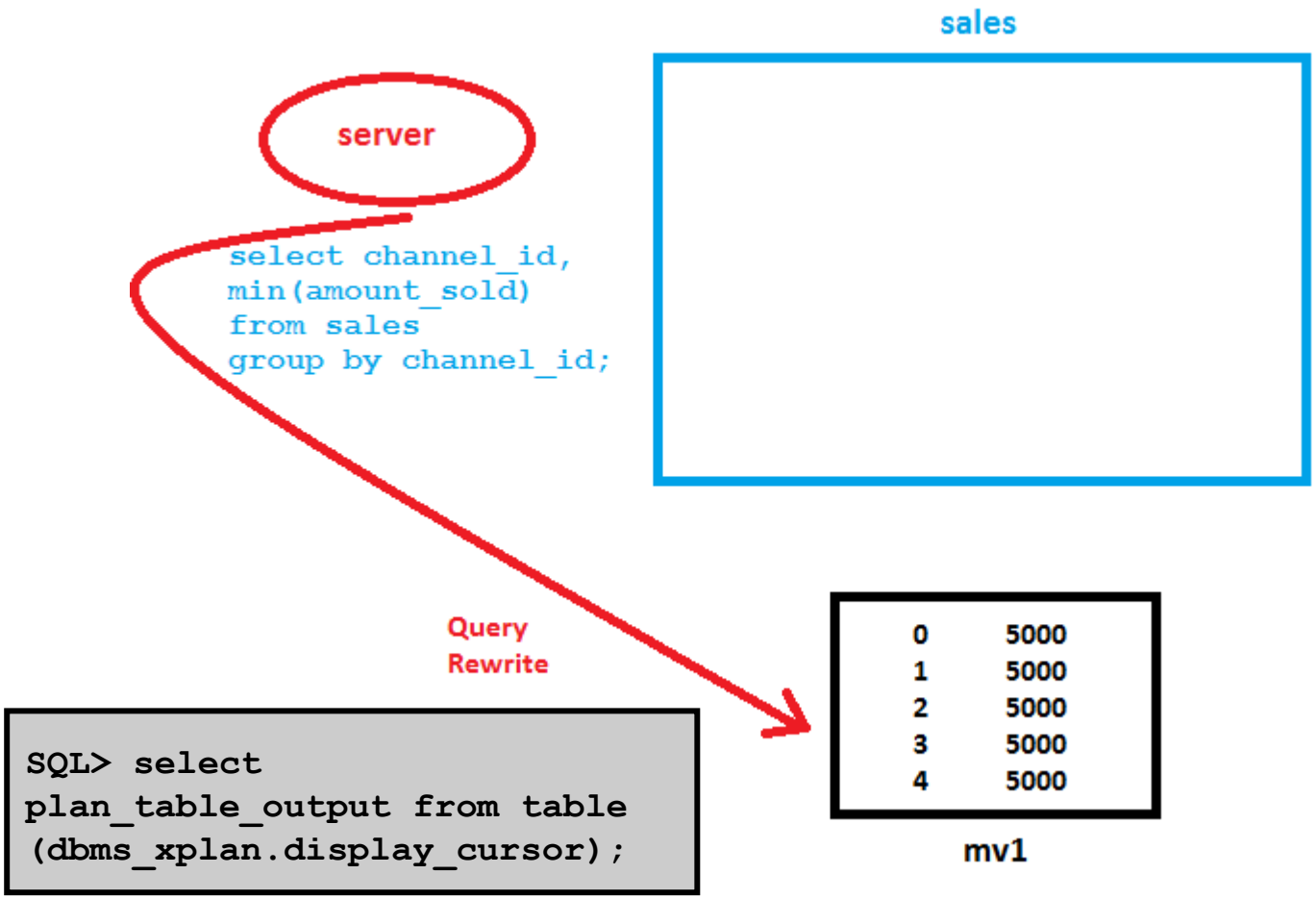
- An aggregation on the base table takes relatively long

```
SQL> select channel_id,min(amount_sold)
      from sales
      group by channel_id;

CHANNEL_ID MIN(AMOUNT_SOLD)
---------- ----------------
         1             5000
         2             5000
         4             5000
         3             5000
         0             5000

Elapsed: 00:00:02.55
```

Materialized Views speed up the query!

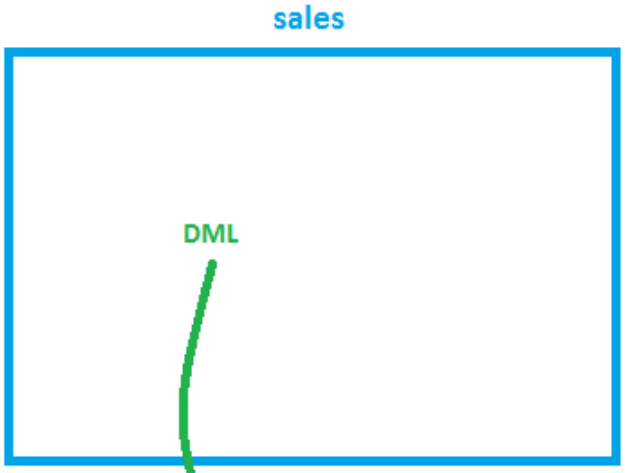
```
SQL> create materialized view mv1
       enable query rewrite as
       select channel_id,min(amount_sold)
       from sales
       group by channel_id;

SQL> select channel_id,min(amount_sold)
       from sales group by channel_id;

CHANNEL_ID MIN(AMOUNT_SOLD)
---------- ----------------
         1             5000
         2             5000
         4             5000
         3             5000
         0             5000

Elapsed: 00:00:00.08
```

Transparent Query Rewrite: Confirmation



Mvs get „stale“ upon DML on the Base Tables

```
SQL> update sales set amount_sold=1 where rownum<2;
SQL> commit;
```




```
SQL> select
     staleness
     from
     user_mviews;

STALENESS
-------------
NEEDS_COMPILE
```

Complete Refresh enables Rewrite again

- A complete refresh is needed in this case, which takes relatively long:

```
SQL> set timing on
SQL> exec dbms_mview.refresh(list=>'MV1',method=>'C')

PL/SQL procedure successfully completed.

Elapsed: 00:00:02.72
```

Fast Refresh requires MV Logs usually

- Customized MV Query & MV Logs are usually prerequisites to enable a Fast Refresh

```
SQL> create materialized view log on sales with
      sequence, rowid (channel_id, amount_sold) including new values;

SQL> create materialized view mv1
      refresh fast
      enable query rewrite as
      select count(*),count(amount_sold),
      channel_id,min(amount_sold) from sales group by channel_id;
```

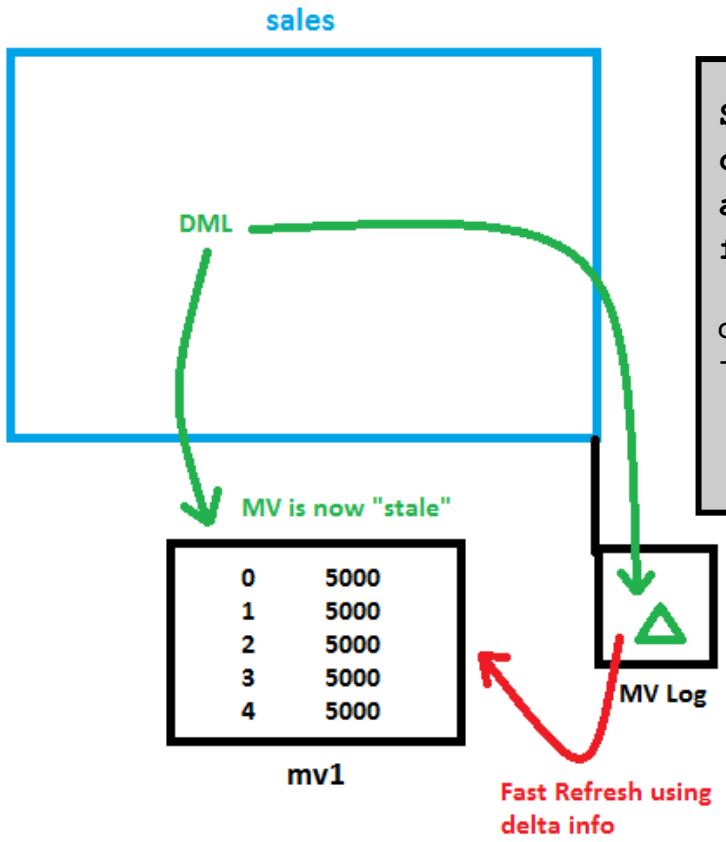
Fast Refresh needs no FTS on Base Table

```
SQL> update sales set amount_sold=0 where rownum<2;
SQL> commit;
```

1

```
SQL> begin
 dbms_mview.refresh
 (list=>'MV1',
  method=>'F');
 end;
 /

SQL> select
last_refresh_type
from user_mviews;

LAST_REF
--------
FAST
```

3




```
SQL> select
channel_id,
amount_sold
from mlog$_sales;

CHANNEL_ID AMOUNT_SOLD
---------- -----------
         1           1
         1           0
```

2

Same Data but Partitioned Base Table

```
SQL> create table sales_part
     (id number,
      channel_id number,
      cust_id number,
      amount_sold number,
      time_id date)
     partition by list (channel_id)
     (partition c0 values(0),
      partition c1 values(1),
      partition c2 values(2),
      partition c3 values(3),
      partition c4 values(4));

SQL> alter table sales_part nologging;

SQL> insert /*+ append */ into sales_part select * from sales;

10000000 rows created.
```

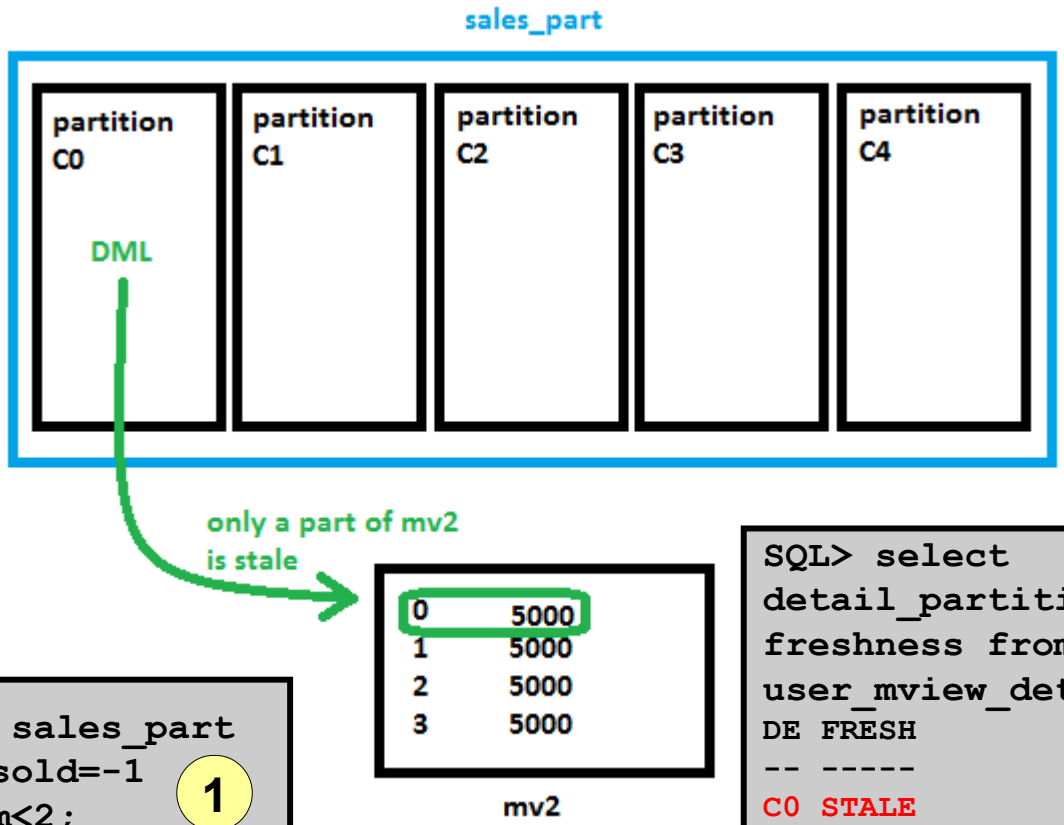
Prerequisit for PCT Benefits

- There must be a relation between the partition key of the Base Table and the query of the Materialized View

```
SQL> create materialized view mv2
      enable query rewrite as
      select channel_id, min(amount_sold)
      from sales_part group by channel_id;
```

- That column is the partition key of the Base Table sales_part

Finer granule of staleness with Partitioned Base Tables:



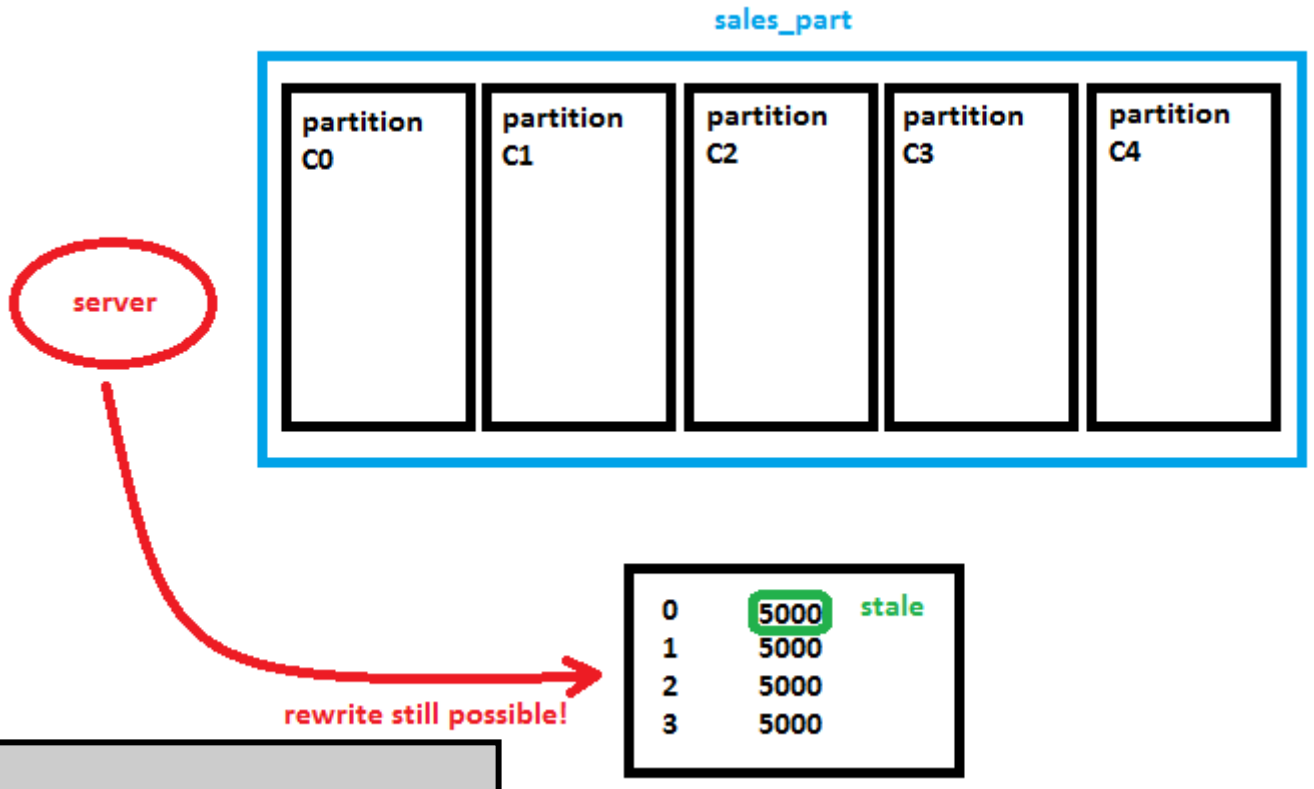

```
SQL> update sales_part
set amount_sold=-1
where rownum<2;
SQL> commit;
```

1

```
SQL> select
detail_partition_name,
freshness from
user_mview_detail_partition;
DE FRESH
-- -----
C0 STALE
C1 FRESH
C2 FRESH
C3 FRESH
C4 FRESH
```

2

PCT Query Rewrite in action:




```
SQL> select min(amount_sold)
from sales_part where
channel_id=2;
```

1

```
SQL> select plan_table_output
   from table
(dbms_xplan.display_cursor);
```

2

PCT Fast Refresh in action:

- There is no MV log on the Base Table sales_part

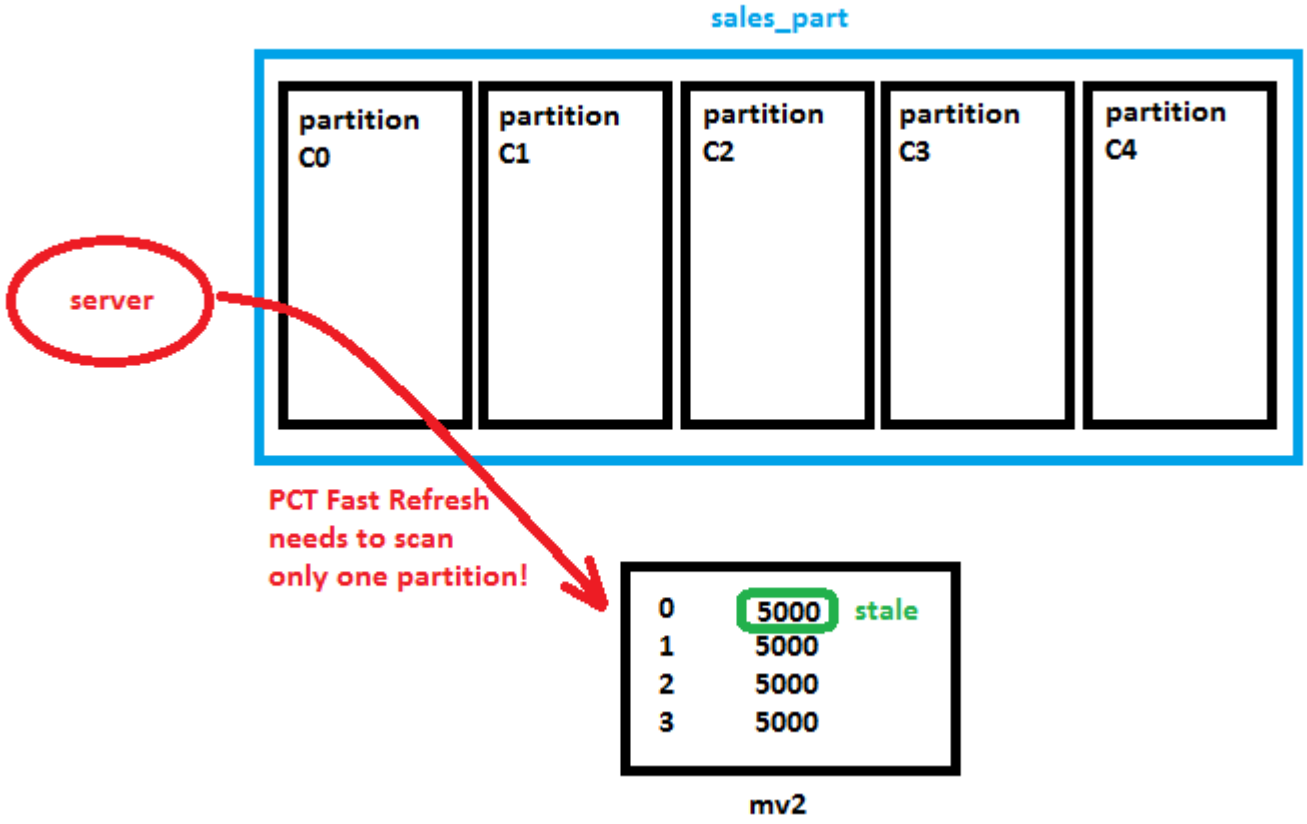
```
SQL> select master from user_mview_logs;
MASTER
------
SALES
```

- Refresh Fast is possible in spite:

```
SQL> exec dbms_mview.refresh(list=>'MV2', method=>'F')

PL/SQL procedure successfully completed.

SQL>  select last_refresh_type from user_mviews where mview_name='MV2';

LAST_REF
--------
FAST_PCT
```

PCT Fast Refresh: No FTS on Base Table

- This refresh is about 5 times faster than a Complete Refresh here



Thank you for your attention!

Any questions or remarks?