
Cost Based Query Transformations

By

Riyaj Shamsudeen

Who am I?

- 15 years using Oracle products
- Over 14 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 AT&T through Avion systems
- OakTable member
- Email: rshams4 at yahoo.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.

Agenda

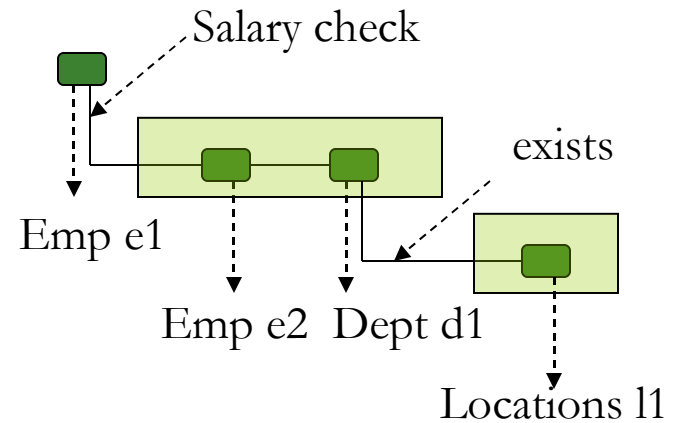
- What is transformation & Why do we need to cost?
- Explanation of query.
- Review concepts behind Query transformations and Map them back to 10053 trace.
- Parameters affecting CBQT behavior.
- Questions.

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';

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

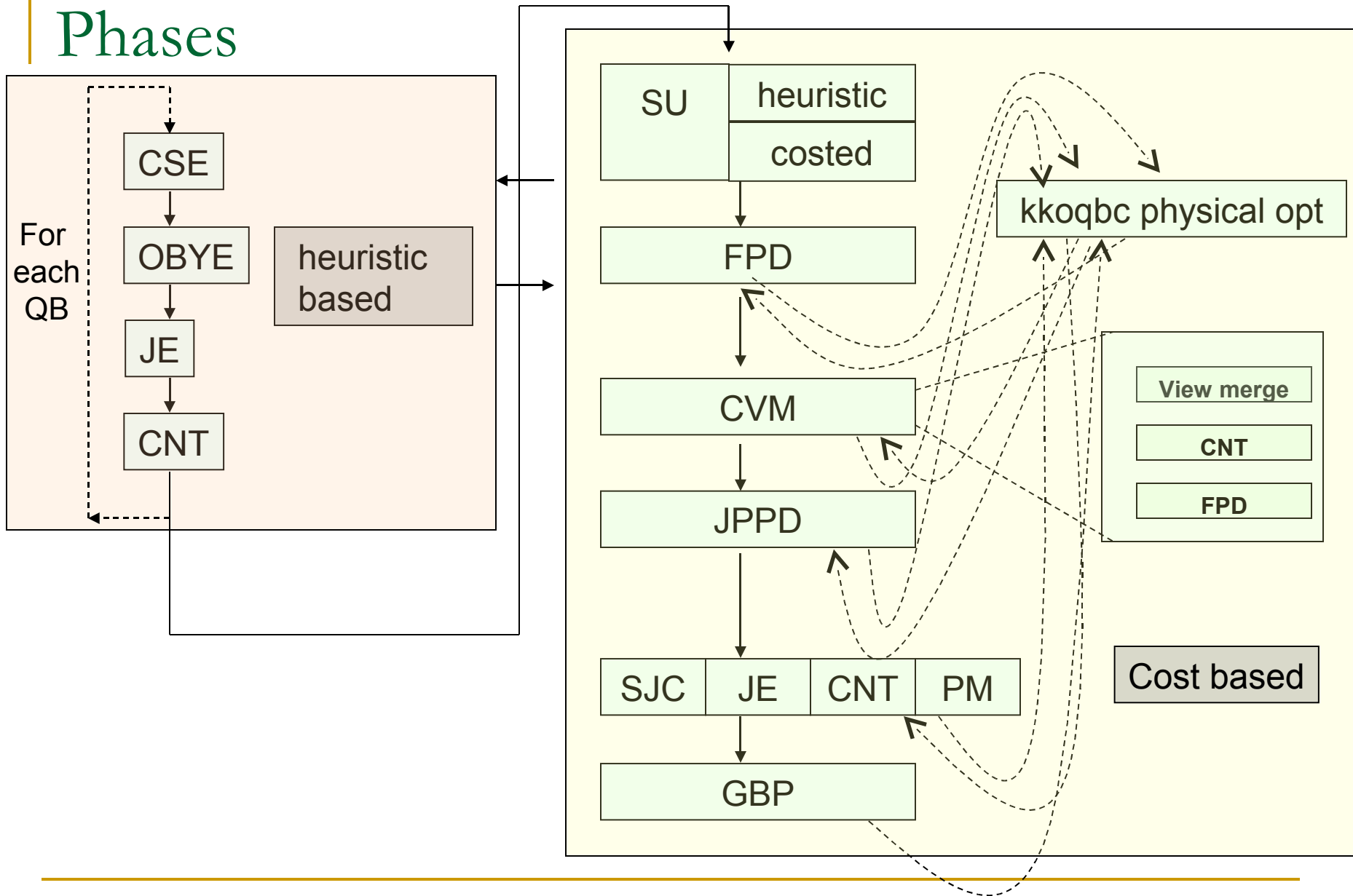
where
e1.hire_date > sysdate - (10*365) and
e1. salary > gbp1.avg_salary

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



Subquery unnest

‘exists’ sub-query for locations
unnested as a semi-join

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

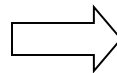
signature (): qb_name=SEL\$D72FB22B nbfros=3 flg=0

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

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

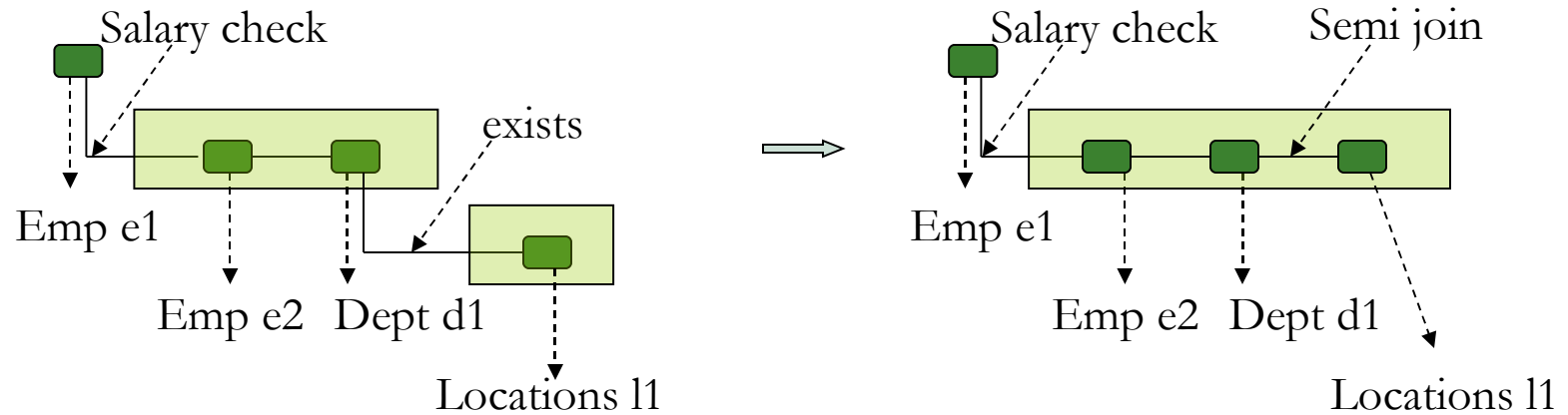
fro(2): flg=0 objn=71160 hint_alias="L1"@L1_INNER"

```
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



Next step

Subquery moved in to a view for subsequent transformation

Registered qb: SEL\$58CDACD2 0x21ee4a5c (SUBQ INTO VIEW FOR COMPLEX UNNEST SEL\$D72FB22B)

QUERY BLOCK SIGNATURE

signature (): qb_name=SEL\$58CDACD2 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"

Registered qb: SEL\$4ADFCC1B 0x21ee5968 (VIEW ADDED E1_OUTER)

QUERY BLOCK SIGNATURE

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"

added emp with alias e1_outer
to create a new query block

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"
```

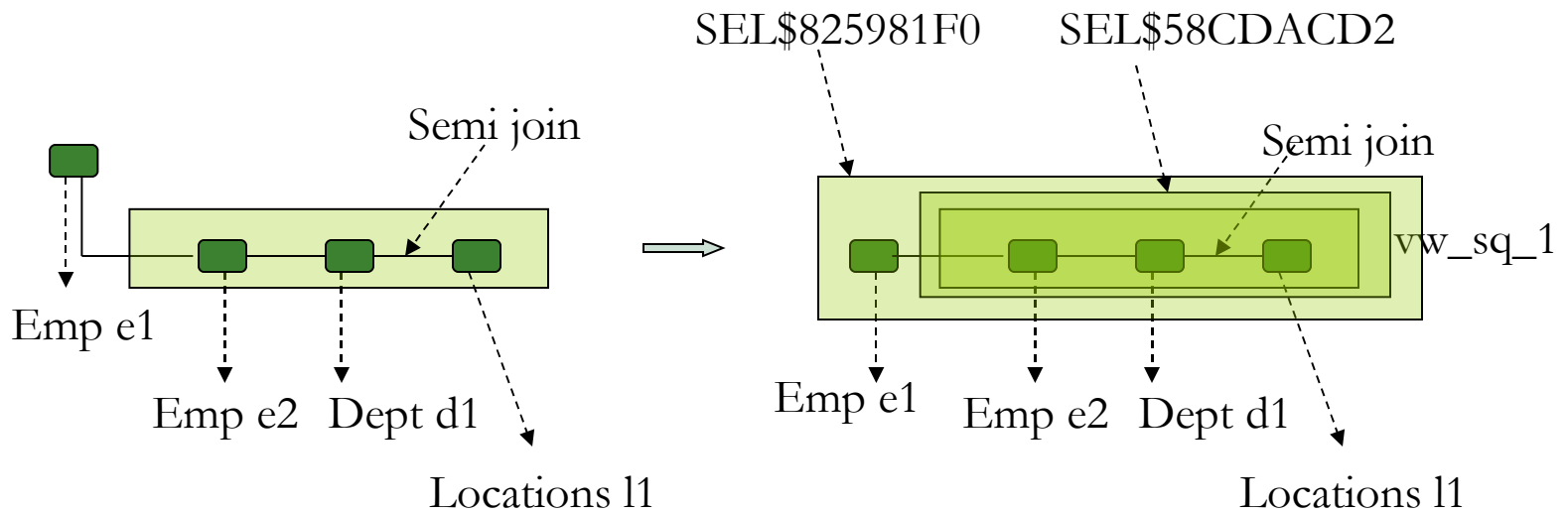
```
Registered qb: SEL$825981F0 0x21ee5968 (SUBQUERY UNNEST SEL$4ADFCC1B; SEL$D72FB22B)
```

```
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
```

Subquery unnest

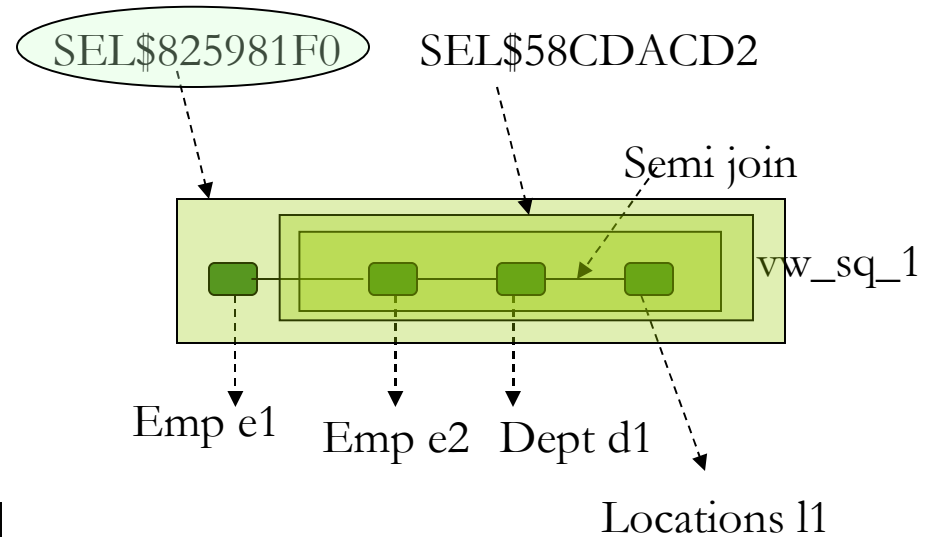
(SUBQUERY UNNEST SEL\$4ADFCC1B; SEL\$D72FB22B)



Transformation - FPD

Filter Push Down (FPD) pushing the Predicates in to named query block SEL\$825981F0

FPD: Considering simple filter push in query block SEL\$825981F0 (#1)
 "E1"."SALARY">"VW_SQ_1"."AVG(SALARY)"
 AND
 "E1"."DEPT_ID"="VW_SQ_1"."ITEM_0"
 AND "E1"."HIRE_DATE">SYSDATE@!-3650



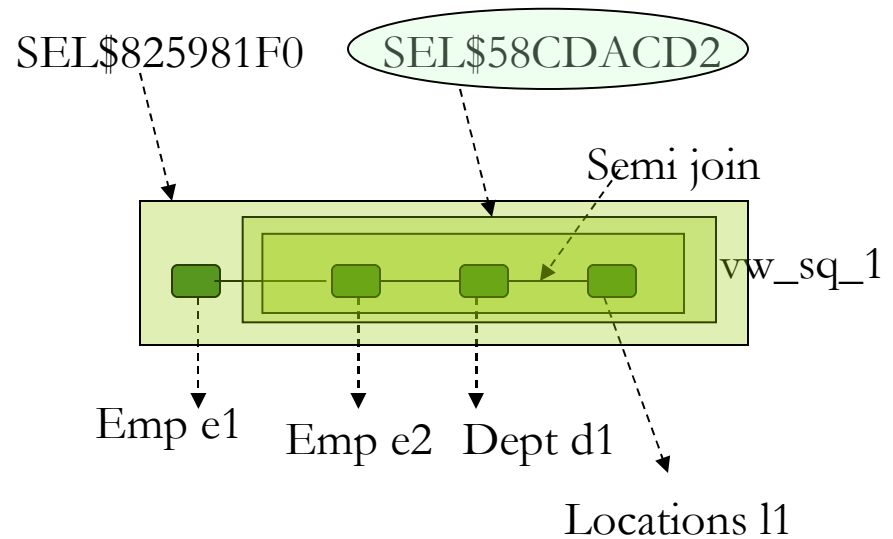
try to generate transitive predicate from check constraints for query block SEL\$825981F0 (#1)
 finally:
 "E1"."SALARY">"VW_SQ_1"."AVG(SALARY)"
 AND
 "E1"."DEPT_ID"="VW_SQ_1"."ITEM_0"
 AND "E1"."HIRE_DATE">SYSDATE@!-3650

Transitive predicate generation in query Block SEL\$825981F0

Transformation - FPD

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"



Next query block SEL\$58CDACD2
Considered here for FPD.

Costing Query blocks

Physical optimizer is called using kkoqbc module for sub-query.

Cost saved as annotations for query blocks.
This reduces amount of time spent in parsing, as calls to kkoqbc modules are reduced.

SU: Costing transformed query.

CBQT: Looking for cost annotations for query block SEL\$58CDACD2, key = SEL\$58CDACD2_00000202_2

CBQT: Could not find stored cost annotations.

kkoqbc: optimizing query block SEL\$58CDACD2 (#2)

...

QUERY BLOCK SIGNATURE

signature (optimizer): qb_name=SEL\$58CDACD2 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"

Typical lines from 10053 trace shown here.

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.

Costing Query blocks

Whole query is costed using kkoqbc module.

```
kkoqbc: optimizing query block SEL$825981F0 (#1)
```

```
QUERY BLOCK SIGNATURE
```

```
signature (optimizer): qb_name=SEL$825981F0 nbfros=2 flg=0
```

```
  fro(0): flg=0 objn=71164 hint_alias="E1"@ "E1_OUTER"
```

```
  fro(1): flg=1 objn=0 hint_alias="VW_SQ_1"@ "SEL$4ADFCC1B"
```

```
Trying or-Expansion on query block SEL$825981F0 (#1)
```

```
Transfer Optimizer annotations for query block SEL$825981F0 (#1)
```

```
Final cost for query block SEL$825981F0 (#1) - All Rows Plan:
```

```
  Best join order: 1
```

```
  Cost: 663.9818 Degree: 1 Card: 1521.0000 Bytes: 94302
```

```
  Resc: 663.9818 Resc_io: 652.0000 Resc_cpu: 265406889
```

```
  Resp: 663.9818 Resp_io: 652.0000 Resc_cpu: 265406889
```

```
kkoqbc-subheap (delete addr=0x07A8D744, in-use=19624, alloc=22500)
```

```
kkoqbc-end:
```

```
kkoqbc: finish optimizing query block SEL$825981F0 (#1)
```

```
CBQT: Saved costed qb# 2 (SEL$58CDACD2), key = SEL$58CDACD2_00000202_2
```

```
CBQT: Saved costed qb# 1 (SEL$825981F0), key = SEL$825981F0_00000000_0
```

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

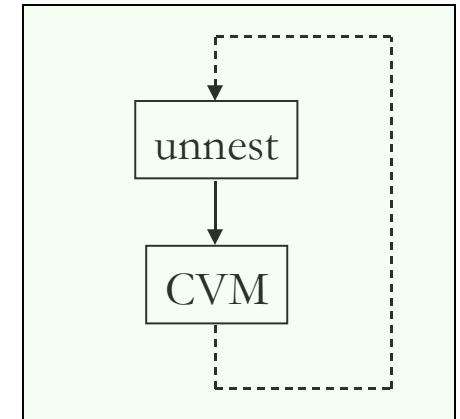
Short summary so far...

- 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

Interleaved CVM

→ Query is undergoing next transformation

→ Unnesting and complex view merging are interleaved here



SU: Considering interleaved complex view merging

SU: Unnesting subquery query block SEL\$D72FB22B (#2)

Subquery elimination for query block SEL\$D72FB22B (#2)

Subquery unchanged.

CVM: Considering view merge (candidate phase) in query block SEL\$825981F0 (#1)

CVM: Considering view merge (candidate phase) in query block SEL\$58CDACD2 (#2)

CNT: Considering count(col) to count(*) on query block SEL\$58CDACD2 (#2)

CVM: Complex view merging

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
```

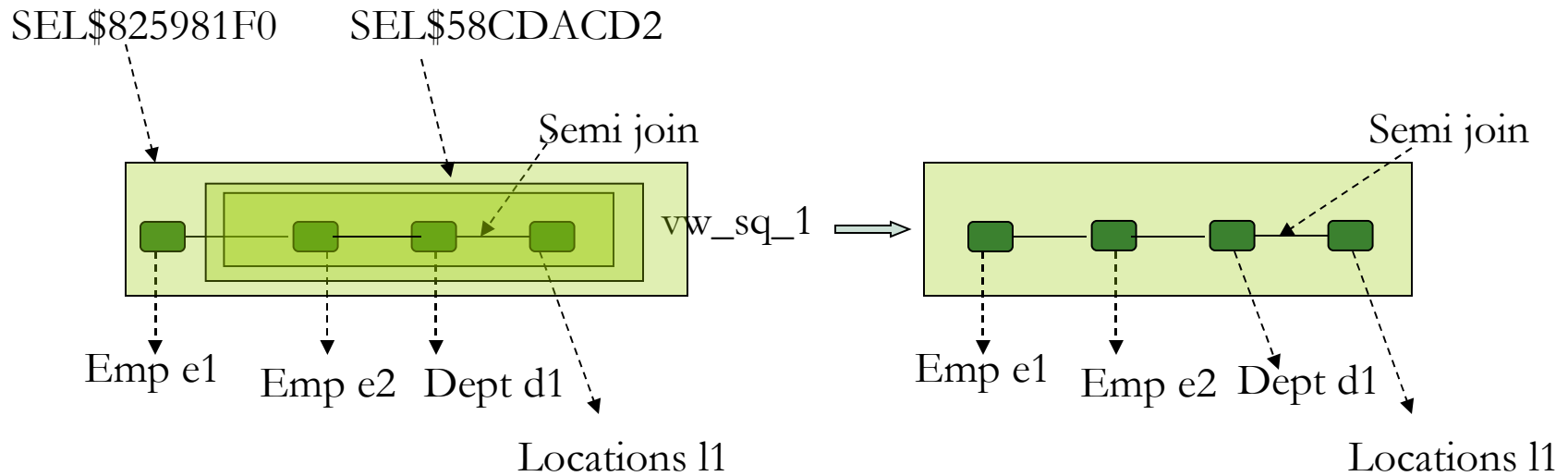
Group by

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

Having e1.salary > avg(e2.salary)

View merge

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



FPD

A new query block is born, result of view merge.

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

QUERY BLOCK SIGNATURE

```
signature (): qb_name=SEL$8370D25A nbfros=4 flg=0
fro(0): flg=0 objn=71164 hint_alias="E1"@ "E1_OUTER"
fro(1): flg=0 objn=71162 hint_alias="D1"@ "E2_INNER"
fro(2): flg=0 objn=71164 hint_alias="E2"@ "E2_INNER"
fro(3): flg=0 objn=71160 hint_alias="L1"@ "L1_INNER"
```

FPD: Considering simple filter push in query block SEL\$8370D25A (#1)
"E1"."DEPT_ID"="E2"."DEPT_ID" AND "E1"."HIRE_DATE">SYSDATE@!-3650 AND
"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\$8370D25A (#1)
finally: "E1"."DEPT_ID"="E2"."DEPT_ID" AND "E1"."HIRE_DATE">SYSDATE@!-3650 AND
"E2"."DEPT_ID"="D1"."DEPT_ID" AND "L1"."LOCATION_ID"="D1"."LOCATION_ID"

Costing transformed query

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)

```
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=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)

Final cost for query block SEL\$8370D25A (#1) - All Rows Plan:

Best join order: 8

Cost: 2169.0252 Degree: 1 Card: 303830.0000 Bytes: 17318310

Resc: 2169.0252 Resc_io: 2144.0000 Resc_cpu: 554331987

Resp: 2169.0252 Resp_io: 2144.0000 Resc_cpu: 554331987

kkoqbc-subheap (delete addr=0x0C2FCDBC, in-use=39568,
alloc=39864)

kkoqbc-end:

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

SU: Finished interleaved complex view merging

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.

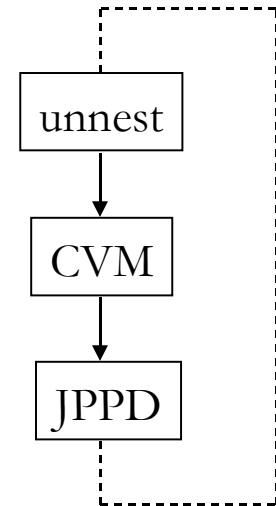
CBQT: Looking for cost annotations for query block

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

CBQT: Could not find stored cost annotations.

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

JPPD



SU: Considering interleaved join pred push down

SU: Unnesting subquery query block SEL\$D72FB22B (#2)

Subquery elimination for query block SEL\$D72FB22B (#2)

Subquery unchanged.

JPPD: Checking validity of push-down in query block SEL\$825981F0 (#1)

JPPD: Checking validity of push-down from query block SEL\$825981F0 (#1)
to query block SEL\$58CDACD2 (#2)

Check Basic Validity for Non-Union View for query block SEL\$58CDACD2 (#2)

JPPD: JPPD bypassed: No valid join condition found.

JPPD: No valid views found to push predicate into.

JPPD: Rejected interleaved query.

SU: Finished interleaved join pred push down

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 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 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
```

Transformation – column level subquery

SU: Starting iteration 2, state space = (2) : (0)

FPD: Considering simple filter push in query block SEL\$D72FB22B (#2)

"E2"."DEPT_ID"=:B1 AND "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\$D72FB22B (#2)

finally: "E2"."DEPT_ID"=:B1 AND "E2"."DEPT_ID"="D1"."DEPT_ID" AND
"L1"."LOCATION_ID"="D1"."LOCATION_ID" AND "D1"."DEPT_ID"=:B2

Note the bind variable, not a join condition, since now
This query is part of column level subquery.

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

Costing outer query block

QUERY BLOCK SIGNATURE

signature (optimizer): qb_name=E1_OUTER nbfros=1 flg=0
fro(0): flg=0 objn=71164 hint_alias="E1"@E1_OUTER

.....

Number of join permutations tried: 1

Final adjusted join cardinality: 1521, sq. fil. factor: 20.000000

Trying or-Expansion on query block E1_OUTER (#1)

Transfer Optimizer annotations for query block E1_OUTER (#1)

Final cost for query block E1_OUTER (#1) - All Rows Plan:

Best join order: 1

Cost: 2694538.1787 Degree: 1 Card: 1521.0000 Bytes: 973216

Resc: 2694538.1787 Resc_io: 2674566.1836 Resc_cpu: 442397934575

Resp: 2694538.1787 Resp_io: 2674566.1836 Resc_cpu: 442397934575

Total cost of this transformation is high

SJC – Set to Join Conversion

1. Some set operations can be transformed in to join conditions.
3. For this SQL, SJC not performed. This is discussed later with a different query

SJC: Considering set-join conversion in query block SEL\$825981F0 (#1)

Set-Join Conversion (SJC)

SJC: Considering set-join conversion in query block SEL\$58CDACD2 (#2)

Set-Join Conversion (SJC)

SJC: not performed

SJC: not performed

CNT: Considering count(col) to count(*) on query block SEL\$58CDACD2 (#2)

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

CNT: COUNT() to COUNT(*) not done.

PM & JE

JE: Considering Join Elimination on query block SEL\$58CDACD2 (#2)

Join Elimination (JE)

JE: cfro: DEPT objn:71160 col#:3 dfro:LOCATIONS dcol#:3

SQL:

Query block SEL\$58CDACD2 (#2) unchanged

JE: Considering Join Elimination on query block SEL\$825981F0 (#1)

Join Elimination (JE)

SQL:

Query block SEL\$825981F0 (#1) unchanged

PM: Considering predicate move-around in query block SEL\$825981F0 (#1)

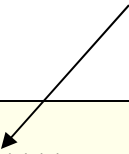
Predicate Move-Around (PM)

PM: Passed validity checks.

PM: PM bypassed: checking.

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"
```


Transformation – GBP #1

```
GBP: Costing transformed query.
CBQT: Looking for cost annotations for query block
      SEL$58CDACD2, key = SEL$58CDACD2_00000002_0
CBQT: Could not find stored cost annotations.
kkoqbc: optimizing query block SEL$58CDACD2 (#2)

      :
      call(in-use=95416, alloc=147368), compile(in-use=369560,
      alloc=407064), execution(in-use=426420, alloc=429324)

kkoqbc-subheap (create addr=0x0C31C7EC)
*****
QUERY BLOCK TEXT
*****
Not available.
-----
QUERY BLOCK SIGNATURE
-----
signature (optimizer): qb_name=SEL$58CDACD2 nb_fros=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"
```

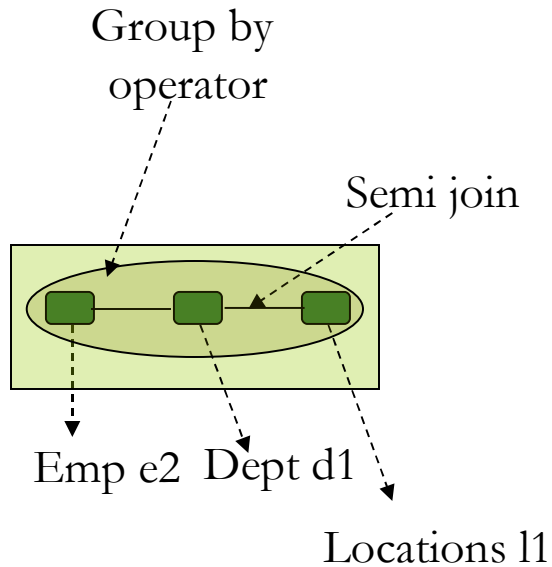
Group by operator applied after joining
All three tables in this transformation



```
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
```

Only sub query block is undergoing
Various transformation, for this SQL

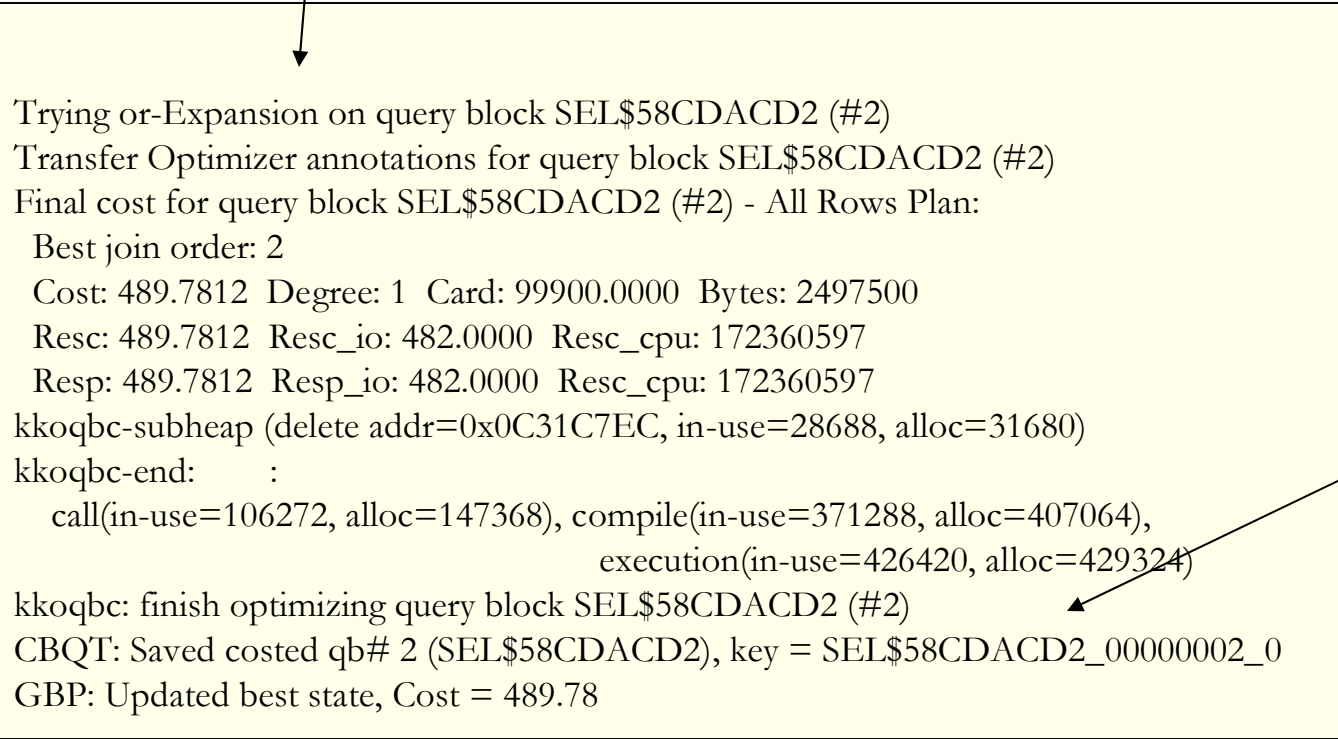
QBP#1



```
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
```

Transformation – GBP #1

Cost for this transformed query
calculated calling Kkoqbc module.



```
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=0x0C31C7EC, in-use=28688, alloc=31680)
kkoqbc-end:      :
  call(in-use=106272, alloc=147368), compile(in-use=371288, alloc=407064),
                                     execution(in-use=426420, alloc=429324)
kkoqbc: finish optimizing query block SEL$58CDACD2 (#2)
CBQT: Saved costed qb# 2 (SEL$58CDACD2), key = SEL$58CDACD2_00000002_0
GBP: Updated best state, Cost = 489.78
```

Cost saved

Transformation –GBP #2

```
GBP: Starting iteration 2, state space = (3,4,5) : (0,C,0)
Registered qb: SEL$24BEC10C 0x21ea6b00
  (QUERY BLOCK TABLES CHANGED SEL$58CDACD2)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL$24BEC10C nbfros=2 flg=0
  fro(0): flg=0 objn=71162 hint_alias="D1"@ "E2_INNER"
  fro(1): flg=0 objn=71160 hint_alias="L1"@ "L1_INNER"
```

```
Registered qb: SEL$6543C244 0x21ea0f64
  (SPLIT/MERGE QUERY BLOCKS SEL$24BEC10C)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL$6543C244 nbfros=1 flg=0
  fro(0): flg=0 objn=71164 hint_alias="E2"@ "E2_INNER"
```

```
Registered qb: SEL$E003ED3F 0x21ea6b00
UNKNOWN QUERY BLOCK ORIGIN
  SEL$58CDACD2; SEL$58CDACD2; LIST 2)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL$E003ED3F nbfros=3 flg=0
  fro(0): flg=0 objn=71162 hint_alias="D1"@ "E2_INNER"
  fro(1): flg=0 objn=71160 hint_alias="L1"@ "L1_INNER"
  fro(2): flg=5 objn=0 hint_alias="VW_GBC_2"@ "SEL$E003ED3F"
```

In this transformation,
group by operator applied to
emp.
Dept & locations joined later
To create a transformed query

```
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
```


Transformation –GBP #2

GBP: Starting iteration 2, state space = (3,4,5) : (0,C,0)
Registered qb: SEL\$24BEC10C 0x21ea6b00
(QUERY BLOCK TABLES CHANGED SEL\$58CDACD2)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL\$24BEC10C nbfros=2 flg=0
fro(0): flg=0 objn=71162 hint_alias="D1"@ "E2_INNER"
fro(1): flg=0 objn=71160 hint_alias="L1"@ "L1_INNER"

2nd iteration. State space bitmap indicates what row sources are changed in this transformation.

D1 & L1 pulled out and a new QB Created.

Registered qb: SEL\$6543C244 0x21ea0f64
(SPLIT/MERGE QUERY BLOCKS SEL\$24BEC10C)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL\$6543C244 nbfros=1 flg=0
fro(0): flg=0 objn=71164 hint_alias="E2"@ "E2_INNER"

E2 moved in to a new query block and Group by operator applied in that block. A new view also added.

Registered qb: SEL\$E003ED3F 0x21ea6b00
UNKNOWN QUERY BLOCK ORIGIN
SEL\$58CDACD2; SEL\$58CDACD2; LIST 2)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL\$E003ED3F nbfros=3 flg=0
fro(0): flg=0 objn=71162 hint_alias="D1"@ "E2_INNER"
fro(1): flg=0 objn=71160 hint_alias="L1"@ "L1_INNER"
fro(2): flg=5 objn=0 hint_alias=
"VW_GBC_2"@ "SEL\$E003ED3F"

These query blocks joined to create Transformed query

Transformation –GBP #2

GBP: Costing transformed query.

CBQT: Looking for cost annotations for query block SEL\$6543C244, key = SEL\$6543C244_00001200_3

CBQT: Could not find stored cost annotations.

kkoqbc: optimizing query block SEL\$6543C244 (#3)

Cost of inner subquery is calculated here

QUERY BLOCK SIGNATURE

signature (optimizer): qb_name=SEL\$6543C244 nbfros=1 flg=0

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

Trying or-Expansion on query block SEL\$6543C244 (#3)

Transfer Optimizer annotations for query block SEL\$6543C244 (#3)

Final cost for query block SEL\$6543C244 (#3) - All Rows Plan:

Best join order: 1

Cost: 514.8488 Degree: 1 Card: 100000.0000 Bytes: 3000000

Resc: 514.8488 Resc_io: 509.0000 Resc_cpu: 129556005

Resp: 514.8488 Resp_io: 509.0000 Resc_cpu: 129556005

kkoqbc-subheap (delete addr=0x0C31E560, in-use=9860, alloc=12356)

kkoqbc-end:

Transformation –GBP #2

QUERY BLOCK SIGNATURE

signature (optimizer): qb_name=SEL\$E003ED3F nbfros=3 flg=0
fro(0): flg=0 objn=71162 hint_alias="D1"@ "E2_INNER"
fro(1): flg=0 objn=71160 hint_alias="L1"@ "L1_INNER"
fro(2): flg=1 objn=0 hint_alias="VW_GBC_2"@ "SEL\$E003ED3F"

Cost of whole subquery calculated
Joining D1 & L1

Trying or-Expansion on query block SEL\$E003ED3F (#2)

Transfer Optimizer annotations for query block SEL\$E003ED3F (#2)

Final cost for query block SEL\$E003ED3F (#2) - All Rows Plan:

Best join order: 3

Cost: 672.7886 Degree: 1 Card: 9990.0000 Bytes: 589410

Resc: 672.7886 Resc_io: 664.0000 Resc_cpu: 194675802

Resp: 672.7886 Resp_io: 664.0000 Resc_cpu: 194675802

kkoqbc-subheap (delete addr=0x0C31F224, in-use=28556, alloc=31680)

kkoqbc-end: :

call(in-use=124192, alloc=180120), compile(in-use=404584, alloc=447744), execution(in-use=468220, alloc=470204)

kkoqbc: finish optimizing query block SEL\$E003ED3F (#2)

CBQT: Saved costed qb# 3 (SEL\$6543C244), key = SEL\$6543C244_00001200_3

CBQT: Saved costed qb# 2 (SEL\$E003ED3F), key = SEL\$E003ED3F_00000042_0

GBP: Updated best state, Cost = 672.79

Final cost of whole query saved.

Transformation –GBP #3

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

Registered qb: SEL\$50B3ADB4 0x21e9dd44
(QUERY BLOCK TABLES CHANGED SEL\$58CDACD2)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL\$50B3ADB4 nbfros=1 flg=0
fro(0): flg=0 objn=71164 hint_alias="E2"@ "E2_INNER"

Registered qb: SEL\$22220E47 0x21ea1be4
(SPLIT/MERGE QUERY BLOCKS SEL\$50B3ADB4)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL\$22220E47 nbfros=2 flg=0
fro(0): flg=0 objn=71162 hint_alias="D1"@ "E2_INNER"
fro(1): flg=0 objn=71160 hint_alias="L1"@ "L1_INNER"

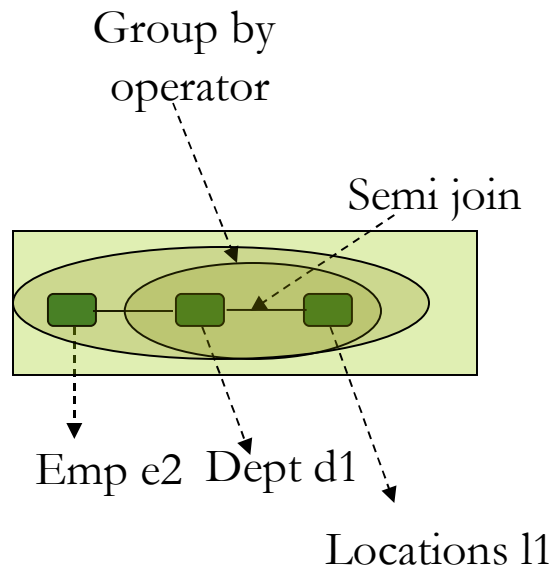
Registered qb: SEL\$35B1C696 0x21e9dd44
(UNKNOWN QUERY BLOCK ORIGIN
SEL\$58CDACD2; SEL\$58CDACD2; LIST 3)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL\$35B1C696 nbfros=2 flg=0
fro(0): flg=0 objn=71164 hint_alias="E2"@ "E2_INNER"
fro(1): flg=5 objn=0 hint_alias="VW_GBF_3"@ "SEL\$35B1C696"

Group by operator applied at dept and locations

Second Group by operator applied
After joining emp

```
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#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
```

Transformation –GBP #3

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

Registered qb: SEL\$50B3ADB4 0x21e9dd44
(QUERY BLOCK TABLES CHANGED SEL\$58CDACD2)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL\$50B3ADB4 nbfros=1 flg=0
fro(0): flg=0 objn=71164 hint_alias="E2"@ "E2_INNER"

← Emp e2 is moved and a new QB Created.

Registered qb: SEL\$22220E47 0x21ea1be4
(SPLIT/MERGE QUERY BLOCKS SEL\$50B3ADB4)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL\$22220E47 nbfros=2 flg=0
fro(0): flg=0 objn=71162 hint_alias="D1"@ "E2_INNER"
fro(1): flg=0 objn=71160 hint_alias="L1"@ "L1_INNER"

← D1 & L1 moved in to a new view and Group by operator applied. A new QB Also created.

Registered qb: SEL\$35B1C696 0x21e9dd44
(UNKNOWN QUERY BLOCK ORIGIN
SEL\$58CDACD2; SEL\$58CDACD2; LIST 3)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL\$35B1C696 nbfros=2 flg=0
fro(0): flg=0 objn=71164 hint_alias="E2"@ "E2_INNER"
fro(1): flg=5 objn=0 hint_alias="VW_GBF_3"@ "SEL\$35B1C696"

← New query block created, e2 and the View joined to create this query block.

Transformation –GBP #3

FPD applied and query shown earlier is the resultant of FPD & GBP



FPD: Considering simple filter push in query block SEL\$35B1C696 (#2)

"E2"."DEPT_ID"="VW_GBF_3"."ITEM_1"

try to generate transitive predicate from check constraints for query block SEL\$35B1C696 (#2)

finally: "E2"."DEPT_ID"="VW_GBF_3"."ITEM_1"

kkqfppRelFilter: Not pushing filter predicates in query block SEL\$22220E47 (#4) because no predicate to push

FPD: Considering simple filter push in query block SEL\$22220E47 (#4)

"L1"."LOCATION_ID"="D1"."LOCATION_ID"

try to generate transitive predicate from check constraints for query block SEL\$22220E47 (#4)

finally: "L1"."LOCATION_ID"="D1"."LOCATION_ID"

GBP: Costing transformed query.

CBQT: Looking for cost annotations for query block SEL\$22220E47, key = SEL\$22220E47_00001200_2

CBQT: Could not find stored cost annotations.

kkoqbc: optimizing query block SEL\$22220E47 (#4)

Transformation –GBP #3

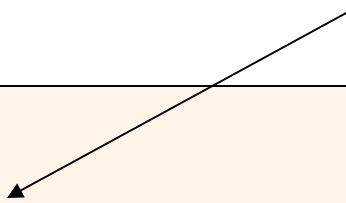
Inner subquery with an alias vw_gbf_3
Costed here.

```
signature (optimizer): qb_name=SEL$22220E47 nbfros=2 flg=0  
fro(0): flg=0 objn=71162 hint_alias="D1"@ "E2_INNER"  
fro(1): flg=0 objn=71160 hint_alias="L1"@ "L1_INNER"
```

```
...  
Trying or-Expansion on query block SEL$22220E47 (#4)  
Transfer Optimizer annotations for query block SEL$22220E47 (#4)  
Final cost for query block SEL$22220E47 (#4) - All Rows Plan:  
Best join order: 1  
Cost: 52.0502 Degree: 1 Card: 9990.0000 Bytes: 159840  
Resc: 52.0502 Resc_io: 51.0000 Resc_cpu: 23263193  
Resp: 52.0502 Resp_io: 51.0000 Resc_cpu: 23263193  
kkoqbc-end:
```

Transformation –GBP #3

Whole query is costed here



QUERY BLOCK SIGNATURE

```
signature (optimizer): qb_name=SEL$35B1C696 nbfros=2 flg=0
fro(0): flg=0 objn=71164 hint_alias="E2"@ "E2_INNER"
fro(1): flg=1 objn=0 hint_alias="VW_GBF_3"@ "SEL$35B1C696"
```

Trying or-Expansion on query block SEL\$35B1C696 (#2)

Transfer Optimizer annotations for query block SEL\$35B1C696 (#2)

Final cost for query block SEL\$35B1C696 (#2) - All Rows Plan:

Best join order: 1

Cost: 727.0640 Degree: 1 Card: 99900.0000 Bytes: 4795200

Resc: 727.0640 Resc_io: 719.0000 Resc_cpu: 178624021

Resp: 727.0640 Resp_io: 719.0000 Resc_cpu: 178624021

kkoqbc-subheap (delete addr=0x0C2FBC08, in-use=18784, alloc=22500)

kkoqbc-end:

kkoqbc: finish optimizing query block SEL\$35B1C696 (#2)

CBQT: Saved costed qb# 4 (SEL\$22220E47), key = SEL\$22220E47_00001200_2

CBQT: Saved costed qb# 2 (SEL\$35B1C696), key = SEL\$35B1C696_00000042_0

GBP: Updated best state, Cost = 727.06

Transformation –GBP #4

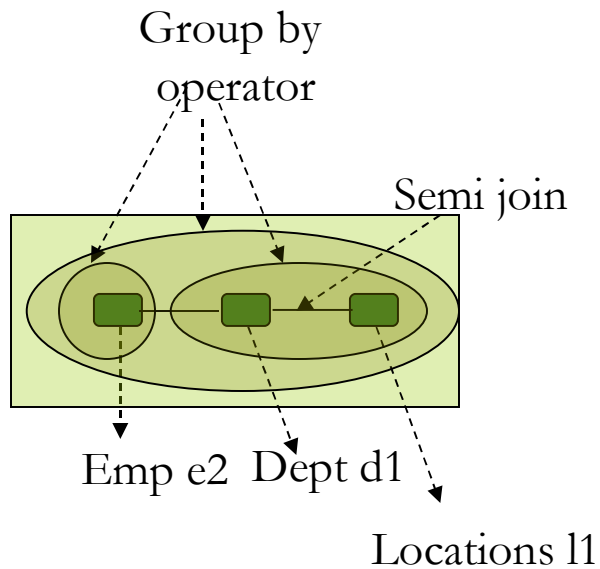
```
GBP: Starting iteration 4, state space = (3,4,5) : (F,C,F)
Registered qb: SEL$C54E6AB0 0x21eb8788
  (QUERY BLOCK TABLES CHANGED SEL$24BEC10C)
QUERY BLOCK SIGNATURE
signature (): qb_name=SEL$C54E6AB0 nbfros=1 flg=0
fro(0): flg=5 objn=0 hint_alias="VW_GBC_4"@SEL$C54E6AB0"
```

```
Registered qb: SEL$C43B7E12 0x21eb8788
  (UNKNOWN QUERY BLOCK ORIGIN SEL$58CDACD2;
  SEL$58CDACD2; LIST LIST 4)
-----
QUERY BLOCK SIGNATURE
-----
signature (): qb_name=SEL$C43B7E12 nbfros=2 flg=0
fro(0): flg=5 objn=0 hint_alias="VW_GBF_5"@SEL$C43B7E12"
fro(1): flg=1 objn=0 hint_alias="VW_GBC_4"@SEL$C54E6AB0"
```

Grouping done at two different levels.

```
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
```

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
```

Transformation –GBP #4

FPD: Considering simple filter push in query block SEL\$C43B7E12 (#2)

"VW_GBC_4"."ITEM_1"="VW_GBF_5"."ITEM_1"

try to generate transitive predicate from check constraints for query block SEL\$C43B7E12 (#2)

finally: "VW_GBC_4"."ITEM_1"="VW_GBF_5"."ITEM_1"

kkqfppRelFilter: Not pushing filter predicates in query block SEL\$7E9F6985 (#6) because no predicate to push

FPD: Considering simple filter push in query block SEL\$7E9F6985 (#6)

"L1"."LOCATION_ID"="D1"."LOCATION_ID"

try to generate transitive predicate from check constraints for query block SEL\$7E9F6985 (#6)

finally: "L1"."LOCATION_ID"="D1"."LOCATION_ID"

kkqfppRelFilter: Not pushing filter predicates in query block SEL\$6543C244 (#5) because no predicate to push

FPD: Considering simple filter push in query block SEL\$6543C244 (#5)

GBP: Costing transformed query.

CBQT: Looking for cost annotations for query block SEL\$6543C244, key = SEL\$6543C244_00001200_2

CBQT: Could not find stored cost annotations.

CBQT: Looking for cost annotations for query block SEL\$7E9F6985, key = SEL\$7E9F6985_00001200_2

CBQT: Could not find stored cost annotations.

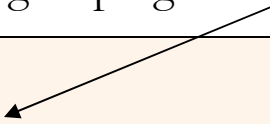
kkqbc: optimizing query block SEL\$6543C244 (#5)

Optimizing transformed query

Transformation –GBP #4

SQL branch grouping table with E2 alias costed.

```
signature (optimizer): qb_name=SEL$6543C244 nbfros=1 flg=0  
fro(0): flg=0 objn=71164 hint_alias="E2"@ "E2_INNER"
```



```
...  
Trying or-Expansion on query block SEL$6543C244 (#5)  
Transfer Optimizer annotations for query block SEL$6543C244 (#5)  
Final cost for query block SEL$6543C244 (#5) - All Rows Plan:  
Best join order: 1  
Cost: 514.8488 Degree: 1 Card: 100000.0000 Bytes: 3000000  
Resc: 514.8488 Resc_io: 509.0000 Resc_cpu: 129556005  
Resp: 514.8488 Resp_io: 509.0000 Resc_cpu: 129556005  
kkoqbc-subheap (delete addr=0x0C329494, in-use=9860, alloc=12356)  
kkoqbc-end:
```

Transformation –GBP #4

SQL branch grouping table with D1 & L1 aliases costed.

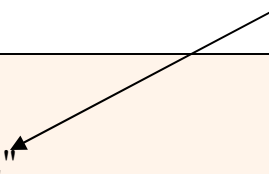
QUERY BLOCK SIGNATURE

```
-----  
signature (optimizer): qb_name=SEL$7E9F6985 nbfros=2 flg=0  
fro(0): flg=0 objn=71162 hint_alias="D1"@ "E2_INNER"  
fro(1): flg=0 objn=71160 hint_alias="L1"@ "L1_INNER"
```

```
...  
Trying or-Expansion on query block SEL$7E9F6985 (#6)  
Transfer Optimizer annotations for query block SEL$7E9F6985 (#6)  
Final cost for query block SEL$7E9F6985 (#6) - All Rows Plan:  
Best join order: 1  
Cost: 52.0502 Degree: 1 Card: 9990.0000 Bytes: 159840  
Resc: 52.0502 Resc_io: 51.0000 Resc_cpu: 23263193  
Resp: 52.0502 Resp_io: 51.0000 Resc_cpu: 23263193  
kkoqbc-subheap (delete addr=0x0C32A158, in-use=19164, alloc=22500)  
kkoqbc-end:
```

Transformation –GBP #4

Join between these two
Group by branches



```
signature (optimizer): qb_name=SEL$C43B7E12 nbfros=2 flg=0  
fro(0): flg=1 objn=0 hint_alias="VW_GBF_5"@SEL$C43B7E12"  
fro(1): flg=1 objn=0 hint_alias="VW_GBC_4"@SEL$C54E6AB0"
```

Trying or-Expansion on query block SEL\$C43B7E12 (#2)

Transfer Optimizer annotations for query block SEL\$C43B7E12 (#2)

Final cost for query block SEL\$C43B7E12 (#2) - All Rows Plan:

Best join order: 1

Cost: 749.9118 Degree: 1 Card: 9990.0000 Bytes: 769230

Resc: 749.9118 Resc_io: 741.0000 Resc_cpu: 197405395

Resp: 749.9118 Resp_io: 741.0000 Resc_cpu: 197405395

kkoqbc-subheap (delete addr=0x0C324424, in-use=18652, alloc=22500)

kkoqbc-end:

:

call(in-use=167804, alloc=212872), compile(in-use=509756, alloc=517176), execution(in-use=540080, alloc=544052)

kkoqbc: finish optimizing query block SEL\$C43B7E12 (#2)

CBQT: Saved costed qb# 5 (SEL\$6543C244), key = SEL\$6543C244_00001200_2

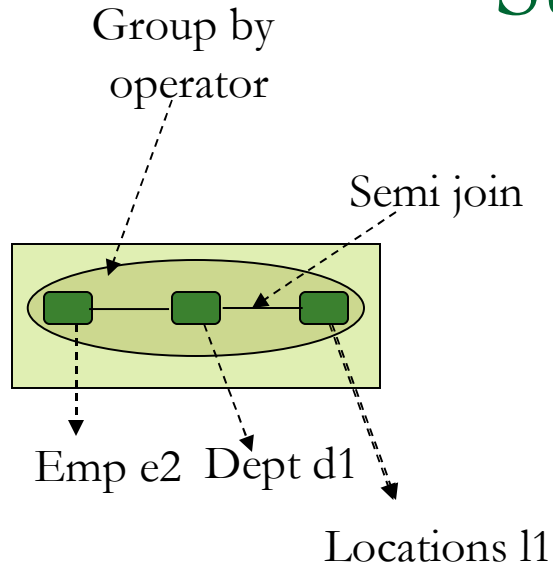
CBQT: Saved costed qb# 6 (SEL\$7E9F6985), key = SEL\$7E9F6985_00001200_2

CBQT: Saved costed qb# 2 (SEL\$C43B7E12), key = SEL\$C43B7E12_00000042_0

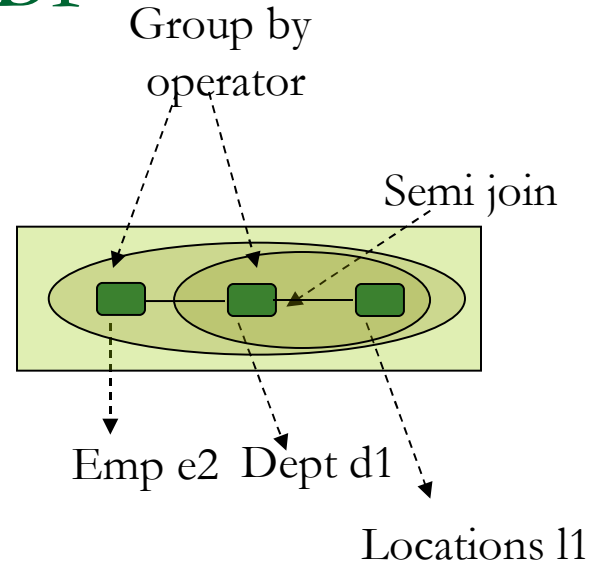
GBP: Updated best state, Cost = 749.91

Summary QBP

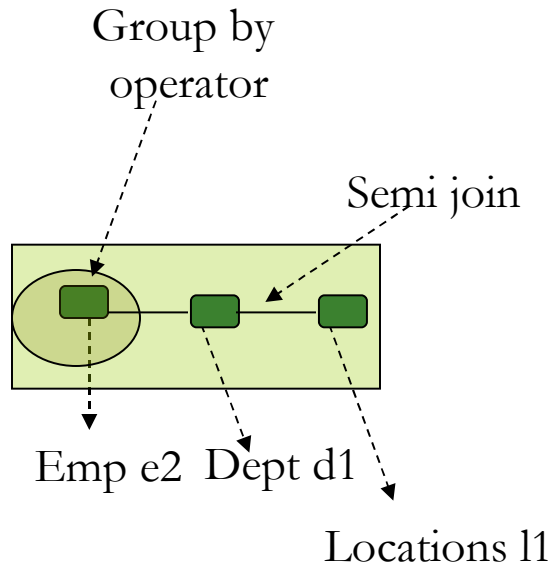
#1



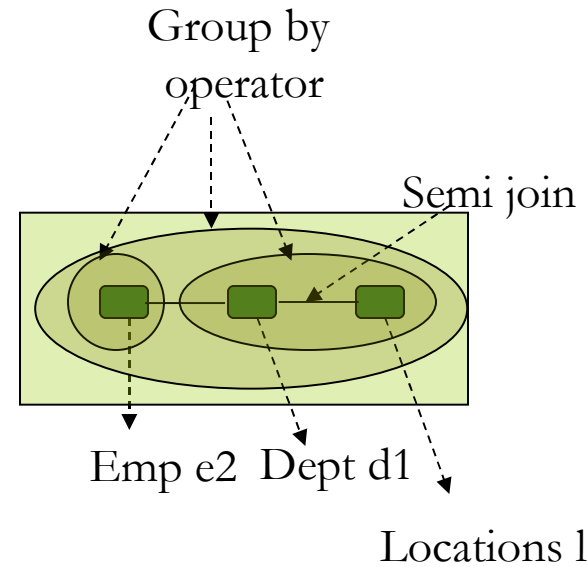
#3



#2



#4



Cost-Based Join Predicate Push-down

JPPD: Checking validity of push-down in query block SEL\$825981F0 (#1)

JPPD: Checking validity of push-down from query block SEL\$825981F0 (#1) to query block SEL\$58CDACD2 (#2)

Check Basic Validity for Non-Union View for query block SEL\$58CDACD2 (#2)

JPPD: JPPD bypassed: No valid join condition found.

JPPD: No valid views found to push predicate into.

JPPD: Applying transformation directives

JPPD: Checking validity of push-down in query block SEL\$825981F0 (#1)

JPPD: No valid views found to push predicate into.

query block E1_OUTER transformed to SEL\$825981F0 (#1)

FPD: Considering simple filter push in query block SEL\$825981F0 (#1)

"E1"."SALARY">"VW_SQ_1"."AVG(SALARY)" AND "E1"."DEPT_ID"="VW_SQ_1"."ITEM_1"
AND "E1"."HIRE_DATE">SYSDATE@!-3650

try to generate transitive predicate from check constraints for query block SEL\$825981F0 (#1)

finally: "E1"."SALARY">"VW_SQ_1"."AVG(SALARY)" AND "E1"."DEPT_ID"="VW_SQ_1"."ITEM_1" AND
"E1"."HIRE_DATE">SYSDATE@!-3650

kkqfppRelFilter: Not pushing filter predicates in query block SEL\$58CDACD2 (#2) because no predicate to push

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"

SJC : Set Join Conversion

Generated predicates

Intersect set operation converted to a join.

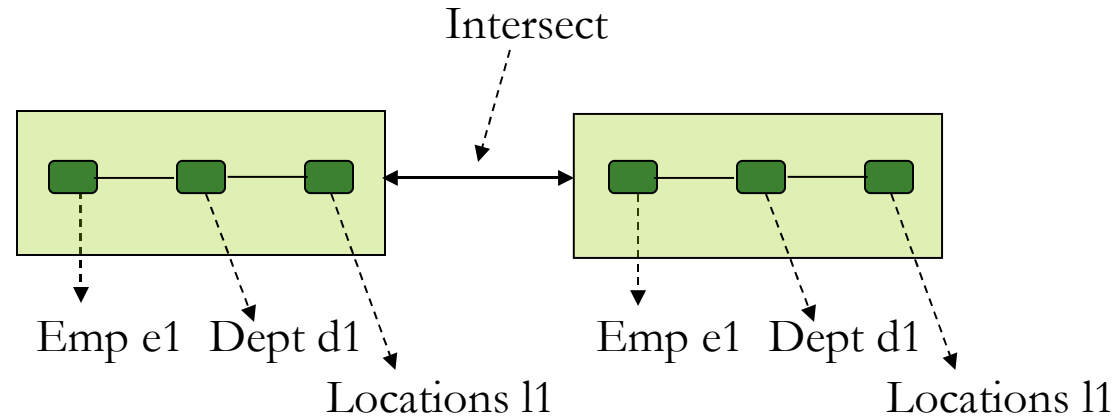
```
Select /*+ qb_name (e1_outer) */ * from
emp e1, dept d1, locations l1
where
e1.dept_id = d1.dept_id and
d1.location_id = l1.location_id and
e1.salary > 100000
intersect
Select /*+ qb_name (e2_outer) */ * from
emp e2 , dept d2, locations l2
where
e2.dept_id = d2.dept_id and
d2.location_id = l2.location_id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



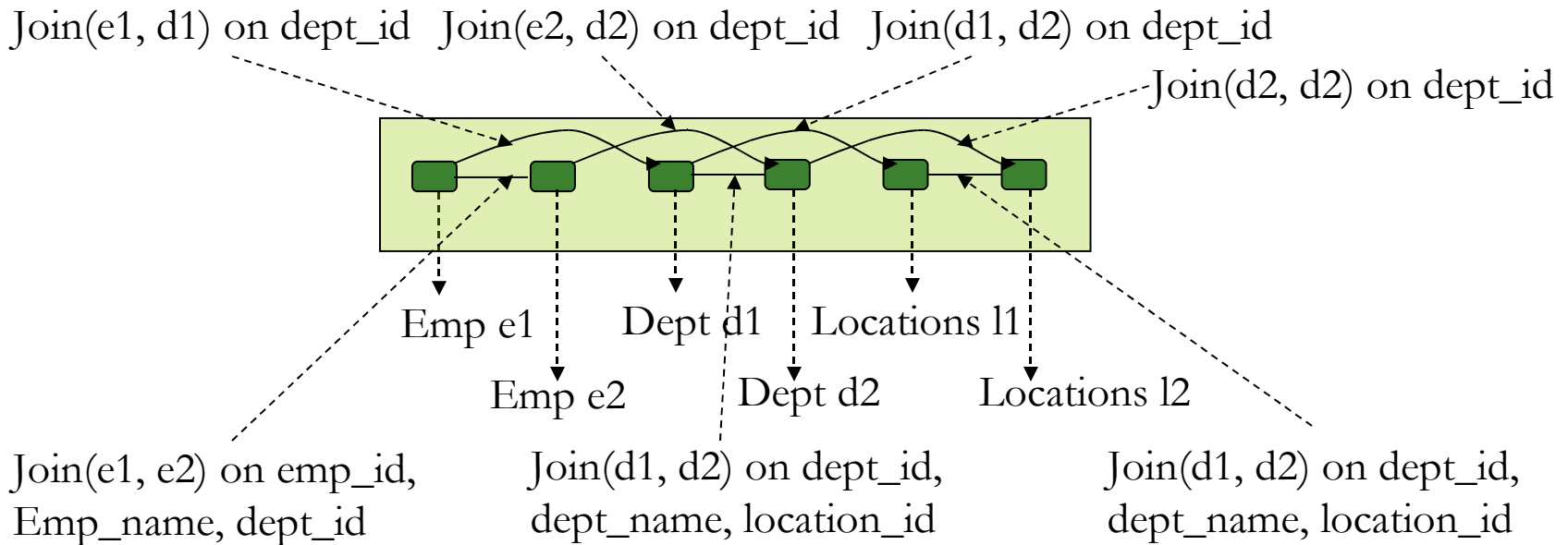
```
select distinct e1.emp_id, e1.emp_name, e1.dept_id,
e1.salary, e1.hire_Date, d1.dept_id, d1.dept_name,
d1.location_id, l1.location_id, l1.city_name, l1.state
from
emp e2, dept d2, locations l2, emp e1, dept d1, locations l1
where
e1.emp_id = e2.emp_id and
sys_op_map_nonnull(e1.emp_name) =
sys_op_map_nonnull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

→ SJC is disabled by default . Parameter `_convert_set_to_join` enables SJC.

```
SJC: Considering set-join conversion in query block SET$1 (#0)
```

```
*****
```

```
Set-Join Conversion (SJC)
```

```
*****
```

```
SJC: Checking validity of SJC on query block SET$1 (#0)
```

```
SJC: Passed validity checks.
```

```
SJC: SJC: Applying SJC on query block SET$1 (#0)
```

```
Registered qb: SET$09AAA538 0x1ded2d00 (SET QUERY BLOCK SET$1; SET$1)
```

```
-----  
QUERY BLOCK SIGNATURE  
-----
```

```
signature (): qb_name=SET$09AAA538 nbfros=2 flg=0
```

```
fro(0): flg=1 objn=0 hint_alias="NULL_HALIAS"@SET$09AAA538"
```

```
fro(1): flg=1 objn=0 hint_alias="NULL_HALIAS"@SET$09AAA538"
```

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"

Join Predicate Push-down

```
explain plan for
Select /*+ qb_name (v_outer) */ v1.* from
  ( select /*+ qb_name (v1) */ e1.*, d1.location_id from
    emp e1, dept d1
    where e1.dept_id = d1.dept_id ) v1,
  (select/*+ qb_name (v2) */
    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
    group by d2.dept_id
  ) v2
where
  v1.dept_id=v2.dept_id and v1.salary > v2.avg_sal_dept
and v1.dept_id=100
```

Join predicates between v1 and v2



JPPD: Starting iteration 2, state space = (2) : (1)

JPPD: Performing join predicate push-down (candidate phase) from query block SEL\$456895EC (#1)
to query block V2 (#2)

JPPD: Pushing predicate "E1"."DEPT_ID"="V2"."DEPT_ID" from query block SEL\$456895EC (#1)
to query block V2 (#2)

JPPD: Push dest of pred 0x20DBC274 is qb 0x1EB38B5C:query block V2 (#2)

Registered qb: SEL\$F7BD14DD 0x1eb38b5c (PUSHED PREDICATE V2; SEL\$456895EC; "V2"@ "V_OUTER" 2)

QUERY BLOCK SIGNATURE

signature (): qb_name=SEL\$F7BD14DD nbfros=3 flg=0
fro(0): flg=0 objn=73174 hint_alias="D2"@ "V2"
fro(1): flg=0 objn=73176 hint_alias="E2"@ "V2"
fro(2): flg=0 objn=73172 hint_alias="L2"@ "V2"

JPPD: Costing transformed query.

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)

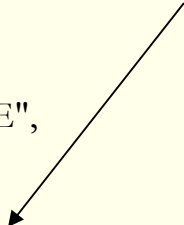
JE: cfro: DEPT objn:73501 col#:3 dfro:LOCATIONS dcol#:3

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

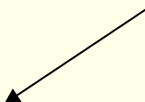
(JOIN REMOVED FROM QUERY BLOCK SEL\$5DA710D3; SEL\$5DA710D3; "L1"@SEL\$2")

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]

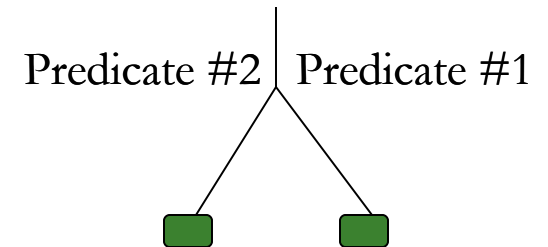
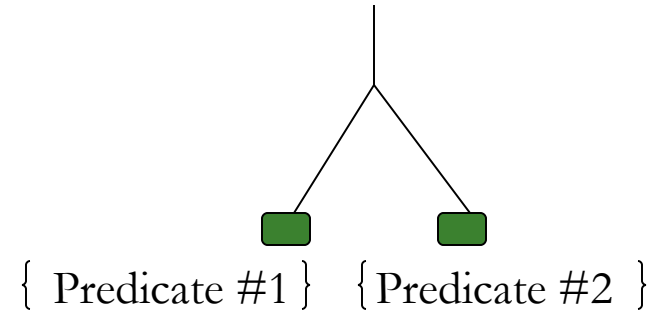
Predicate pull up

- Tree traversed bottom-up

Predicate push down

- Tree traversed top-down

Duplicate predicate removal



PM

Predicate Move-Around (PM)

PM: Passed validity checks.

PM: Pulled up predicate "V1"."DEPT_ID "=300

from query block V1 (#2) to query block V_OUTER (#1)

PM: Pulled up predicate ("V1"."EMP_ID", "E1"."DEPT_ID ")=300

from query block V1 (#2) to query block V_OUTER (#1)

PM: Pulled up predicate "V1"."DEPT_ID "=200

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

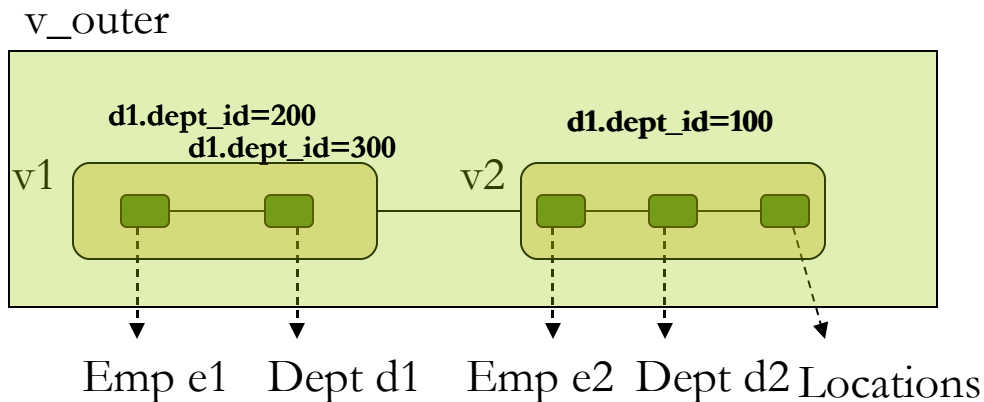
from query block V_OUTER (#1) to query block V2 (#3)

PM

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

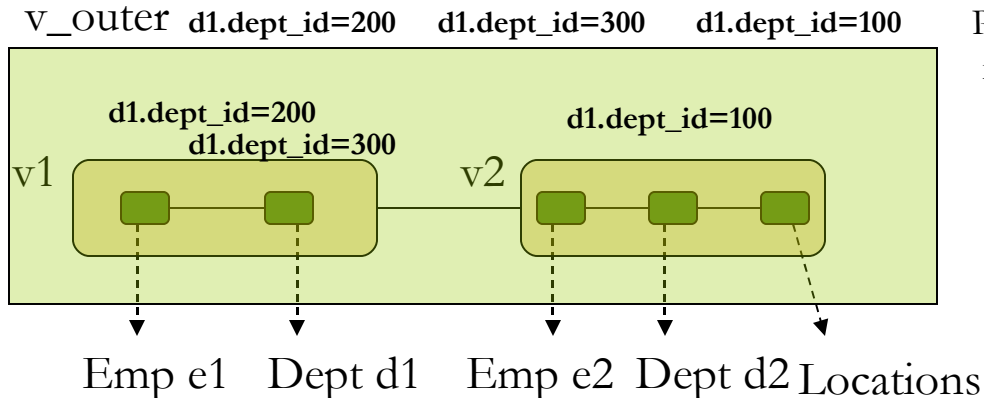
PM: Pulled up predicate "V1"."DEPT_ID"=300
from query block V1 (#2) to query block V_OUTER (#1)

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



12

PM



12

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
```

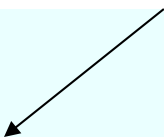
Query transformations (QT)

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 Joze Senegacnik, UKOUG 2007