
Cost Based Query Transformations

By

Riyaj Shamsudeen



your database maestros

www.pythian.com

Support Centers in :
North America | South America | Europe
Middle East | Asia | Australia

Managed services for:
Oracle | SQL Server | MySQL

Who am I?

- 16 years using Oracle products
- Over 15 years as Oracle DBA
- Certified DBA versions 7.0,7.3,8,8i &9i
- Specializes in performance tuning, Internals and E-business suite
- Currently consulting for The Pythian Group
- OakTable member
- Email: rshamsud at gmail.com or my blog:
<http://orainternals.wordpress.com>



Warning

- Concepts discussed in this paper are not documented completely by Oracle Corporation.
- Enough care has been taken to improve accuracy, by probing with test cases.
- But, possibility of inaccuracies still exist.

Version & command used

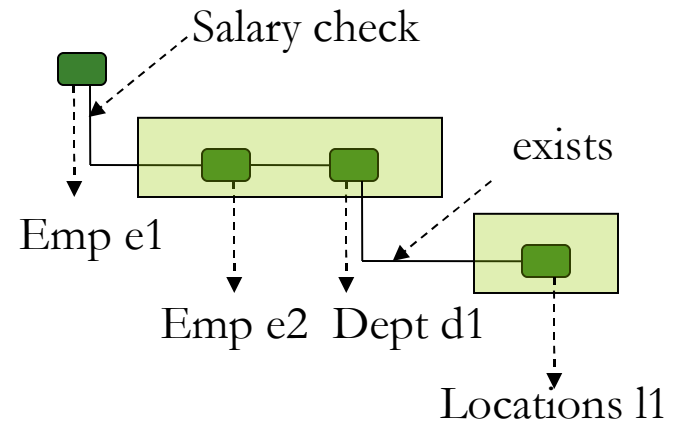
- Windows XP, Oracle 11.1.0.6
- Not all features were enabled in 10gR2, even though CBQT was introduced in 10g.
- Command to generate 10053 trace file:
Alter session set events '10053 trace name context forever, level 1';

Agenda

- What is transformation & Why do we need to cost?
- Explanation of query.
- Review concepts behind Query transformations and Map them back to 10053 trace.
- Parameters affecting CBQT behavior.
- Questions.

Query

```
Select /*+ qb_name (e1_outer) */ * from
emp e1 where
salary >
(select /*+ qb_name (e2_inner) */
avg(salary) from
emp e2, dept d1
where e1.dept_id = e2.dept_id and
e2.dept_id = d1.dept_id and
exists
(select /*+ qb_name (l1_inner) */
1 from locations l1
where l1.location_id=d1.location_id ))
and e1.hire_date > sysdate - (10*365)
```



Qb_name improves readability
Of trace files. User friendly
Names for query blocks instead
Of system generated names

What is a transformation?

Transformed query:

```
Select /*+ qb_name (e1_outer) */ * from  
emp e1,
```

```
(select /*+ qb_name (e2_inner) */  
d1.dept_id, avg(salary) avg_salary  
from emp e2, dept d1  
where e2.dept_id = d1.dept_id and  
exists  
(select /*+ qb_name (l1_inner) */  
1 from locations l1  
where l1.location_id=d1.location_id )  
group by dept_id ) gbp1  
where
```

```
e1.hire_date > sysdate - (10*365) and  
e1. salary > gbp1.avg_salary and  
E1.dept_id = gbp1.dept_id
```

Aggregation performed before
Join, using group by placement
Technique here.

Queries are transformed using
Various techniques such as

- Subquery Unnesting
- Group by placement
- Predicate Push down

etc..

Why costing transformations?

Comparing these two costs..

Say, cost of 100 for correlated subquery

cost of aggregation 10,000, then

$N \times 100$

$10000 + N \times J$, assuming join cost = 0.1

For $N=1$ row, cost is 100

For $N=100$ rows, cost is 10,000

For $N=10000$ rows, cost is 1,000,000

For $N=1$ row, cost is 10,000.1

For $N=100$ rows, cost is 10,010

For $N=10000$ rows, cost is 11,000

Select /*+ qb_name (e1_outer) */ * from
emp e1 where
e1.hire_date > sysdate - (10*365) and
salary >

(select /*+ qb_name (e2_inner) */
avg(salary) from
emp e2, dept d1
where e1.dept_id = e2.dept_id and
e2.dept_id = d1.dept_id and
exists

(select /*+ qb_name (l1_inner) */
1 from locations l1
where l1.location_id=d1.location_id))

Cost=100

Select /*+ qb_name (e1_outer) */ * from
emp e1,

(select /*+ qb_name (e2_inner) */
dept_id, avg(salary) avg_salary
from emp e2, dept d1
where e1.dept_id = e2.dept_id and
e2.dept_id = d1.dept_id and
exists

(select /*+ qb_name (l1_inner) */
1 from locations l1
where l1.location_id=d1.location_id)
group by dept_id) gbp1

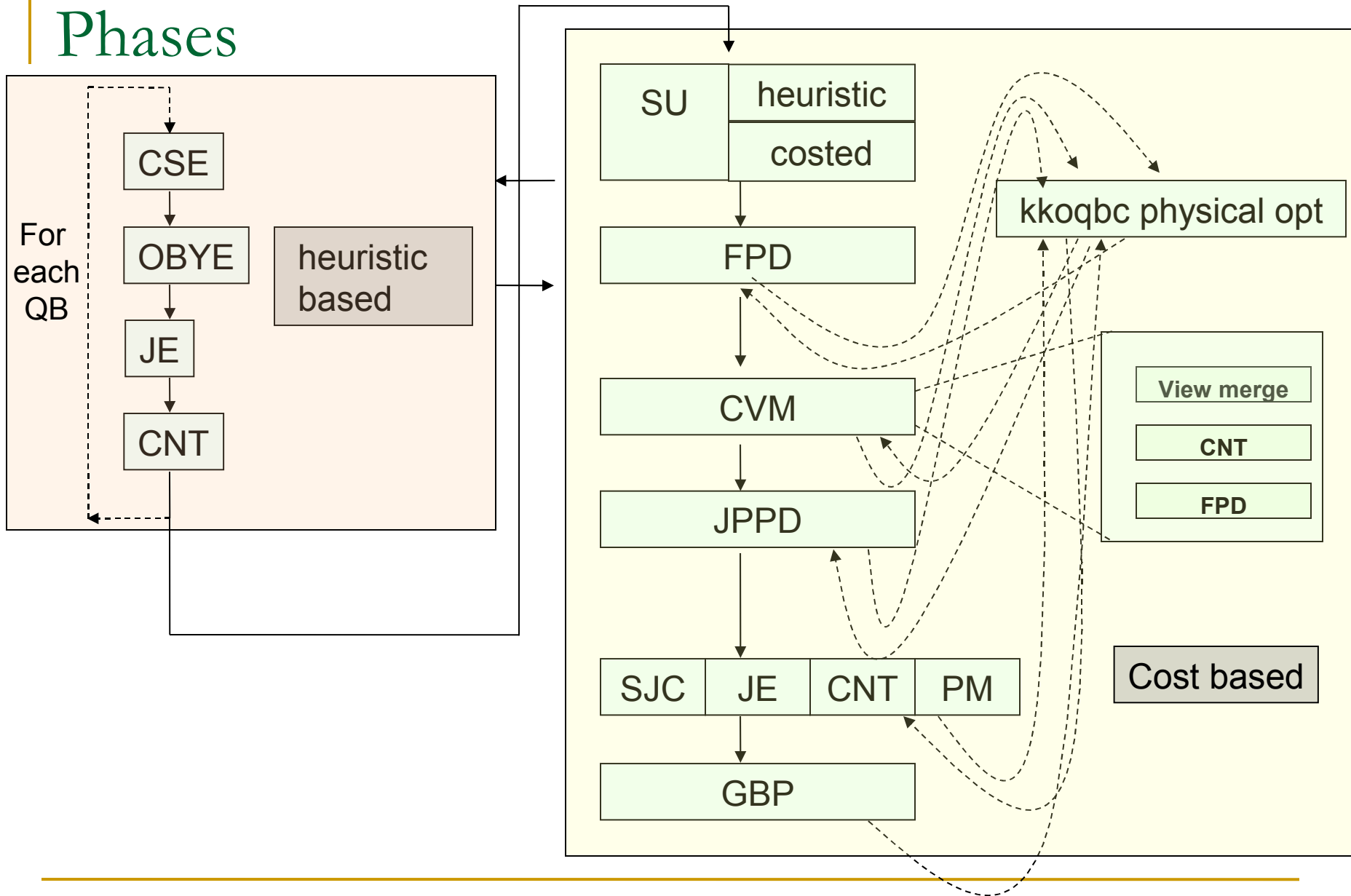
where
e1.hire_date > sysdate - (10*365) and
e1.salary > gbp1.avg_salary

Cost=10000

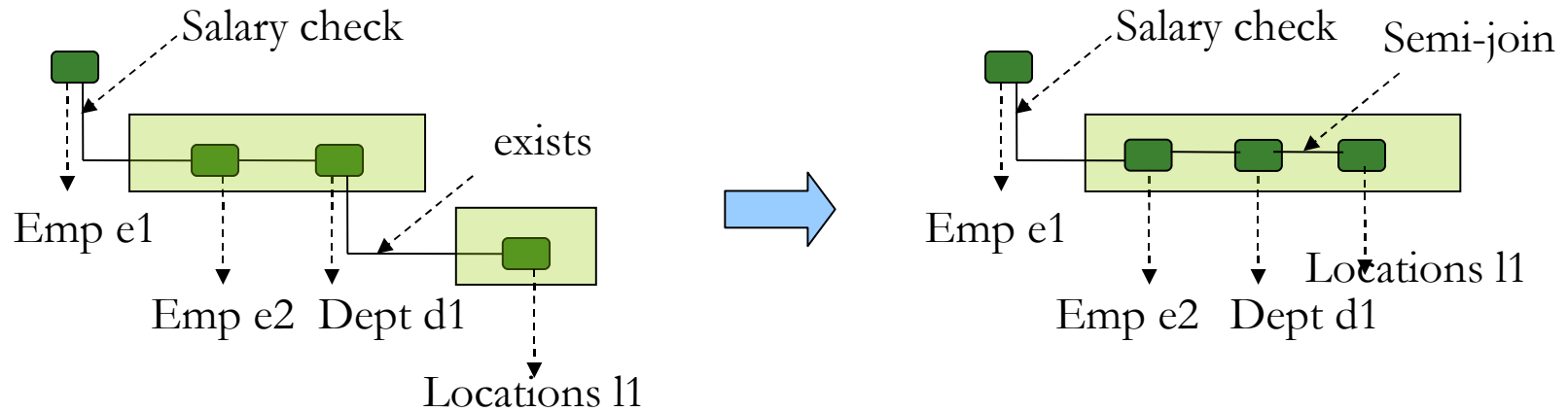
Why costing transformations?

- What if outer subquery is very selective and returns very few rows ?
- What if inner table has many rows and cost of aggregation is high ?
- What if inner subquery does not have selective indices ?
- What if access to locations table is not so optimal ?
- Point is that, equation is very complicated in real life.
- Most of transformed queries need to be costed to find optimal plan
- And, there are some transformations that does not need costing..

Phases



SU: Exists to Semi Join



Subquery unnest

'exists' sub-query for locations
unnested as a semi-join

Registered qb: **SEL\$D72FB22B** 0x21ee71cd (**SUBQUERY UNNEST E2_INNER; L1_INNER**)

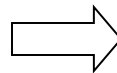
signature (): qb_name=SEL\$D72FB22B nbfros=3 flg=0

fro(0): flg=0 objn=71162 hint_alias="D1"@E2_INNER"

fro(1): flg=0 objn=71164 hint_alias="E2"@E2_INNER"

fro(2): flg=0 objn=71160 hint_alias="L1"@L1_INNER"

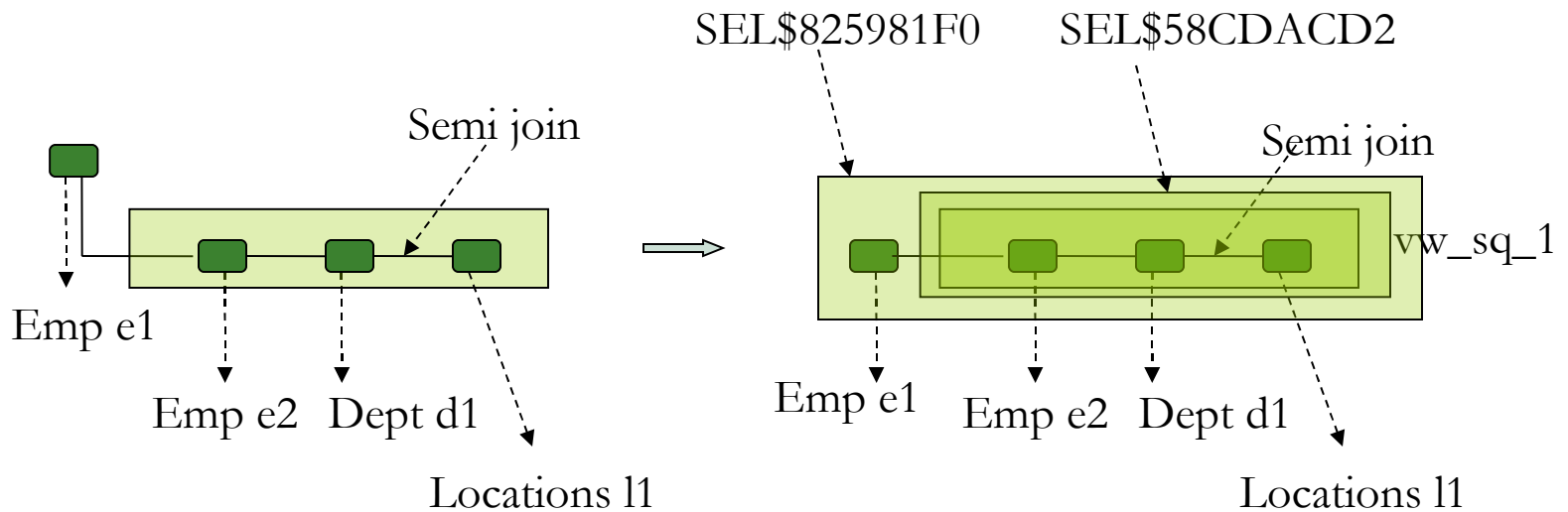
```
select /*+ qb_name (e1_outer) */ * from
emp e1 where
salary >
(select /*+ qb_name (e2_inner) */ avg(salary)
from emp e2, dept d1
where e1.dept_id = e2.dept_id and
e2.dept_id = d1.dept_id and
exists
(select /*+ qb_name (l1_inner) */ 1
from locations l1
where l1.location_id=d1.location_id))
and e1.hire_date > sysdate - (10*365)
```



```
select /*+ qb_name (e1_outer) */ * from
emp e1 where
salary >
(select /*+ qb_name (e2_inner) */
avg(salary) from
emp e2, dept d1, locations l1
where e1.dept_id = e2.dept_id and
e2.dept_id = d1.dept_id and
l1.location_id S= d1.location_id)
and e1.hire_date > sysdate - (10*365)
```

Subquery unnest

(SUBQUERY UNNEST SEL\$4ADFCC1B; SEL\$D72FB22B)



Subquery unnest

These two QBs converted to equi-join, by unnesting them.

```
signature (): qb_name=SEL$4ADFCC1B nbfros=2 flg=0  
fro(0): flg=0 objn=71164 hint_alias="E1"@ "E1_OUTER"  
fro(1): flg=5 objn=0 hint_alias="VW_SQ_1"@ "SEL$4ADFCC1B"
```

```
Registered qb: SEL$825981F0 0x21ee5968 (SUBQUERY UNNEST SEL$4ADFCC1B; SEL$D72FB22B)
```

```
select /*+ qb_name (e1_outer) */ * from  
emp e1 where  
salary >  
(select /*+ qb_name (e2_inner) */  
avg(salary) from  
emp e2, dept d1, locations l1  
where e1.dept_id = e2.dept_id and  
e2.dept_id = d1.dept_id and  
l1.location_id S= d1.location_id )  
and e1.hire_date > sysdate - (10*365)
```

```
select /*+ qb_name (e1_outer) */ e1.* from  
emp e1,  
(select /*+ qb_name (e2_inner) */  
avg(salary) avg1,  
e2.dept_id item_0  
from emp e2, dept d1, locations l1  
where e2.dept_id = d1.dept_id and  
l1.location_id S= d1.location_id  
group by e2.dept_id ) vw_sq_1  
where e1.salary > vw_sq_1.avg1  
and e1.hire_date > sysdate - (10*365)  
and e1.dept_id = vw_sq_1.item_0
```

Costing Query blocks

Cost annotations are stored

```
Trying or-Expansion on query block SEL$58CDACD2 (#2)
Transfer Optimizer annotations for query block SEL$58CDACD2 (#2)
Final cost for query block SEL$58CDACD2 (#2) - All Rows Plan:
  Best join order: 2
  Cost: 489.7812 Degree: 1 Card: 99900.0000 Bytes: 2497500
  Resc: 489.7812 Resc_io: 482.0000 Resc_cpu: 172360597
  Resp: 489.7812 Resp_io: 482.0000 Resc_cpu: 172360597
kkoqbc-subheap (delete addr=0x07A8C150, in-use=28672, alloc=31212)
kkoqbc-end:
```

Indicating end of kkoqbc call.

Reuse of optimized query block [2]

Optimized query blocks tracked in this structure.

Query blocks checked in this structure before physically optimizing it and reused if query block found with matching signature and attributes.

QB identifier	State	QB type	Cost	Cardinality	selectivity	Pointer
1. SEL\$58CDACD2_00000202_2 2. SEL\$825981F0_00000000_0			489 663			

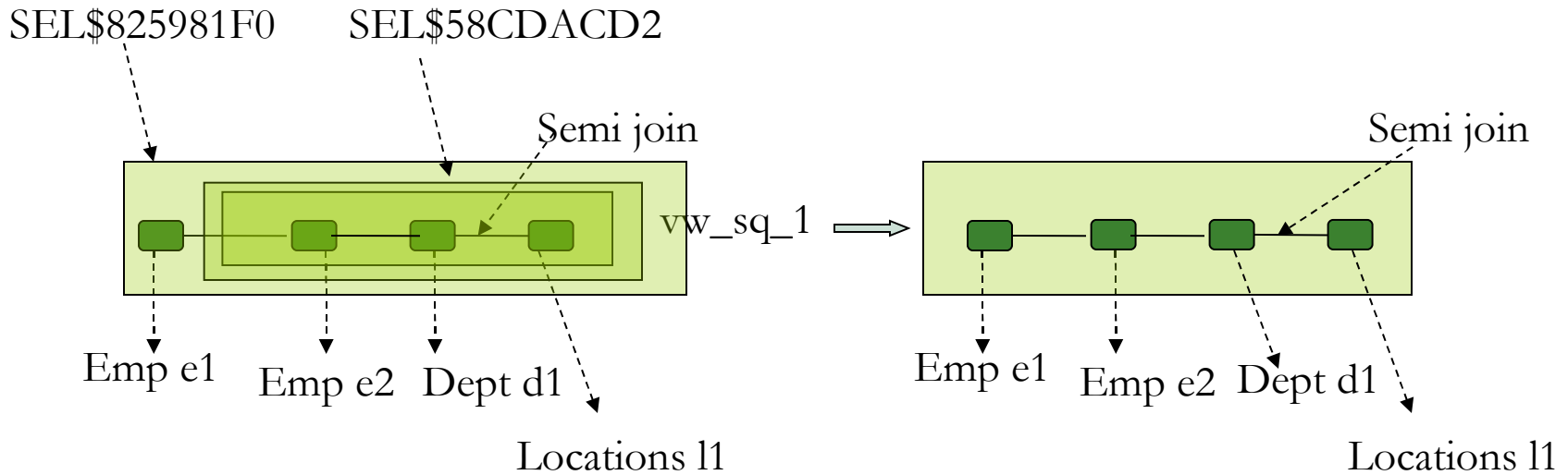
`_optimizer_reuse_cost_annotations`, default true, is controlling this behavior

Short summary so far...

- Exists subquery unnested in to a semi join.
- Next level subquery unnested in to a join.
- Filter predicates pushed down to various query blocks
- Kkoqbc module called for inner query block separately.
- Cost of that query block saved as annotation for further reuse
- Kkoqbc module called for whole query.
- Cost of that query block saved as annotation for further reuse

View merge

VIEW MERGE SEL\$825981F0; SEL\$58CDACD2



Transformation using view merge

CVM: CBQT Marking query block SEL\$58CDACD2 (#2) as valid for CVM.

CVM: Merging complex view SEL\$58CDACD2 (#2) into SEL\$825981F0 (#1).

qbcpr:

vqbcpr:

CVM: result SEL\$825981F0 (#1)

Registered qb: SEL\$8370D25A 0x21ee0968 (VIEW MERGE SEL\$825981F0; SEL\$58CDACD2)

```
select /*+ qb_name (e1_outer) */ * from
emp e1,
```

```
(select /*+ qb_name (e2_inner) */
  avg(salary) ,
  dept_id item_0
from
  emp e2, dept d1, locations l1
  where e2.dept_id = d1.dept_id and
        l1.location_id S= d1.location_id
  group by dept_id) "vw_sq_1"
```

where

```
e1.salary > vw_sq_1."avg(salary)"
and e1.hire_date > sysdate - (10*365)
and e1.dept_id = vw_sq_1.item_0
```



```
select /*+ qb_name (e1_outer) */ e1.* from
emp e1, emp e2, dept d1, locations l1
```

where

```
e1.dept_id =e2.dept_id and
e1.hire_Date >sysdate@!-3650 and
e2.dept_id = d1.dept_id and
l1.location_id S= d1.location_id
```

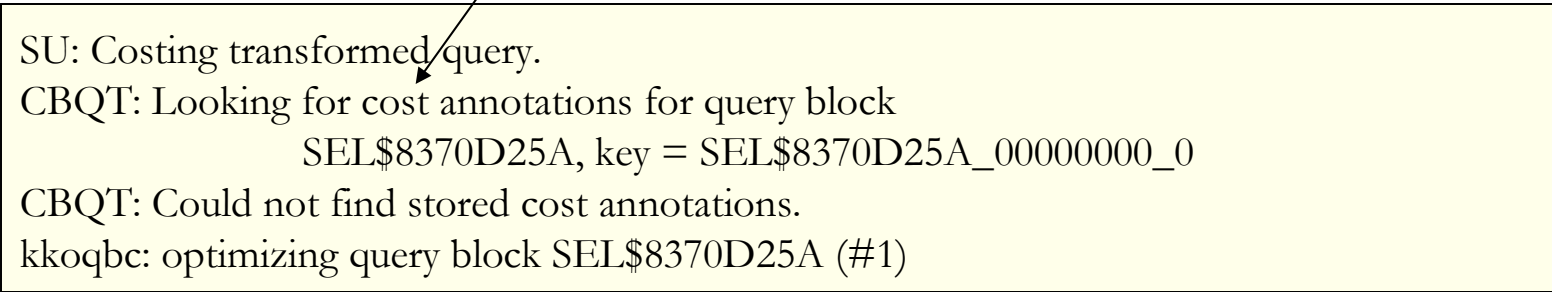
Group by

```
e2.Dept_id, e1.hire_date, e1.salary , e1.dept_id,
e1.emp_name, e1.emp_id
```

Having e1.salary > avg(e2.salary)

Costing transformed query

CBO is keeping track of cost annotations created already.
Query blocks are costed only if that query block with same
Signature is not costed already.



SU: Costing transformed query.

CBQT: Looking for cost annotations for query block

SEL\$8370D25A, key = SEL\$8370D25A_00000000_0

CBQT: Could not find stored cost annotations.

kkoqbc: optimizing query block SEL\$8370D25A (#1)

Transformation to a column level subquery

This subquery is rearranged in to a column level subquery.

```
select /*+ qb_name (e1_outer) */ * from
emp e1,
(select /*+ qb_name (e2_inner) */
  avg(salary), dept_id item_0
from emp e2, dept d1, locations l1
 where e1.dept_id = e2.dept_id and
        e2.dept_id = d1.dept_id and
        l1.location_id = d1.location_id
 group by dept_id) "vw_sq_1"
where
  e1.salary > vw_sq_1."avg(salary)"
  and e1.hire_date > sysdate - (10*365)
  and e1.dept_id = vw_sq_1.item_0
```



```
select emp_id, emp_name, dept_id, salary, hire_Date
from ( select e1.*,
  (select avg(salary) from
    emp e2, dept d1, locations l1
   where e1.dept_id = e2.dept_id and
         e2.dept_id = d1.dept_id and
         l1.location_id = d1.location_id
  ) avg_sal
from emp e1
)
where salary > avg_sal and hire_date > sysdate-3650
```

Costing column level sub-query

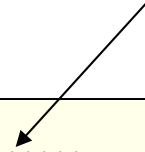
```
kkoqbc: optimizing query block SEL$D72FB22B (#2)
call(in-use=48584, alloc=98240), compile(in-use=256500, alloc=280672), execution(in-use=298148, alloc=298236)
signature (optimizer): qb_name=SEL$D72FB22B nbfros=3 flg=0
  fro(0): flg=0 objn=71162 hint_alias="D1"@ "E2_INNER"
  fro(1): flg=0 objn=71164 hint_alias="E2"@ "E2_INNER"
  fro(2): flg=0 objn=71160 hint_alias="L1"@ "L1_INNER"

kkoqbc-subheap (create addr=0x0C2F42D8)
*****
(newjo-save) [0 2 1 ]
Trying or-Expansion on query block SEL$D72FB22B (#2)
Transfer Optimizer annotations for query block SEL$D72FB22B (#2)
Final cost for query block SEL$D72FB22B (#2) - All Rows Plan:
  Best join order: 1
  Cost: 173.2842 Degree: 1 Card: 1.0000 Bytes: 21
  Resc: 173.2842 Resc_io: 172.0000 Resc_cpu: 28447119
```

Cost of one execution of column level subquery

GBP

Performance of SQL can be improved by moving around group by Operators to different levels. Group by operators are moved around and Cost calculated for transformed queries.



```
*****
```

```
Cost-Based Group By Placement
```

```
*****
```

```
GBP: Checking validity of GBP for query block SEL$58CDACD2 (#2)
```

```
GBP: Checking validity of group-by placement for query block SEL$58CDACD2 (#2)
```

```
GBP: Using search type: exhaustive
```

```
GBP: Considering group-by placement on query block SEL$58CDACD2 (#2)
```

```
GBP: Starting iteration 1, state space = (3,4,5) : (0,0,0)
```

```
GBP: Transformed query
```

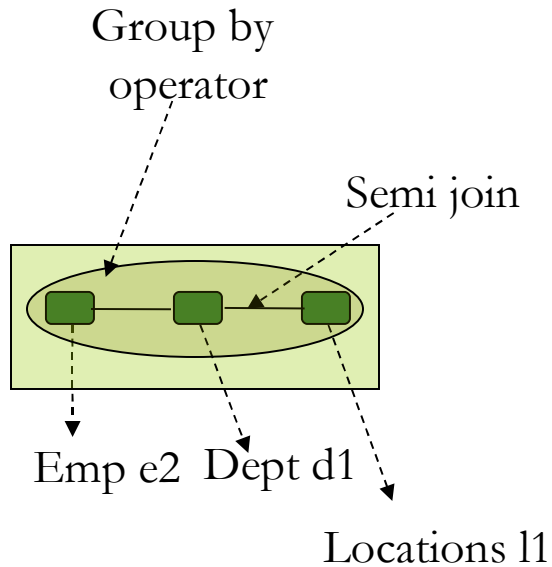
```
FPD: Considering simple filter push in query block SEL$58CDACD2 (#2)
```

```
"E2"."DEPT_ID"="D1"."DEPT_ID" AND "L1"."LOCATION_ID"="D1"."LOCATION_ID"
```

```
try to generate transitive predicate from check constraints for query block SEL$58CDACD2 (#2)
```

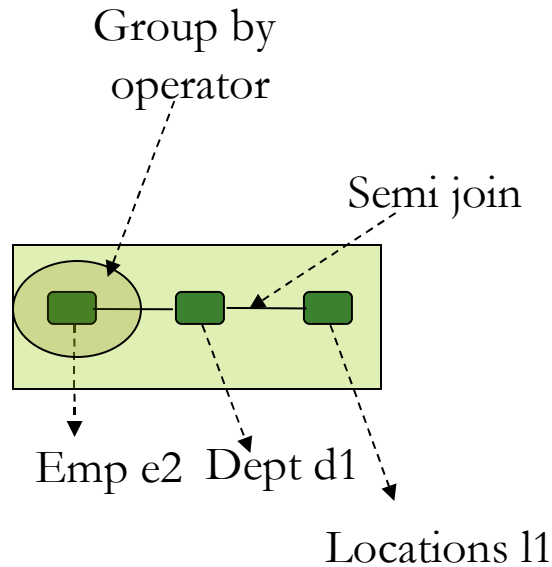
```
finally: "E2"."DEPT_ID"="D1"."DEPT_ID" AND "L1"."LOCATION_ID"="D1"."LOCATION_ID"
```

QBP#1



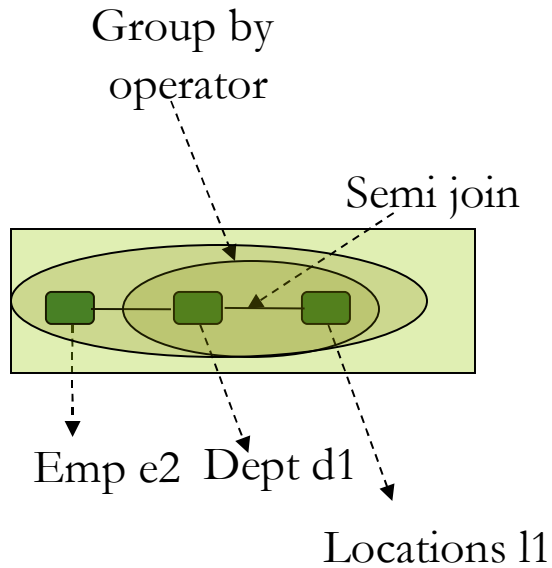
```
select avg(salary), dept_id item_1 from
emp e2 ,
dept d1,
locations l1
where
e2.dept_id = d1.dept_id and
d1.location_id S= l1.lcation_id
group by dept_id
```

QBP#2



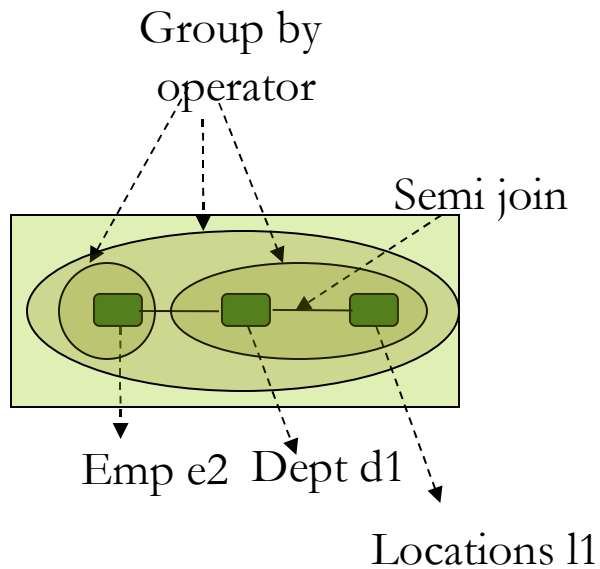
```
select vw_gbc_2.dept_id, avg_salary
from
  ( select avg(salary) avg_salary, dept_id item_1
    from emp e2
      group by dept_id ) vw_gbc_2,
dept d1,
locations l1
where
  vw_gbc_2.dept_id = d1.dept_id and
  and d1.location_id = l1.location_id
```

QBP#3



```
select
    sum(e2.salary*vw_gbf_3.item_2)/
    sum(decode(e2.salary, null, 0, vw_gbf_3.item_3))
        avg(salary),
    e2.dept_id item_1
from
    ( select d1.dept_id item_1, count(*) item_2,
          count(*) item_3
      from dept d1, locations l1
      where l1.location_id = d1.location_id
      group by d1.dept_id) vw_gbf_3,
    emp e2
where e2.dept_id = vw_gbf_3.item_1
group by d2.dept_id
```

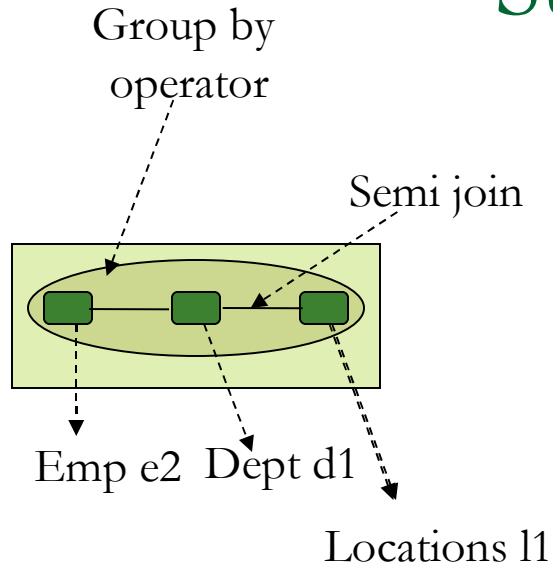
QBP#4



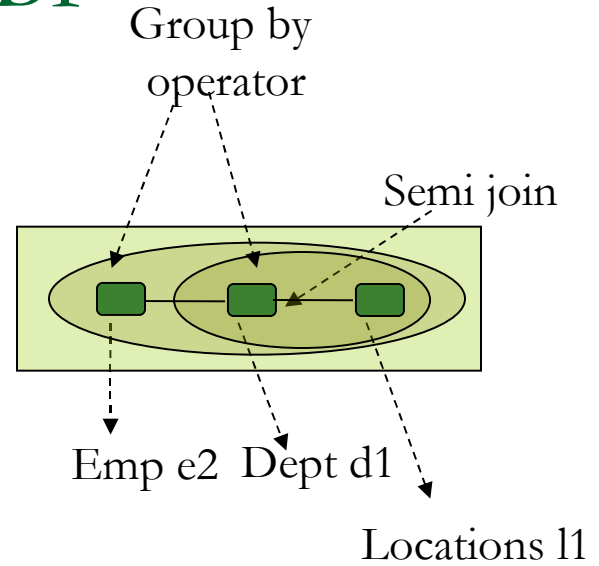
```
select
  sum (vw_gbc_4.item_2*vw_gbf_5.item_2)/
  sum(vw_gbc_4.item_3*vw_gbf_5.item_3)
  avg_salary, vw_gbc_4.item_4 item_1
from
  ( select d1.dept_id item_1, count(*) item_2,
        count(*) item_3
    from dept d1, locations l1 where
        l1.location_id = d1.location_id
      group by d1.dept_id) vw_gbf_5,
  (select e2.dept_id item_1, sum(e2.salary) item_2,
        count(e2.salary) item_3,
        e2.dept_id item_4,e2.dept_id item_5
    from emp e2
      group by e2.dept_id, e2.dept_id, e2.dept_id)
  vw_gbc_4
where
  vw_gbc_4.item_1 = vw_gbf_5.item_1
group by vw_gbc_4.item_5
```

Summary QBP

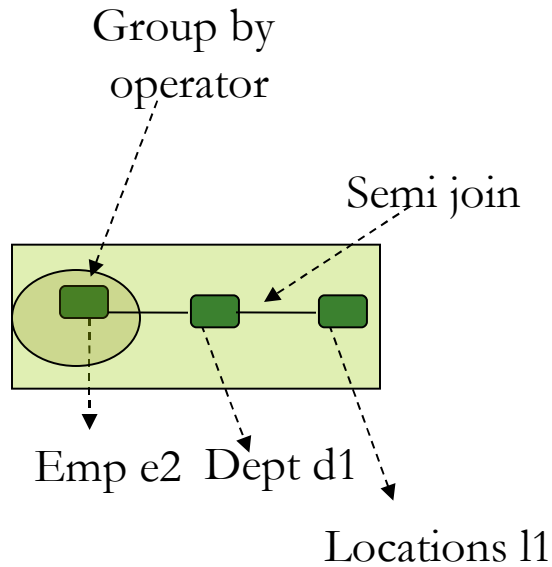
#1



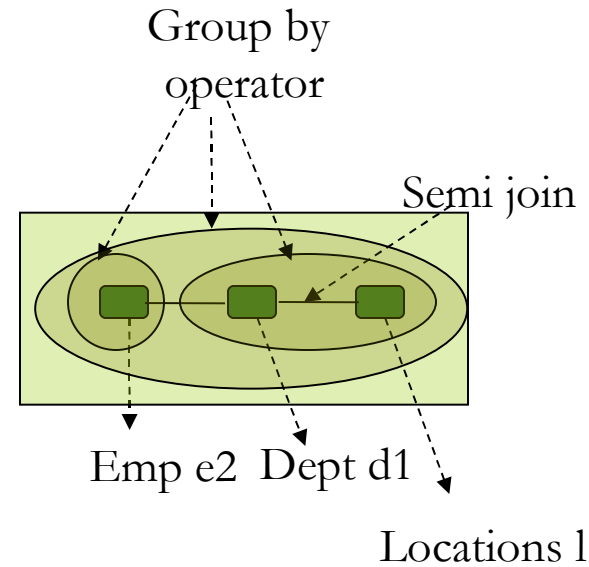
#3



#2



#4



SJC : Set Join Conversion

Generated predicates

Intersect set operation converted to a join.

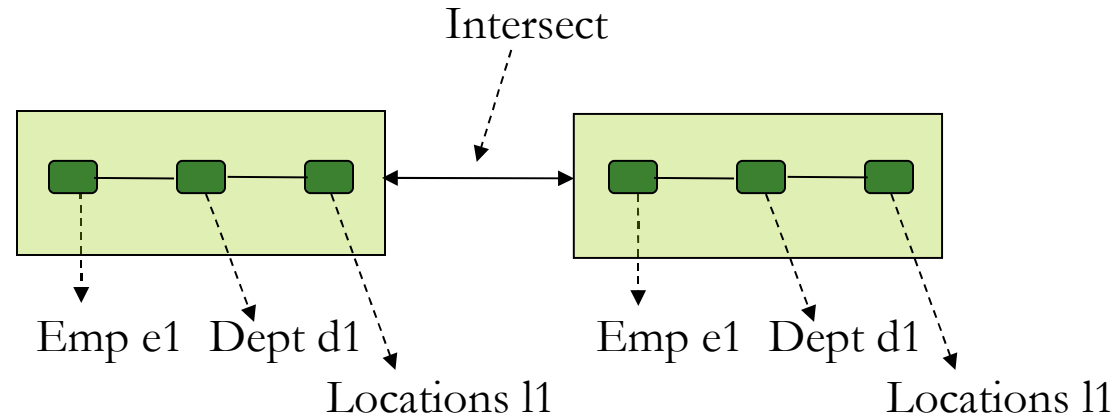
```
Select /*+ qb_name (e1_outer) */ * from
emp e1, dept d1, locations l1
where
e1.dept_id = d1.dept_id and
d1.location_id = l1.location_id and
e1.salary > 100000
intersect
Select /*+ qb_name (e2_outer) */ * from
emp e2 , dept d2, locations l2
where
e2.dept_id = d2.dept_id and
d2.location_id = l2.location_id and
hire_date > sysdate-365*10
```



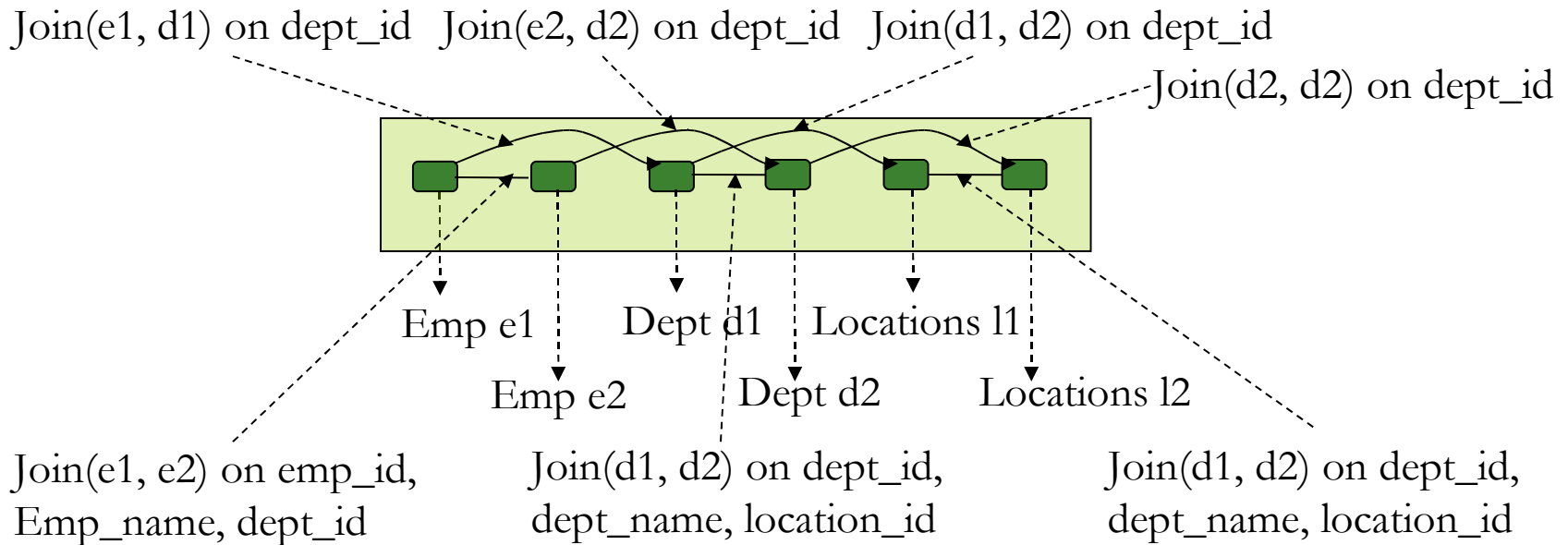
```
select distinct e1.emp_id, e1.emp_name, e1.dept_id,
e1.salary, e1.hire_Date, d1.dept_id, d1.dept_name,
d1.location_id, l1.location_id, l1.city_name, l1.state
from
emp e2, dept d2, locations l2, emp e1, dept d1, locations l1
where
e1.emp_id = e2.emp_id and
sys_op_map_nonnull(e1.emp_name) =
sys_op_map_nonnull (e2.emp_name) and
e1.dept_id = e2.dept_id and e1.salary = e2.salary and
e1.hire_date = e2.hire_Date and d1.dept_id = d2.dept_id and
sys_op_map_nonnull (d1.dept_name) =
sys_op_map_nonnull (d2.dept_name) and
d1.location_id = d2.location_id and
l1.location_id = l2.location_id and
sys_op_map_nonnull ( l1.city_name) =
sys_op_map_nonnull (l2.city_name) and
sys_op_map_nonnull ( l1.state) =
sys_op_map_nonnull(l2.state) and
e1.dept_id =d1.dept_id and d1.location_id = l1.location_id and
e1.salary > 100000 and e2.dept_id = d2.dept_id and
d2.location_id = l2.location_id and
e2.hire_date- sysdate@! -365*10
```

Predicates from query

SJC



↓ SJC



SJC : Set Join Conversion

Join conditions for self join
For same tables

sys_op_map_nonnull function is
An Undocumented function.
Null =Null simulated here

Join conditions from SQL before
transformation

```
select distinct e1.emp_id, e1.emp_name, e1.dept_id,  
               e1.salary, e1.hire_Date, d1.dept_id, d1.dept_name,  
               d1.location_id, l1.location_id, l1.city_name, l1.state  
from  
  emp e2, dept d2, locations l2, emp e1, dept d1, locations l1  
where
```

```
  e1.emp_id = e2.emp_id and  
  sys_op_map_nonnull(e1.emp_name) =  
    sys_op_map_nonnull (e2.emp_name) and  
  e1.dept_id = e2.dept_id and e1.salary = e2.salary and  
  e1.hire_date = e2.hire_Date and d1.dept_id = d2.dept_id and  
  sys_op_map_nonnull (d1.dept_name) =  
    sys_op_map_nonnull (d2.dept_name) and  
  d1.location_id = d2.location_id and  
  l1.location_id = l2.location_id and  
  sys_op_map_nonnull ( l1.city_name) =  
    sys_op_map_nonnull (l2.city_name) and  
  sys_op_map_nonnull ( l1.state) =  
    sys_op_map_nonnull(l2.state) and
```

```
  e1.dept_id =d1.dept_id and d1.location_id = l1.location_id and  
  e1.salary > 100000 and e2.dept_id = d2.dept_id and  
  d2.location_id = l2.location_id and  
  e2.hire_date- sysdate@! -365*10
```

SJC : Set Join Conversion

Registered qb: SEL\$02B15F54 0x1ded2d00 (VIEW MERGE SET\$09AAA538; SEL\$1 SEL\$2)

QUERY BLOCK SIGNATURE

signature (): qb_name=SEL\$02B15F54 nbfros=6 flg=0
fro(0): flg=0 objn=73174 hint_alias="D1"@SEL\$1"
fro(1): flg=0 objn=73176 hint_alias="E1"@SEL\$1"
fro(2): flg=0 objn=73172 hint_alias="L1"@SEL\$1"
fro(3): flg=0 objn=73174 hint_alias="D1"@SEL\$2"
fro(4): flg=0 objn=73176 hint_alias="E1"@SEL\$2"
fro(5): flg=0 objn=73172 hint_alias="L1"@SEL\$2"

JE: Join Elimination

Foreign key between dept and
Locations added.

```
alter table dept add constraint dept_fk foreign key (location_id )  
references locations (location_id);
```

```
alter session set "_optimizer_join_elimination_enabled" =false;
```

explain plan for

```
select d1.* from dept d1 where
```

```
exists (select 1 from locations l1 where l1.location_id = d1.location_id );
```

Plan hash value: 762406906

Exists converted to semi join
between dept and locations

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	91	2 (0)	00:00:01
1	NESTED LOOPS SEMI		1	91	2 (0)	00:00:01
2	TABLE ACCESS FULL	DEPT	1	78	2 (0)	00:00:01
* 3	INDEX UNIQUE SCAN	SYS_C0010444	1	13	0 (0)	00:00:01

JE: Join Elimination

```
alter session set "_optimizer_join_elimination_enabled" =true ;
```

explain plan for

```
select d1.* from dept d1 where
```

```
exists (select 1 from locations l1 where l1.location_id = d1.location_id );
```

Id	Operation	Name	Rows	Bytes	Cost	Time
0	SELECT STATEMENT				2	
1	TABLE ACCESS FULL	DEPT	1	78	2	00:00:01

Predicate Information:

1 - filter("D1"."LOCATION_ID" IS NOT NULL)

Missing locations from the
Access plan...

JE

JE: Considering Join Elimination on query block SEL\$5DA710D3 (#1)

Join Elimination (JE)

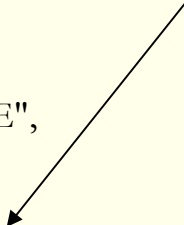
JE: cfro: DEPT objn:73501 col#:3 dfro:LOCATIONS dcol#:3

Query block (1E68C358) before join elimination:

SQL:***** UNPARSED QUERY IS *****

```
SELECT "D1"."DEPT_ID" "DEPT_ID","D1"."DEPT_NAME" "DEPT_NAME",  
       "D1"."LOCATION_ID" "LOCATION_ID"  
FROM "CBQT2"."LOCATIONS" "L1","CBQT2"."DEPT" "D1"  
WHERE "D1"."LOCATION_ID"="L1"."LOCATION_ID"
```

Locations table eliminated
as validated foreign key
Constraint guarantees that.



JE: eliminate table: LOCATIONS

Registered qb: SEL\$48A72308 0x1e68c358

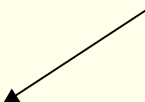
(JOIN REMOVED FROM QUERY BLOCK SEL\$5DA710D3; SEL\$5DA710D3; "L1"@SEL\$2")

QUERY BLOCK SIGNATURE

signature (): qb_name=SEL\$48A72308 nbfros=1 flg=0

fro(0): flg=0 objn=73503 hint_alias="D1"@SEL\$1"

Rewritten query does not have
reference to locations table.



SQL:***** UNPARSED QUERY IS *****

```
SELECT "D1"."DEPT_ID" "DEPT_ID","D1"."DEPT_NAME" "DEPT_NAME","D1"."LOCATION_ID"  
       "LOCATION_ID" FROM "CBQT2"."DEPT" "D1" WHERE "D1"."LOCATION_ID" IS NOT NULL
```

PM: Predicate Move around

explain plan for

```
Select /*+ qb_name (v_outer) */ v1.* from
  ( select /*+ qb_name (v1) no_merge */ e1.*, d1.location_id from
    emp e1, dept d1
    where e1.dept_id = d1.dept_id and d1.dept_id=200) v1,
  (select/*+ qb_name (v2) no_merge */ avg(salary) avg_sal_dept, d2.dept_id from
    emp e2, dept d2, locations l2
    where
      e2.dept_id = d2.dept_id and
      l2.location_id=d2.location_id and d2.dept_id=100
    group by d2.dept_id
  )
  v2_dept
where
  v1.dept_id =v2_Dept.dept_id and v1.salary > v2_dept.avg_sal_dept
and v2_dept.dept_id=300
;
```

PM – General steps [3]

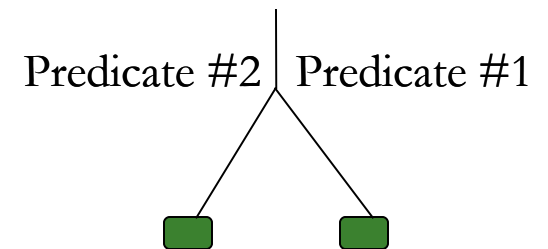
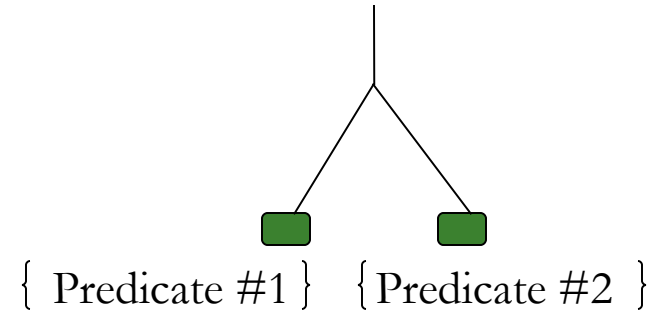
Predicate pull up

- Tree traversed bottom-up

Predicate push down

- Tree traversed top-down

Duplicate predicate removal

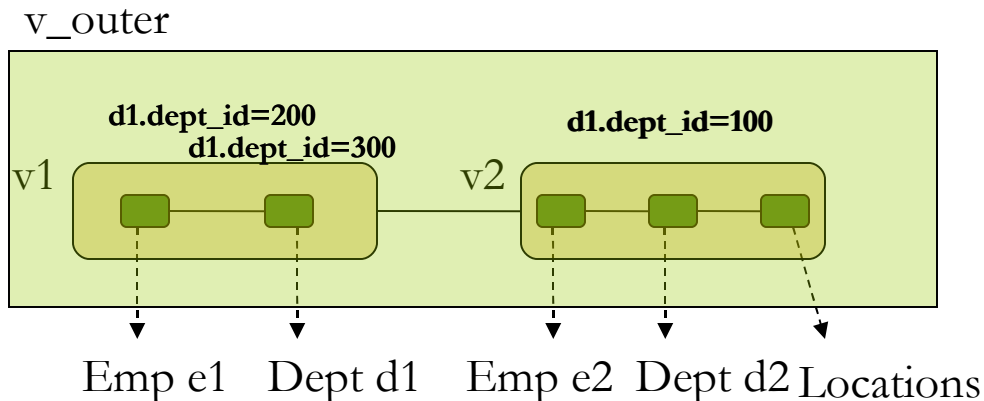


PM

PM: Pulled up predicate "V1"."DEPT_ID`=300
from query block V1 (#2) to query block V_OUTER (#1)

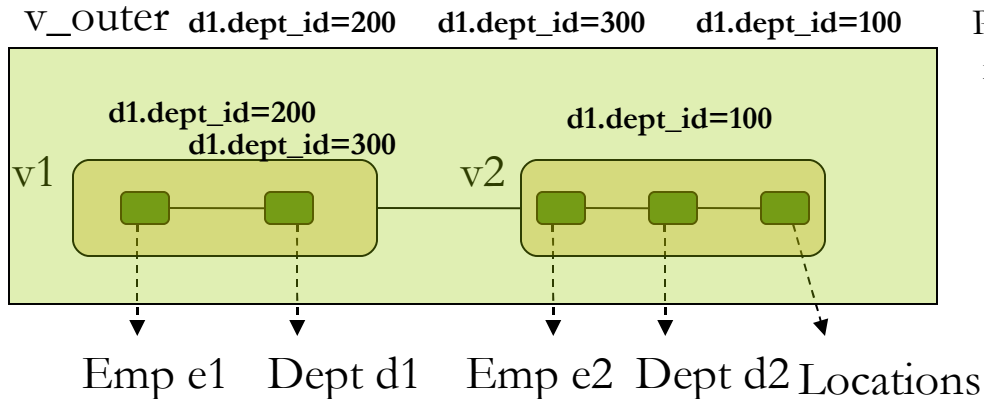
PM: Pulled up predicate "V1"."DEPT_ID`=300
from query block V1 (#2) to query block V_OUTER (#1)

PM: Pulled up predicate "V2_DEPT"."DEPT_ID`=100
from query block V2 (#3) to query block V_OUTER (#1)



12

PM



PM: Pulled up predicate "V1"."DEPT_ID "=300
from query block V1 (#2) to query block V_OUTER (#1)

PM: Pulled up predicate "V1"."DEPT_ID "=300
from query block V1 (#2) to query block V_OUTER (#1)

PM: Pulled up predicate "V2_DEPT"."DEPT_ID "=100
from query block V2 (#3) to query block V_OUTER (#1)

PM: Pushed down predicate "E1"."DEPT_ID "=100
from query block V_OUTER (#1) to query block V1 (#2)

PM: Pushed down predicate "D2"."DEPT_ID "=200
from query block V_OUTER (#1) to query block V2 (#3)

CNT(col) to CNT(*)

- Count (col) are converted to COUNT(*)

explain plan for
select count(emp_id) from emp e1
where exists (
 select 1 from emp e2, dept d2
 where e2.dept_id = d2.dept_id
 and e1.dept_id = e2.dept_id
)
and e1.hire_Date > sysdate-90

CNT: Considering count(col) to count(*) on query block SEL\$1 (#0)

Count(col) to Count(*) (CNT)

CNT: Converting COUNT(EMP_ID) to COUNT(*).

CNT: COUNT() to COUNT(*) done.

_optimizer_cost_based_transformation

- Various values are possible to turn on cost based transformation, for this parameter:
 - two_pass
 - on
 - exhaustive
 - linear
 - iterative

- No difference was found between various modes of this transformation for this SQL.

- This parameter can be modified at Session level

More parameters..

```
_unnest_subquery           = true # Unnest subquery
_eliminate_common_subexpr  = true # Eliminate common sub expression
_pred_move_around         = true # Predicate move around
_convert_set_to_join       = false # Convert set to join
_optimizer_order_by_elimination_enabled = true # Order by elimination check
_optimizer_distinct_elimination = true # Distinct elimination
_optimizer_multi_level_push_pred = true # push predicates multiple level
_optimizer_group_by_placement = true # Consider group by placement
_optimizer_reuse_cost_annotations = true # Reuse cost annotations
```

Memory usage

- Memory managed better during parsing.
- Sub heaps can be de-allocated before process death.
- These heaps are needed only during parsing and not at execution

```
_optimizer_use_subheap          = true # Use sub heap  
_optimizer_free_transformation_heap = true # Free heaps after transformation  
_optimizer_or_expansion_subheap  = true
```

```
kkoqbc-subheap (create addr=0x090AC150)  
...  
kkoqbc-subheap (delete addr=0x090AC150, in-use=28616, alloc=31212)  
....
```

Model & CBQT

CBQT is disabled for Model SQL clause.

```
select a.* from (  
select item, location, week, inventory, sales_so_far,sales_qty, rcpt_qty  
  from item_data  
  model return updated rows  
  partition by (item)  
  dimension by (location, week)  
  measures ( 0 inventory , sales_qty, rcpt_qty, 0 sales_so_far)  
  rules sequential order (  
inventory [location, week] = nvl(inventory [cv(location), cv(week)-1 ] ,0)  
      - sales_qty [cv(location), cv(week) ] +  
      + rcpt_qty [cv(location), cv(week) ],  
sales_so_far [location, week] = sales_qty [cv(location), cv(week)] +  
      nvl(sales_so_far [cv(location), cv(week)-1],0)  
  ) order by item , location,week  
  ) a, locations l  
where a.location =l.location_id
```

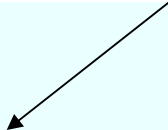
Query transformations (QT)

CBQT bypassed for query block SEL\$1 (#0): contained query block.

CBQT: Validity checks failed for 4um58uddcnx2k.

Analytic functions & CBQT

CBQT is enabled for analytic functions
Though.



```
explain plan for
select a.*, i.item
from (
select item, location, week, sales_qty, rcpt_qty,
       sum (sales_qty) over (
         partition by item, location
         order by week
         rows between unbounded preceding and current row
       ) running_sales_total
from item_data )a , item_data i
where a.item=i.item
```

Check Basic Validity for Non-Union View for query block SEL\$2 (#0)

CBQT: Validity checks passed for 6rd24ujfu08s5.

CSE: Considering common sub-expression elimination in query block SEL\$1 (#0)

CTAS & CBQT

Parts of CBQT disabled for CTAS

```
create table cbqt_test as
Select /*+ qb_name (e1_outer) */ * from emp e1 where
salary >
(select /*+ qb_name (e2_inner) */ avg(salary) from emp e2, dept d1
where e1.dept_id = e2.dept_id and e2.dept_id = d1.dept_id
and exists
(select /*+ qb_name (l1_inner) */ 1 from locations l1
where l1.location_id=d1.location_id ))
and e1.hire_date > sysdate - (10*365)
```

Predicate Move-Around (PM)

PM: ~~PM bypassed: Not a SELECT statement~~

_enable_pmo_ctas seems to be
Controlling this behavior.

Cost-Based Group By Placement

GBP: Checking validity of GBP for query block SEL\$58CDACD2 (#2)

GBP: Checking validity of group-by placement for query block SEL\$58CDACD2 (#2)

GBP: ~~Bypassed: create table.~~

Performance comparison

So, What's the performance improvement expected ?

`_optimizer_cost_based_transformation=linear`

Elapsed: 00:00:05.60

Statistics

0	recursive calls
0	db block gets
21192	consistent gets
10892	physical reads
0	redo size
6556650	bytes sent via SQL*Net to client
110911	bytes received via SQL*Net from client
10047	SQL*Net roundtrips to/from client
1	sorts (memory)
0	sorts (disk)
150689	rows processed

`_optimizer_cost_based_transformation=OFF`

Elapsed: 00:00:39.61

Statistics

33	recursive calls
0	db block gets
11206	consistent gets
14481	physical reads
0	redo size
6556650	bytes sent via SQL*Net to client
110911	bytes received via SQL*Net from client
10047	SQL*Net roundtrips to/from client
1	sorts (memory)
0	sorts (disk)
150689	rows processed

Script: `perf_comparison.sql`

References

- [1] Cost based query transformation in Oracle – VLDB Sept 06
ACM 1-59593-385-9/06/09
Rafi Ahmed, Allison Lee, Andrew Witkowski, Dinesh Das, Hong Su, Mohamed Zait
Thierry Cruanes
- [2] Reusing optimized query block in query processing: US Patent 7,246,108B2
Rafi Ahmed, Oracle Corporation
- [3] Query optimization by predicate move around: US Patent 5,659, 725
Alon Yitzchak Levy, Inderpal Singh Mumick,
- [4] Cost Based Oracle Fundamentals By Jonathan Lewis, APRESS publication,
ISBN 1-59059-636-6
- [5] Query Transformation, presentation By Jose Senegacnik, UKOUG 2007