Introduction to SQL Tuning

Zoran Pavlović, Oracle Technical Architect, Parallel
Maja Veselica, Consultant, Parallel

Contact

Zoran Pavlović, *Oracle Technical Architect*
z.m.pavlovic@gmail.com

Twitter: ChallengeZoran

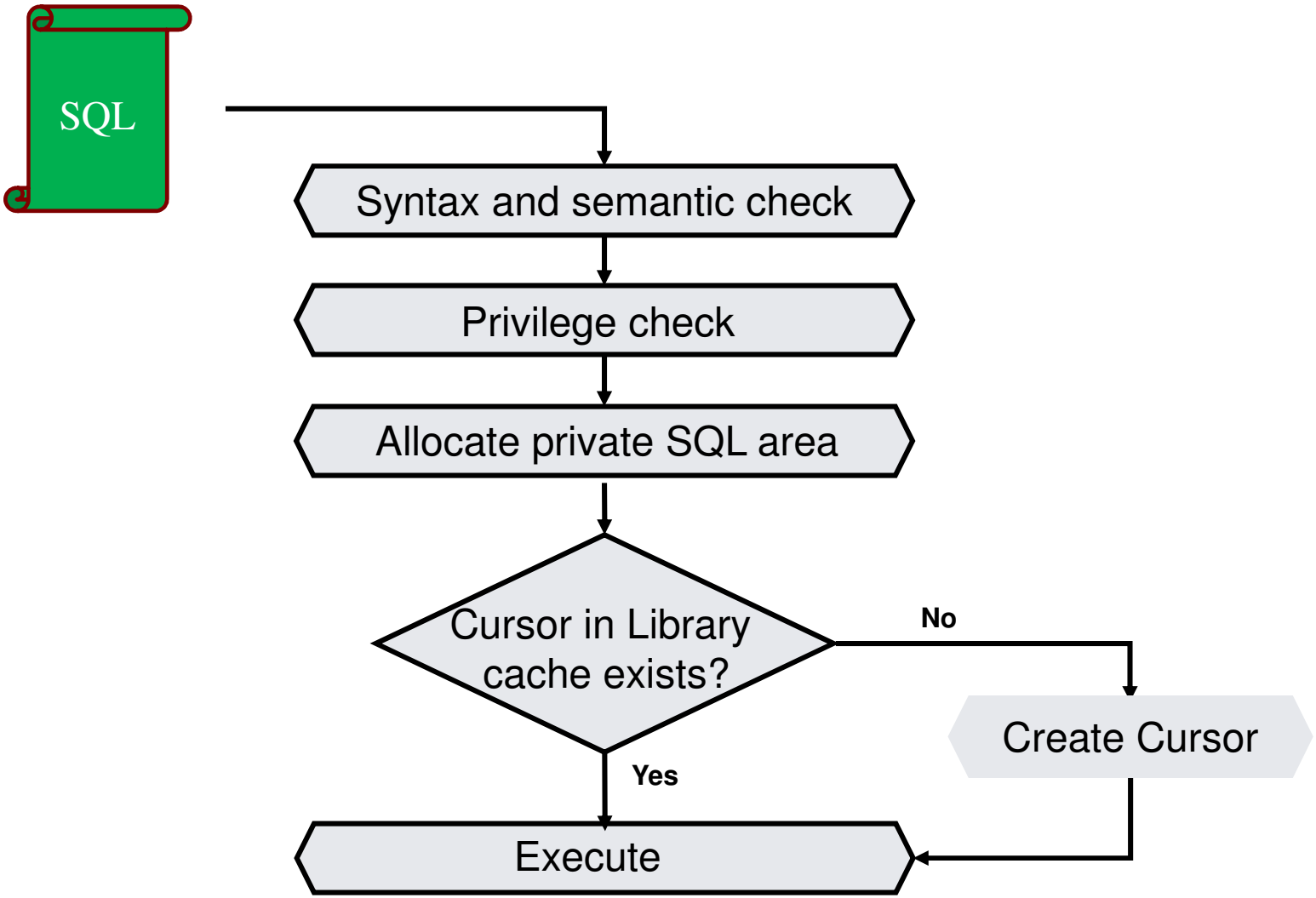
Maja Veselica, *Consultant*
maja.veselica@gmail.com

Twitter: orapassion

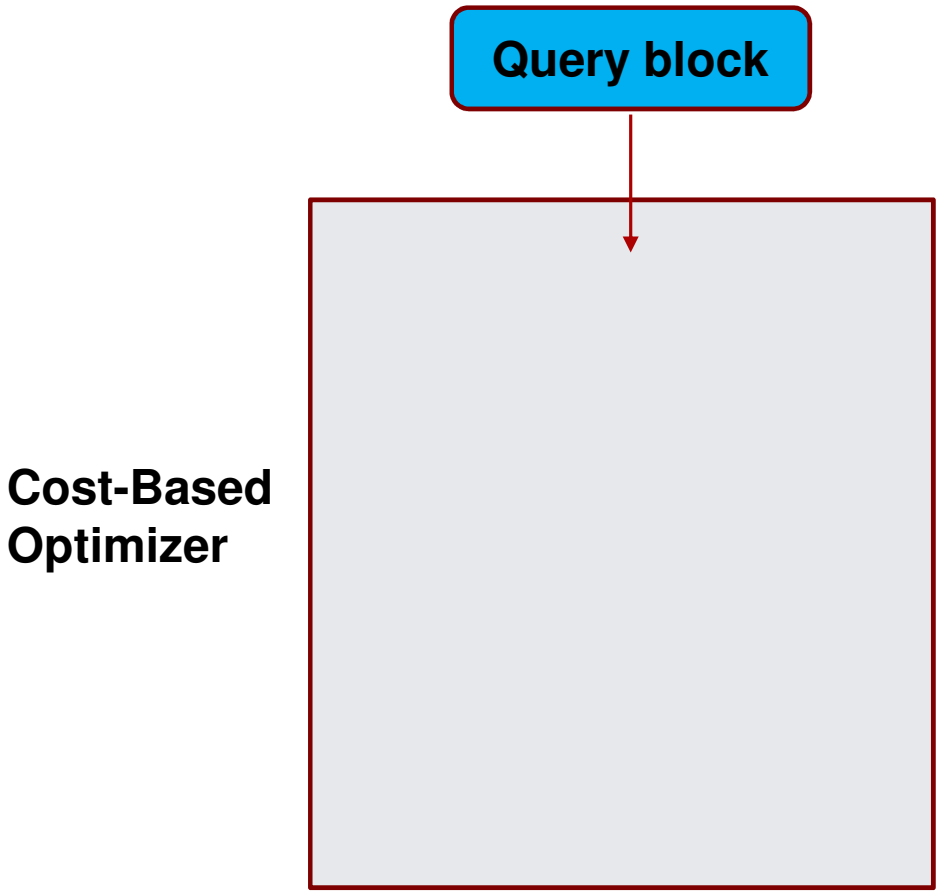
Our Websites

www.challengezoran.com – forum
www.orapassion.com – blog
www.parallel.rs - Company

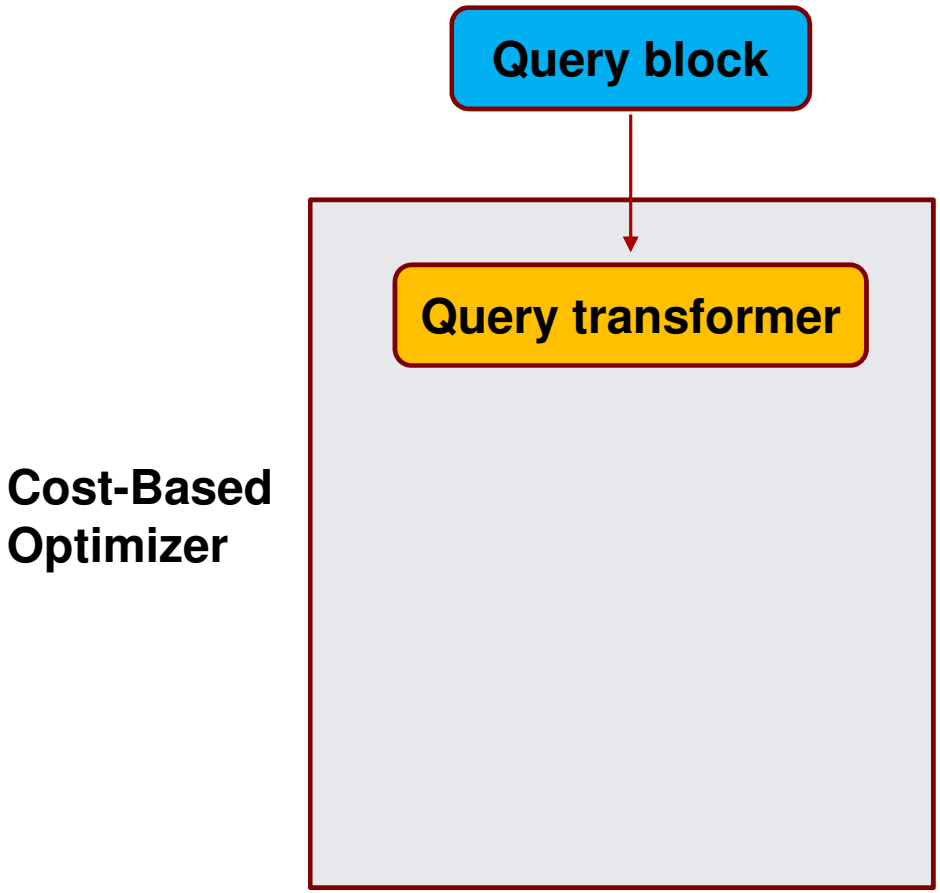
SQL Execution



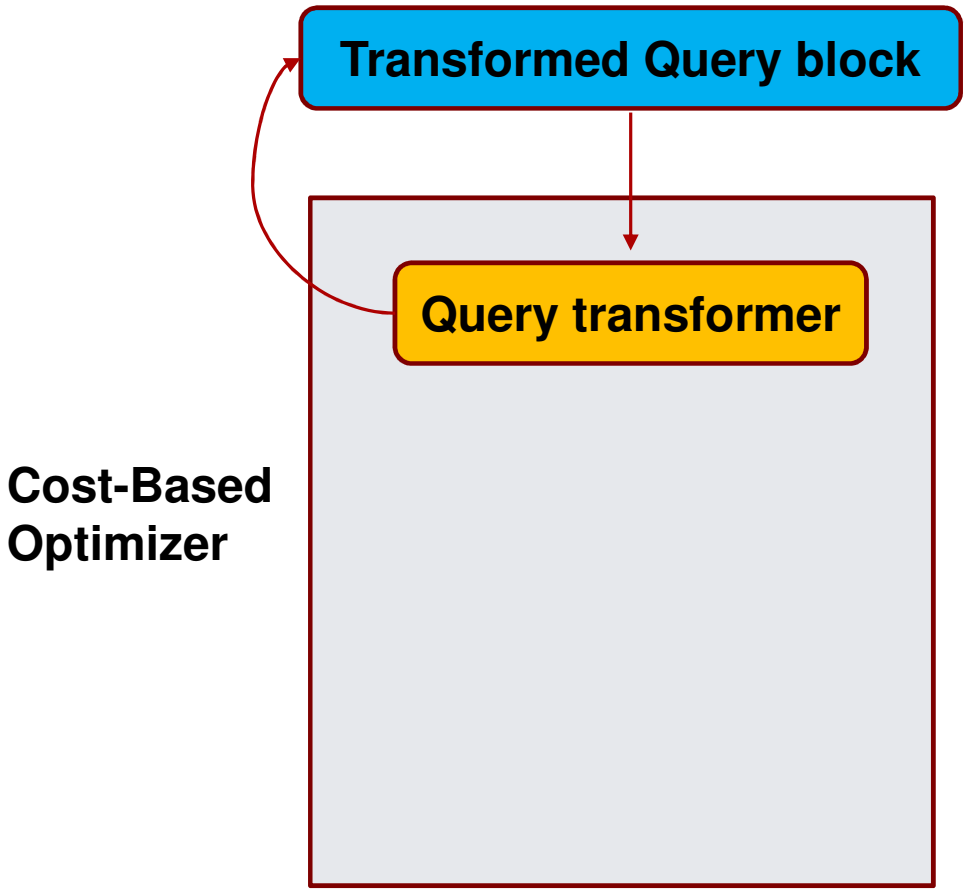
Cost-Based Optimizer



Cost-Based Optimizer



Cost-Based Optimizer



Example 1 - Query Transformer - Expand

Original Query block - SEL$1

```
SELECT FIRST_NAME, LAST_NAME, EMAIL, SALARY
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 60 OR MANAGER_ID = 100;
```

Example 1 - Query Transformer - Expand

Original Query block - SEL$1

```
SELECT FIRST_NAME, LAST_NAME, EMAIL, SALARY
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 60 OR MANAGER_ID = 100;
```

```
-----------------------------------------------------------------------------------------------------
| Id  | Operation                            | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |                    |       |       |     2 (100)|          |
|   1 |  CONCATENATION                       |                    |       |       |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED| EMPLOYEES          |     5 |   170 |     1   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN                  | EMP_DEPARTMENT_IX  |     1 |       |     1   (0)| 00:00:01 |
|*  4 |   TABLE ACCESS BY INDEX ROWID BATCHED| EMPLOYEES          |    13 |   442 |     1   (0)| 00:00:01 |
|*  5 |    INDEX RANGE SCAN                  | EMP_MANAGER_IX     |     1 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("DEPARTMENT_ID"=60)
   4 - filter(LNNVL("DEPARTMENT_ID"=60))
   5 - access("MANAGER_ID"=100)
```

Example 1 - Query Transformer - Expand

Original Query block - SEL$1

```
SELECT FIRST_NAME, LAST_NAME, EMAIL, SALARY
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 60 OR MANAGER_ID = 100;
```


Transformed Query block - SEL$1

```
SELECT FIRST_NAME, LAST_NAME, EMAIL, SALARY
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 60
UNION ALL
SELECT FIRST_NAME, LAST_NAME, EMAIL, SALARY
FROM EMPLOYEES
WHERE MANAGER_ID = 100 AND DEPARTMENT_ID <> 60;
```

Example 1 - Query Transformer – No_Expand

Original Query block - SEL$1

```
SELECT /*+ no_expand */ FIRST_NAME, LAST_NAME, EMAIL, SALARY
FROM EMPLOYEES
WHERE DEPARTMENT_ID = 60 OR MANAGER_ID = 100;
```

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |       |       |     3 (100)|          |
|*  1 |  TABLE ACCESS FULL| EMPLOYEES |    18 |   612 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(("MANAGER_ID"=100 OR "DEPARTMENT_ID"=60))
```

Example 2 - Query Transformer – Transitive predicate

Original Query block - SEL$1

```
SELECT FIRST_NAME, LAST_NAME, DEPARTMENT_NAME
FROM HR.EMPLOYEES E, HR.DEPARTMENTS D
WHERE E.DEPARTMENT_ID = 50
AND E.DEPARTMENT_ID = D.DEPARTMENT_ID;
```

Example 2 - Query Transformer – Transitive predicate

Original Query block - SEL$1

```
SELECT FIRST_NAME, LAST_NAME, DEPARTMENT_NAME
FROM HR.EMPLOYEES E, HR.DEPARTMENTS D
WHERE E.DEPARTMENT_ID = 50
AND E.DEPARTMENT_ID = D.DEPARTMENT_ID;
```

```
FPD: Considering simple filter push in query block SEL$1 (#0)
"E"."DEPARTMENT_ID"=50 AND "E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID"
try to generate transitive predicate from check constraints for query
block SEL$1 (#0)
finally: "E"."DEPARTMENT_ID"=50 AND
"E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID" AND "D"."DEPARTMENT_ID"=50

FPD:   transitive predicates are generated in query block SEL$1 (#0)
"E"."DEPARTMENT_ID"=50 AND "E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID" AND
"D"."DEPARTMENT_ID"=50
```

Example 2 - Query Transformer – Transitive predicate

Transformed Query block - SEL$1

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT "E"."FIRST_NAME" "FIRST_NAME","E"."LAST_NAME"
"LAST_NAME","D"."DEPARTMENT_NAME" "DEPARTMENT_NAME"
FROM "HR"."EMPLOYEES" "E","HR"."DEPARTMENTS" "D"
WHERE "E"."DEPARTMENT_ID"=50
AND "E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID"
AND "D"."DEPARTMENT_ID"=50
```

Example 2 - Query Transformer – Transitive predicate

```
-------------------------------------------------------------------------------------------
| Id  | Operation                            | Name               | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |                    |       |       |     1 (100)|
|   1 |  NESTED LOOPS                        |                    |    45 |  1530 |     0   (0)|
|   2 |   TABLE ACCESS BY INDEX ROWID        | DEPARTMENTS        |     1 |    16 |     0   (0)|
|*  3 |    INDEX UNIQUE SCAN                 | DEPT_ID_PK         |     1 |       |     0   (0)|
|   4 |   TABLE ACCESS BY INDEX ROWID BATCHED| EMPLOYEES          |    45 |   810 |     0   (0)|
|*  5 |    INDEX RANGE SCAN                  | EMP_DEPARTMENT_IX  |     1 |       |     0   (0)|
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("D"."DEPARTMENT_ID"=50)
   5 - access("E"."DEPARTMENT_ID"=50)
```

Example 3 - Query Transformer – View Merge

View – Query block SEL$2

```
CREATE VIEW V_ORD_07_08 AS
SELECT C.CUSTOMER_ID, C.CUST_FIRST_NAME, C.CUST_LAST_NAME,
C.CUST_ADDRESS, C.CREDIT_LIMIT, C.CUST_EMAIL, O.ORDER_DATE,
O.ORDER_STATUS, O.ORDER_TOTAL, P.PRODUCT_NAME, I.UNIT_PRICE
FROM OE.CUSTOMERS C, OE.ORDERS O, OE.ORDER_ITEMS I,
OE.PRODUCT_INFORMATION P
WHERE C.CUSTOMER_ID = O.CUSTOMER_ID
AND O.ORDER_ID = I.ORDER_ID
AND I.PRODUCT_ID = P.PRODUCT_ID
AND O.ORDER_DATE BETWEEN TO_DATE('01/01/2007',
'DD/MM/YYYY') AND TO_DATE('31/12/2008', 'DD/MM/YYYY');
```

Example 3 - Query Transformer – View Merge

Original Query block - SEL$1

```
SELECT customer_id, cust_first_name, cust_last_name,
cust_email, order_total
FROM v_ord_07_08
WHERE order_total > 4000;
```

Example 3 - Query Transformer – View Merge

Creating new query block `SEL$F5BB74E1` by view merge

```
CVM:   Merging SPJ view SEL$2 (#0) into SEL$1 (#0)
Registered qb: SEL$F5BB74E1 0xc946f0a8 (VIEW MERGE SEL$1; SEL$2)
```

Part of 10053 – Trace file

Example 3 - Query Transformer – View Merge

New query block - `SEL$F5BB74E1`

```
SELECT "C"."CUSTOMER_ID" "CUSTOMER_ID","C"."CUST_FIRST_NAME"
"CUST_FIRST_NAME","C"."CUST_LAST_NAME"
"CUST_LAST_NAME","C"."CUST_EMAIL" "CUST_EMAIL","O"."ORDER_TOTAL"
"ORDER_TOTAL"
FROM "OE"."CUSTOMERS" "C","OE"."ORDERS" "O","OE"."ORDER_ITEMS"
"I","OE"."PRODUCT_INFORMATION" "P"
WHERE "O"."ORDER_TOTAL">4000
AND "C"."CUSTOMER_ID"="O"."CUSTOMER_ID"
AND "O"."ORDER_ID"="I"."ORDER_ID"
AND "I"."PRODUCT_ID"="P"."PRODUCT_ID"
AND "O"."ORDER_DATE">=TO_DATE('01/01/2007','dd/mm/yyyy')
AND "O"."ORDER_DATE"<=TO_DATE('31/12/2008','dd/mm/yyyy')
```

Part of 10053 – Trace file

Example 3 - Query Transformer – Join Elimination

New query block - `SEL$F4A4219D`

```
JE: cfro: ORDER_ITEMS objn:93052 col#:3 dfro:PRODUCT_INFORMATION dcol#:1
JE: eliminate table: PRODUCT_INFORMATION (P)
.
.
.
Registered qb: SEL$F4A4219D 0xc946f0a8 (JOIN REMOVED FROM QUERY BLOCK
SEL$F5BB74E1; SEL$F5BB74E1; "P"@"SEL$2")
```

Part of 10053 – Trace file

Example 3 - Query Transformer – Join Elimination

New query block - `SEL$F4A4219D`

```
SQL:******* UNPARSED QUERY IS *******
SELECT "C"."CUSTOMER_ID" "CUSTOMER_ID","C"."CUST_FIRST_NAME"
"CUST_FIRST_NAME","C"."CUST_LAST_NAME"
"CUST_LAST_NAME","C"."CUST_EMAIL" "CUST_EMAIL","O"."ORDER_TOTAL"
"ORDER_TOTAL"
FROM "OE"."CUSTOMERS" "C","OE"."ORDERS" "O","OE"."ORDER_ITEMS" "I"
WHERE "O"."ORDER_TOTAL">4000
AND "C"."CUSTOMER_ID"="O"."CUSTOMER_ID"
AND "O"."ORDER_ID"="I"."ORDER_ID"
AND "O"."ORDER_DATE">=TO_DATE('01/01/2007','dd/mm/yyyy')
AND "O"."ORDER_DATE"<=TO_DATE('31/12/2008','dd/mm/yyyy')
```

Part of 10053 – Trace file

Example 3 - Query Transformer – Transitive predicate

New query block - `SEL$F5BB74E1`

```
try to generate transitive predicate from check constraints for query
block SEL$F4A4219D (#0)
constraint: "O"."ORDER_TOTAL">=0
constraint: "C"."CUSTOMER_ID">0

finally: "O"."ORDER_TOTAL">4000 AND "C"."CUSTOMER_ID"="O"."CUSTOMER_ID"
AND "O"."ORDER_ID"="I"."ORDER_ID" AND "O"."ORDER_DATE">=TIMESTAMP' 2007-
01-01 00:00:00' AND "O"."ORDER_DATE"<=TIMESTAMP' 2008-12-31 00:00:00'
AND TIMESTAMP' 2008-12-31 00:00:00'>=TIMESTAMP' 2007-01-01 00:00:00' AND
"O"."CUSTOMER_ID">0

FPD: transitive predicates are generated in query block SEL$F4A4219D(#0)
"O"."ORDER_TOTAL">4000 AND "C"."CUSTOMER_ID"="O"."CUSTOMER_ID" AND
"O"."ORDER_ID"="I"."ORDER_ID" AND "O"."ORDER_DATE">=TIMESTAMP' 2007-01-
01 00:00:00' AND "O"."ORDER_DATE"<=TIMESTAMP' 2008-12-31 00:00:00' AND
TIMESTAMP' 2008-12-31 00:00:00'>=TIMESTAMP' 2007-01-01 00:00:00' AND
"O"."CUSTOMER_ID">0
```

Part of 10053 – Trace file

Example 3 - Query Transformer – Final query

New query block - `SEL$F5BB74E1`

```
Final query after transformations:******* UNPARSED QUERY IS *******

SELECT "C"."CUSTOMER_ID" "CUSTOMER_ID","C"."CUST_FIRST_NAME"
"CUST_FIRST_NAME","C"."CUST_LAST_NAME"
"CUST_LAST_NAME","C"."CUST_EMAIL" "CUST_EMAIL","O"."ORDER_TOTAL"
"ORDER_TOTAL"
FROM "OE"."CUSTOMERS" "C","OE"."ORDERS" "O","OE"."ORDER_ITEMS" "I"
WHERE "O"."ORDER_TOTAL">4000
AND "C"."CUSTOMER_ID"="O"."CUSTOMER_ID"
AND "O"."ORDER_ID"="I"."ORDER_ID"
AND "O"."ORDER_DATE">=TIMESTAMP' 2007-01-01 00:00:00'
AND "O"."ORDER_DATE"<=TIMESTAMP' 2008-12-31 00:00:00'
AND TIMESTAMP' 2008-12-31 00:00:00'>=TIMESTAMP' 2007-01-01
00:00:00'
AND "O"."CUSTOMER_ID">0
```

Part of 10053 – Trace file

Example 3 - Query Transformer – Final query

Execution plan for `SEL$F5BB74E1`

```
-----------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                     |       |       |     9 (100)|          |
|*  1 |  FILTER                                  |                     |       |       |            |          |
|*  2 |   HASH JOIN                              |                     |   431 | 35342 |     9   (0)| 00:00:01 |
|*  3 |    HASH JOIN                             |                     |    68 |  5304 |     7   (0)| 00:00:01 |
|*  4 |     TABLE ACCESS BY INDEX ROWID BATCHED| ORDERS              |    68 |  1700 |     2   (0)| 00:00:01 |
|*  5 |      INDEX RANGE SCAN                    | ORD_ORDER_DATE_IX |    88 |       |     1   (0)| 00:00:01 |
|   6 |     TABLE ACCESS FULL                    | CUSTOMERS           |   319 | 16907 |     5   (0)| 00:00:01 |
|   7 |    INDEX FAST FULL SCAN                  | ITEM_ORDER_IX       |   665 |  2660 |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(TIMESTAMP' 2008-12-31 00:00:00'>=TIMESTAMP' 2007-01-01 00:00:00')
   2 - access("O"."ORDER_ID"="I"."ORDER_ID")
   3 - access("C"."CUSTOMER_ID"="O"."CUSTOMER_ID")
   4 - filter(("O"."ORDER_TOTAL">4000 AND "O"."CUSTOMER_ID">0))
   5 - access("O"."ORDER_DATE">=TIMESTAMP' 2007-01-01 00:00:00' AND "O"."ORDER_DATE"<=TIMESTAMP'
              2008-12-31 00:00:00')
```

Example 3 – What we really wanted

```
SELECT C.CUSTOMER_ID, C.CUST_FIRST_NAME, C.CUST_LAST_NAME,
C.CUST_EMAIL, O.ORDER_TOTAL
FROM OE.CUSTOMERS C, OE.ORDERS O
WHERE C.CUSTOMER_ID = O.CUSTOMER_ID
AND ORDER_TOTAL > 4000
AND O.ORDER_DATE BETWEEN TO_DATE('01/01/2007', 'DD/MM/YYYY') AND
TO_DATE('31/12/2008', 'DD/MM/YYYY');
```

Example 3 – What we really wanted

```
-------------------------------------------------------------------------------------------------------------------
|   Id  | Operation                                    | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT                             |                     |       |       |     7 (100)|          |
|  *  1 |  FILTER                                      |                     |       |       |            |          |
|  *  2 |   HASH JOIN                                  |                     |    67 |  4958 |     7   (0)| 00:00:01 |
|-    3 |    NESTED LOOPS                             |                     |    67 |  4958 |     7   (0)| 00:00:01 |
|-    4 |     NESTED LOOPS                            |                     |       |       |            |          |
|-    5 |      STATISTICS COLLECTOR                    |                     |       |       |            |          |
|  *  6 |       TABLE ACCESS BY INDEX ROWID BATCHED    | ORDERS              |    67 |  1407 |     2   (0)| 00:00:01 |
|  *  7 |        INDEX RANGE SCAN                      | ORD_ORDER_DATE_IX   |    88 |       |     1   (0)| 00:00:01 |
|- *  8 |      INDEX UNIQUE SCAN                       | CUSTOMERS_PK        |       |       |            |          |
|-    9 |     TABLE ACCESS BY INDEX ROWID              | CUSTOMERS           |     1 |    53 |     5   (0)| 00:00:01 |
|    10 |    TABLE ACCESS FULL                         | CUSTOMERS           |   319 | 16907 |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(TIMESTAMP' 2008-12-31 00:00:00'>=TIMESTAMP' 2007-01-01 00:00:00')
   2 - access("C"."CUSTOMER_ID"="O"."CUSTOMER_ID")
   6 - filter(("ORDER_TOTAL">4000 AND "O"."CUSTOMER_ID">0))
   7 - access("O"."ORDER_DATE">=TIMESTAMP' 2007-01-01 00:00:00' AND "O"."ORDER_DATE"<=TIMESTAMP'
              2008-12-31 00:00:00')
   8 - access("C"."CUSTOMER_ID"="O"."CUSTOMER_ID")
```

Avoid common mistakes

```
SELECT MAX (amount_sold)
FROM ZSALES;
```

```
-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                    |       |       |     3 (100)|          |
|   1 |  SORT AGGREGATE              |                    |     1 |     5 |            |          |
|   2 |   INDEX FULL SCAN (MIN/MAX)| ZSALES_AMOUNT_IDX |     1 |     5 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

Avoid common mistakes

```
SELECT MAX (amount_sold*2.3)
FROM ZSALES;
```

```
-------------------------------------------------------------------------------------
| Id  | Operation             | Name               | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                    |       |       | 33829 (100)|          |
|   1 |  SORT AGGREGATE       |                    |     1 |     5 |            |          |
|   2 |   INDEX FAST FULL SCAN| ZSALES_AMOUNT_IDX  |    58M|   280M| 33829   (1)| 00:00:02 |
-------------------------------------------------------------------------------------
```

Avoid common mistakes

```
CREATE INDEX AMOUNT_SOLD_FIDX ON ZSALES (AMOUNT_SOLD*2.3);
```

Avoid common mistakes

```
SELECT MAX (amount_sold*2.3)
FROM ZSALES;
```

```
-------------------------------------------------------------------------------------
| Id  | Operation                    | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                   |       |       |     3 (100)|          |
|   1 |  SORT AGGREGATE              |                   |     1 |    13 |            |          |
|   2 |   INDEX FULL SCAN (MIN/MAX)| AMOUNT_SOLD_FIDX  |     1 |    13 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------
```

Avoid common mistakes

```
SELECT DEPARTMENT_ID, AVG(SALARY)
FROM HR.EMPLOYEES
GROUP BY DEPARTMENT_ID
HAVING DEPARTMENT_ID > 20;
```

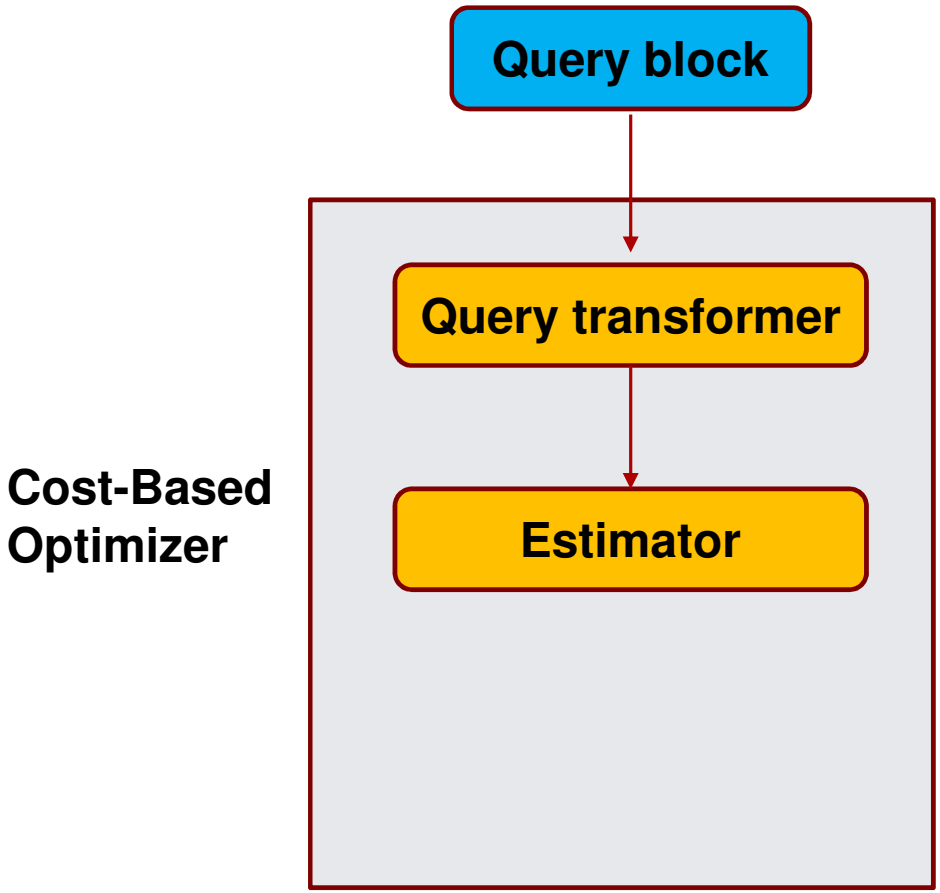
```
-------------------------------------------------------------------------------
| Id  | Operation            | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |           |       |       |     4 (100)|          |
|*  1 |  FILTER              |           |       |       |            |          |
|   2 |   HASH GROUP BY      |           |    11 |    77 |     4  (25)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| EMPLOYEES |   107 |   749 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Avoid common mistakes

```
SELECT DEPARTMENT_ID, AVG(SALARY)
FROM HR.EMPLOYEES
WHERE DEPARTMENT_ID > 20
GROUP BY DEPARTMENT_ID;
```

```
-------------------------------------------------------------------------------------
| Id  | Operation                     | Name               | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                    |       |       |     1 (100)|
|   1 |  SORT GROUP BY NOSORT         |                    |    11 |    77 |     0   (0)|
|   2 |   TABLE ACCESS BY INDEX ROWID| EMPLOYEES          |   103 |   721 |     0   (0)|
|*  3 |    INDEX RANGE SCAN           | EMP_DEPARTMENT_IX |     1 |       |     0   (0)|
-------------------------------------------------------------------------------------
```

Cost-Based Optimizer



Estimator

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

Return all car users in specified country

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE COUNTRY = 'USA';
```

Estimator

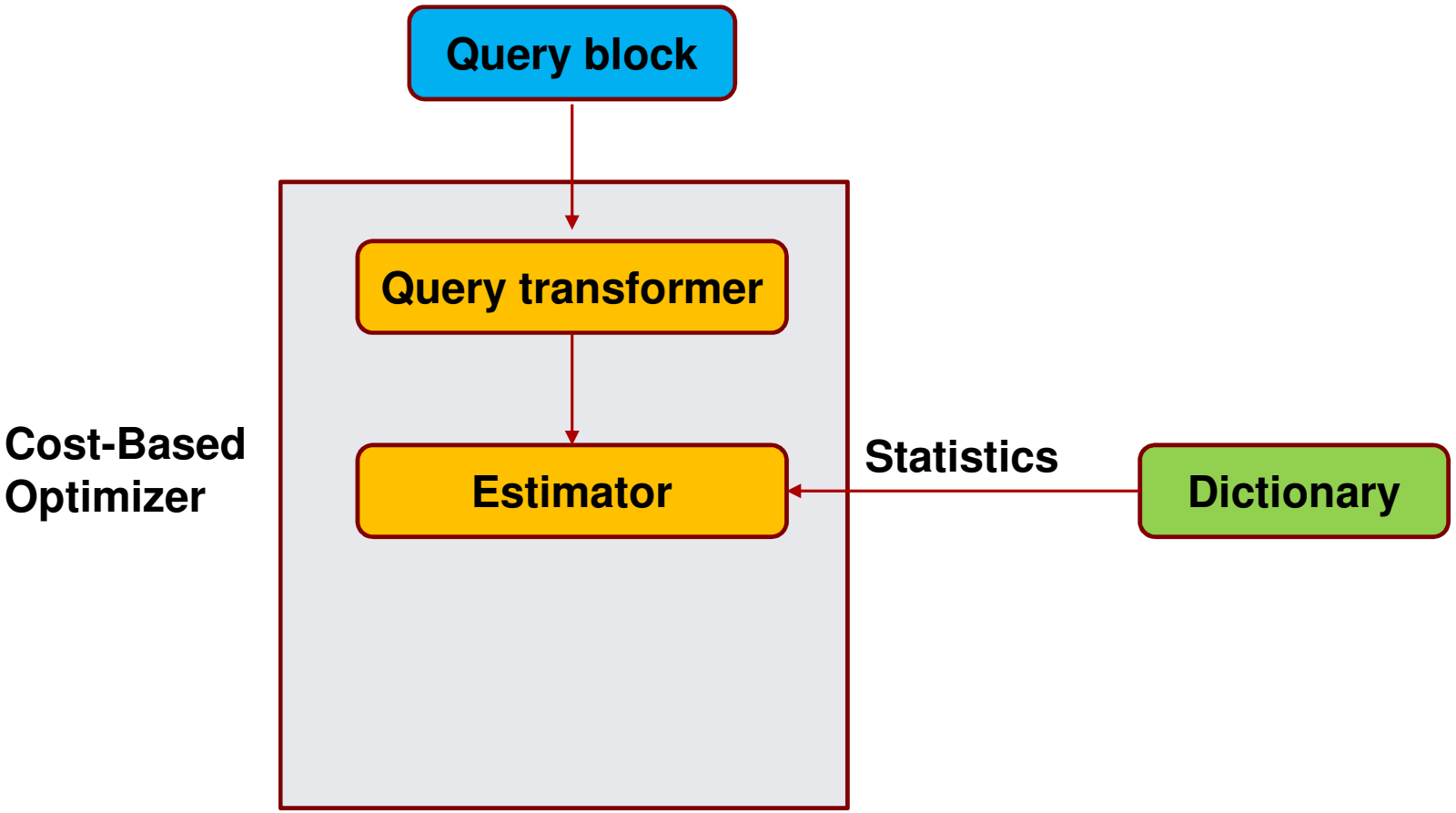
CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE COUNTRY = 'USA';
```

How many users and how many countries?

Cost-Based Optimizer



Estimator

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

There are 743 438 652 rows

There are 125 countries

Object statistics

Estimator

$$\text{Selectivity} = \frac{\text{Number of rows that satisfy condition}}{\text{Total number of rows}}$$

$$\text{Cardinality} = \text{Total number of rows} * \text{Selectivity}$$

Estimator

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE COUNTRY = 'USA';
```

There are 743 438 652 rows

There are 125 countries

$$\text{Selectivity} = \frac{1}{125}$$

Estimator

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE COUNTRY = 'USA';
```

There are 743 438 652 rows

There are 125 countries

$$Selectivity = \frac{1}{125}$$

$$Cardinality = \frac{1}{125} * 743\,436\,652 = 5\ \ 947\ \ 494$$

Estimator

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE COUNTRY = 'USA';
```

There are 743 438 652 rows

There are 125 countries

$$Selectivity = \frac{1}{125}$$

$$Cardinality = \frac{1}{125} * 743\,436\,652 = 5\ \ 947\ \ 494$$

$$Actual\ Cardinality = 216\ \ 327\ \ 872$$

Estimator

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE COUNTRY = 'San Marino';
```

There are 743 438 652 rows

There are 125 countries

$$\text{Selectivity} = \frac{1}{125}$$

$$\text{Cardinality} = \frac{1}{125} * 743\,436\,652 = 5\ \ 947\ \ 494$$

Estimator

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE COUNTRY = 'San Marino';
```

There are 743 438 652 rows

There are 125 countries

$$Selectivity = \frac{1}{125}$$

$$Cardinality = \frac{1}{125} * 743\,436\,652 = 5\ \ 947\ \ 494$$

$$Actual\ Cardinality = 73\ \ 872$$

Estimator - Histograms

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
BEGIN DBMS_STATS.GATHER_TABLE_STATS (ownname => 'ZORAN',
tabname => 'CARUSERS', method_opt => 'FOR COLUMNS COUNTRY');
```

Estimator – Multi column predicate

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE MAKE = 'BMW'
AND MODEL = 'OCTAVIA';
```

Estimator – Multi column predicate

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE MAKE = 'BMW'
AND MODEL = 'OCTAVIA';
```

There are 19 car makers

There are 72 car models

Estimator – Multi column predicate

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE MAKE = 'BMW'
AND MODEL = 'OCTAVIA';
```

There are 19 car makers

There are 72 car models

$$\text{Selectivity} = \frac{1}{19} * \frac{1}{72}$$

Estimator – Multi column predicate

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE MAKE = 'BMW'
AND MODEL = 'OCTAVIA';
```

There are 19 car makers

There are 72 car models

$$Selectivity = \frac{1}{19} * \frac{1}{72}$$

$$Cardinality = \frac{1}{19} * \frac{1}{72} * 743\,436\,652 = 543\ \ 448$$

Estimator – Multi column predicate

CARUSERS Table

USERID	NAME	MAKE	MODEL	COUNTRY

```
SELECT USERID, NAME, MAKE, MODEL
FROM CARUSERS
WHERE MAKE = 'BMW'
AND MODEL = 'OCTAVIA';
```

There are 19 car makers

There are 72 car models

$$Selectivity = \frac{1}{19} * \frac{1}{72}$$

$$Cardinality = \frac{1}{19} * \frac{1}{72} * 743\,436\,652 = 543\ \ 448$$

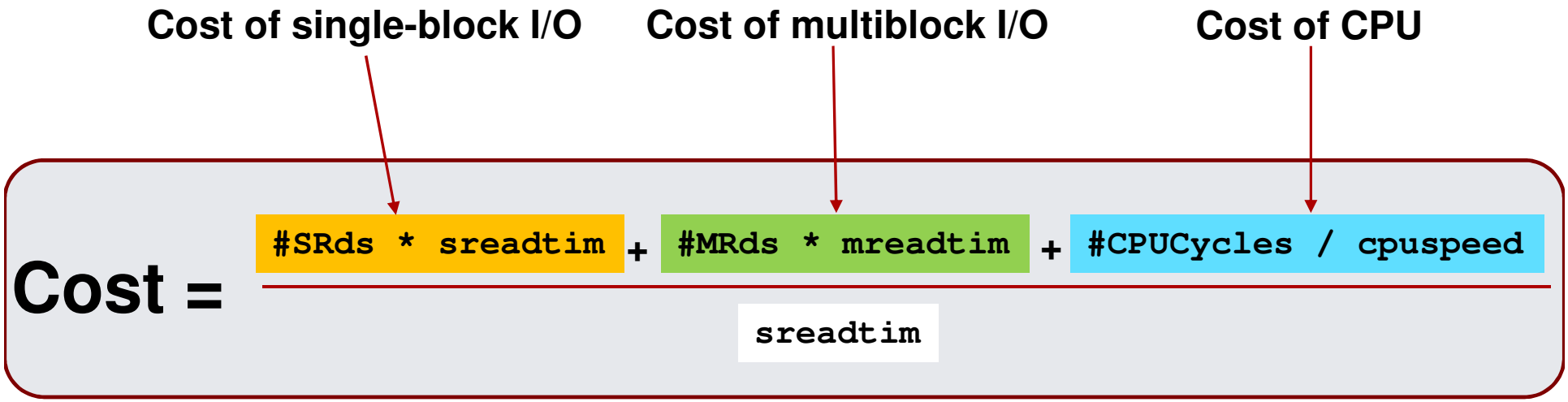
Actual Cardinality = 0

Estimator – Multi column predicate

```
SELECT SYS.DBMS_STATS.CREATE_EXTENDED_STATS
('ZORAN','CARUSERS','(make,model)') FROM DUAL;

EXEC DBMS_STATS.GATHER_TABLE_STATS(`ZORAN', `CARUSERS`);
```

Calculating Cost



`#SRds`: *Number of single-block reads*
`#MRds`: *Number of multiblock reads*
`#CPUCycles`: *Number of CPU Cycles*

`sreadtim`: *Single-block read time*
`mreadtim`: *Multiblock read time*
`cpuspeed`: *Millions instructions per second*

Single Table Access Paths

- Table Acces
  - Full Table Scan
  - Sample Table Scan
  - RowId Scan
- Index Access
  - Index Unique Scan
  - Index Range Scan
  - Index Full Scan
  - Index Skip Scan
  - Index Fast Full Scan

Joining methods

- Hash Join
- Nested-Loops Join
- Sort-Merge Join

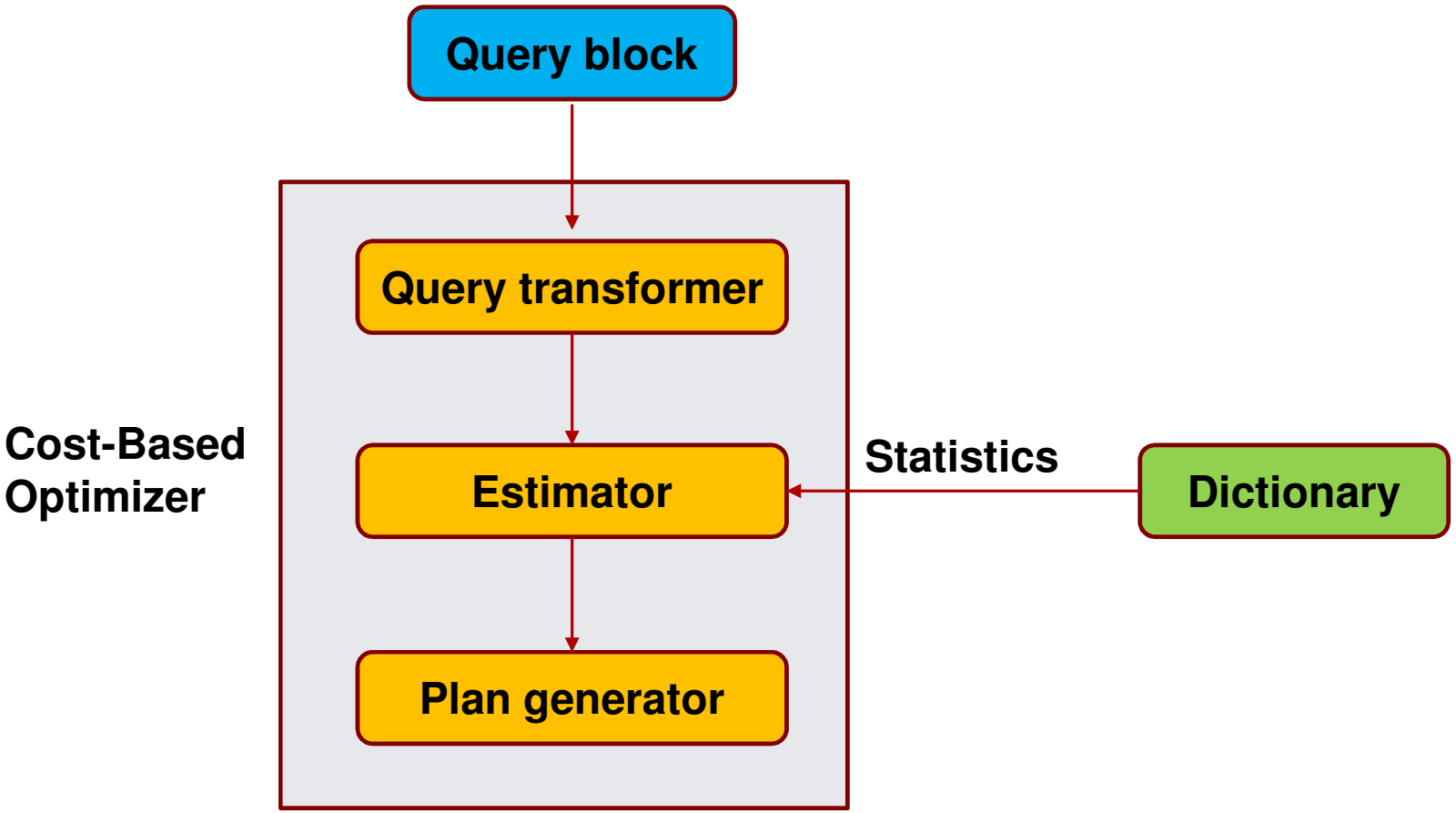
Estimator – Finding Best Access Path

```
Access path analysis for DEPARTMENTS
***************************************
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for DEPARTMENTS[D]
  Table: DEPARTMENTS  Alias: D
    Card: Original: 27.000000  Rounded: 1  Computed: 1.000000
Non Adjusted: 1.000000
  Access Path: TableScan
    Cost:  3.001033  Resp: 3.001033  Degree: 0
      Cost_io: 3.000000  Cost_cpu: 41027
      Resp_io: 3.000000  Resp_cpu: 41027
 ****** Costing Index DEPT_ID_PK
  Access Path: index (UniqueScan)
    Index: DEPT_ID_PK
    resc_io: 0.000000  resc_cpu: 1220
    ix_sel: 0.037037  ix_sel_with_filters: 0.037037
    Cost: 0.000031  Resp: 0.000031  Degree: 1
  Best:: AccessPath: IndexUnique
  Index: DEPT_ID_PK
         Cost: 0.000031  Degree: 1  Resp: 0.000031  Card:
1.000000  Bytes: 0.000000
```

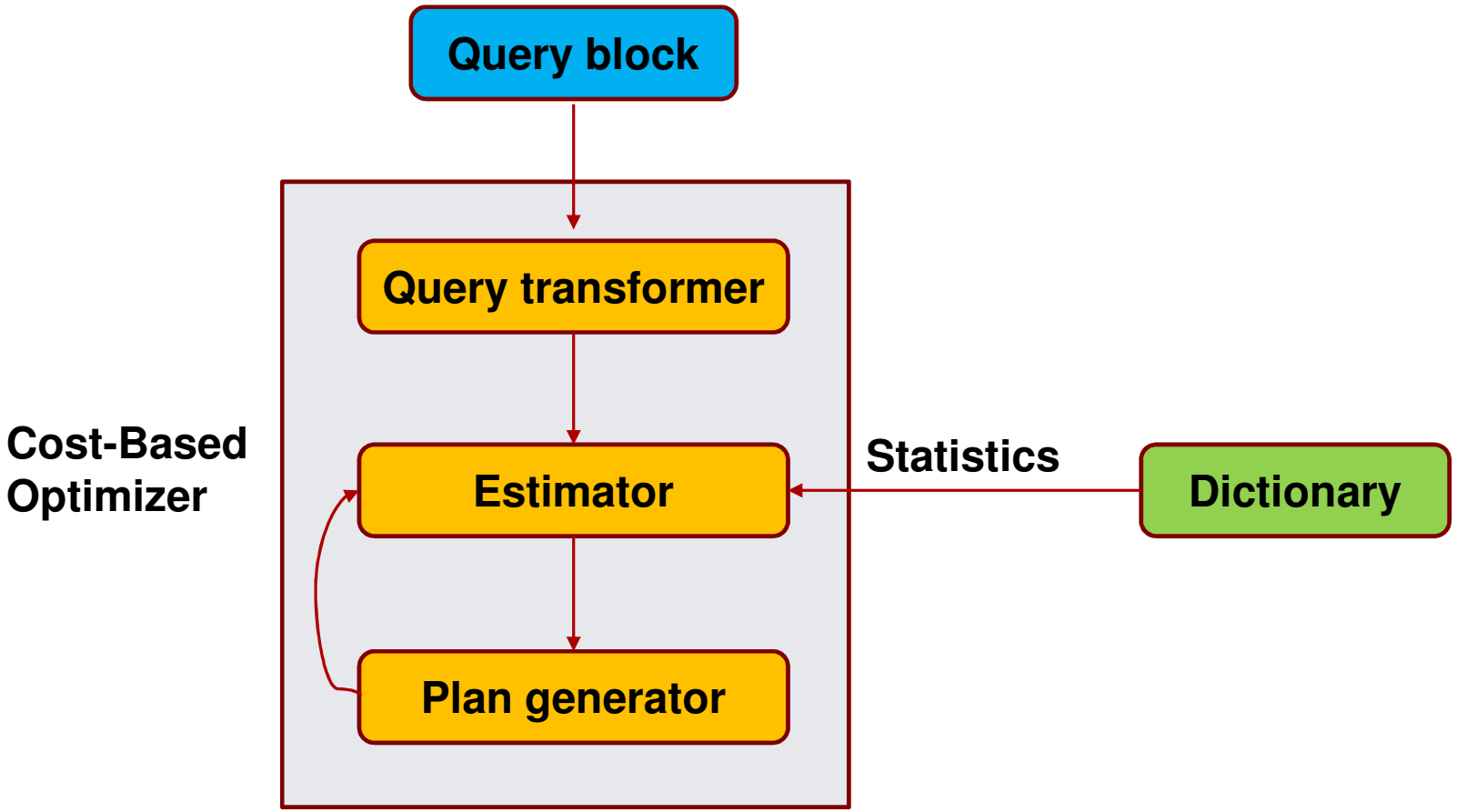
Estimator – Finding Best Access Path

```
NL Join
  Outer table: Card: 1.000000  Cost: 0.000031  Resp: 0.000031  Degree: 1
Bytes:
Access path analysis for EMPLOYEES
  Inner table: EMPLOYEES  Alias: E
  Access Path: TableScan
    NL Join:  Cost: 3.002063
 ****** Costing Index EMP_DEPARTMENT_IX
  Access Path: index (AllEqJoin)
    Index: EMP_DEPARTMENT_IX
    NL Join : Cost: 0.000037
  Best NL cost: 0.000037
SM Join
  SM cost: 2.000316
HA Join
  HA cost: 1.015263
Best:: JoinMethod: NestedLoop
       Cost: 0.000037
**********************
Best so far:  Table#: 0  cost: 0.000031
              Table#: 1  cost: 0.000037
```

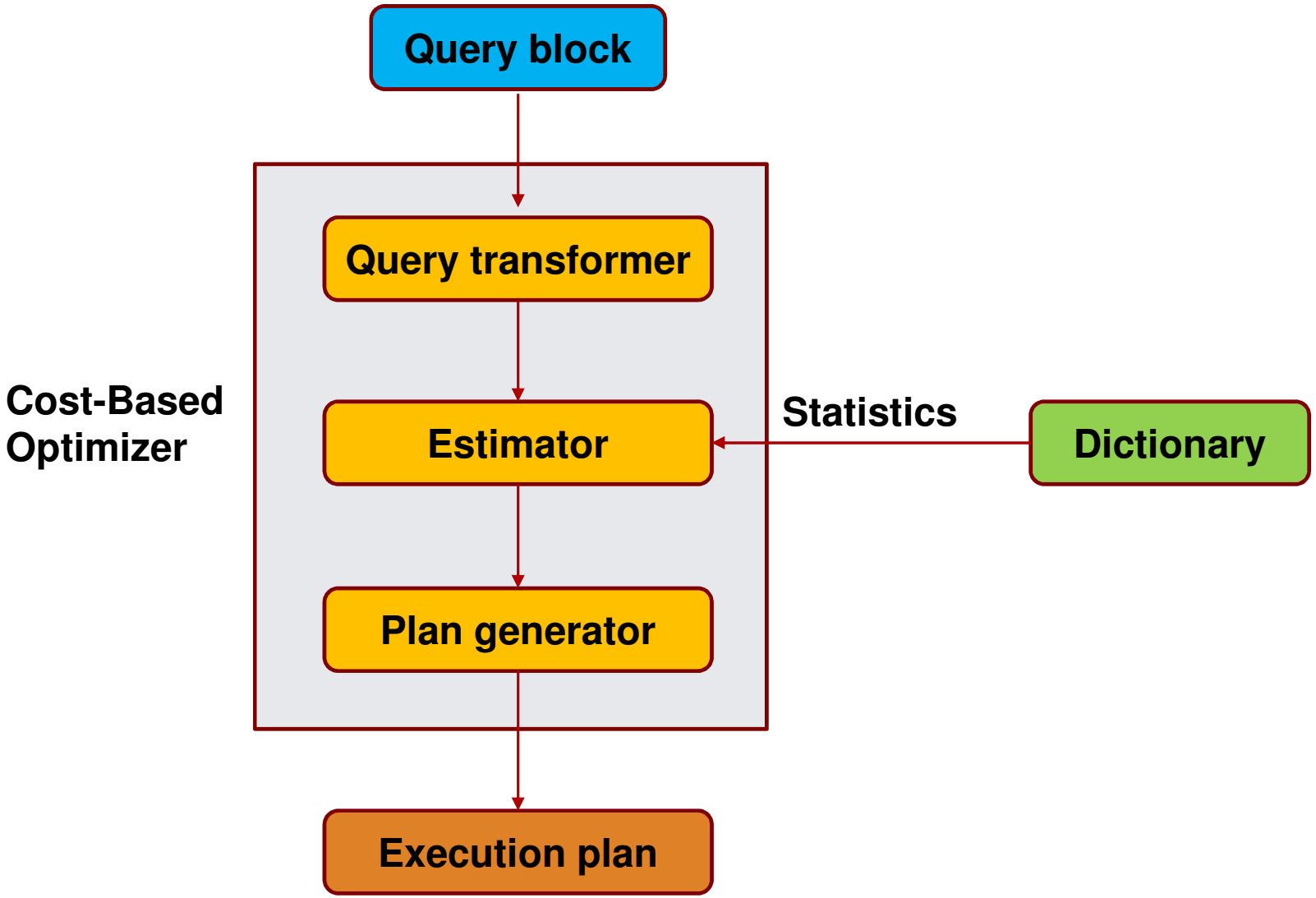
Cost-Based Optimizer



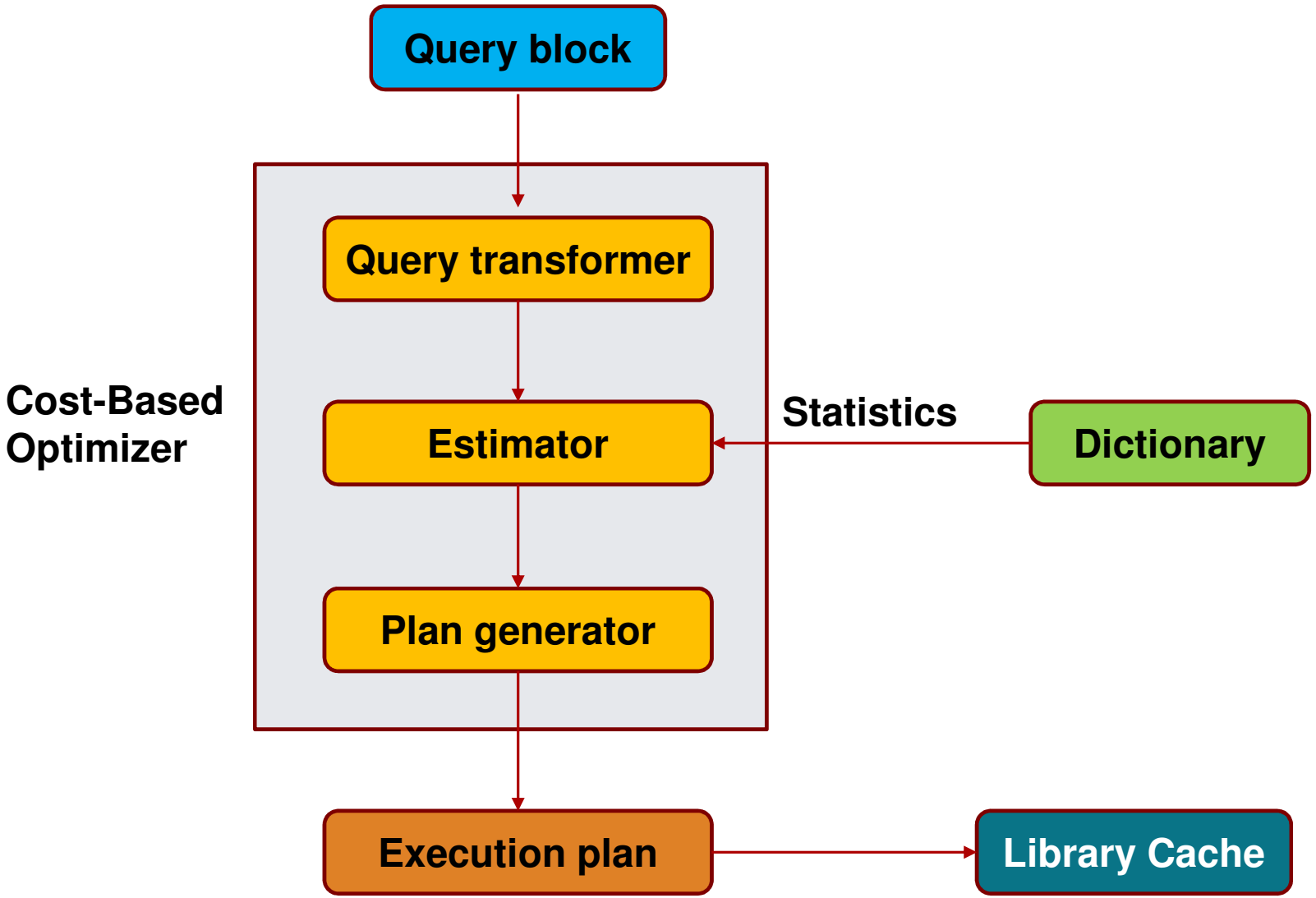
Cost-Based Optimizer



Cost-Based Optimizer



Cost-Based Optimizer



Bloom filters

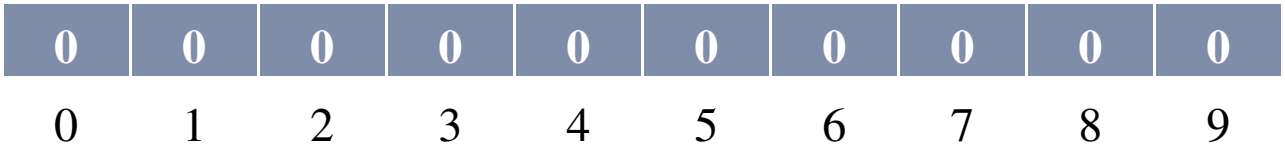
- Created by Burton Howard Bloom in 1970.
- Used to test whether element is member of a set.
- Time needed to test membership is independent on number of elements in set
- False positives are possible
- False negatives are NOT possible
- First introduced in Oracle 10g Release 2

Bloom filters - Create

SET = {}

FNV:

MURMUR:

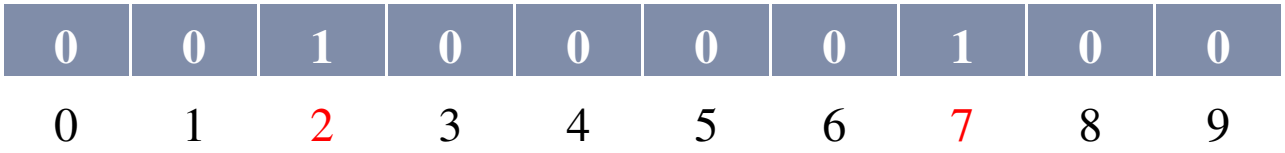


Bloom filters - Create

SET = {fval}

FNV: 7

MURMUR: 2



Bloom filters - Create

SET = {fval, 2ndval}

FNV: 8

MURMUR: 9

0	0	1	0	0	0	0	1	1	1
0	1	2	3	4	5	6	7	8	9

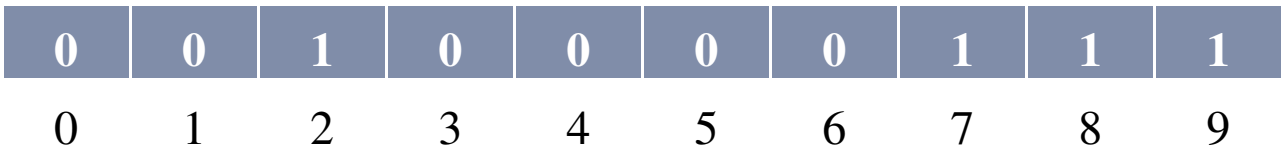
Bloom filters - Test

Test Value = {2ndval}

FNV: 8

MURMUR: 9

POSSIBLE MATCH!



SET = {fval, 2ndval}

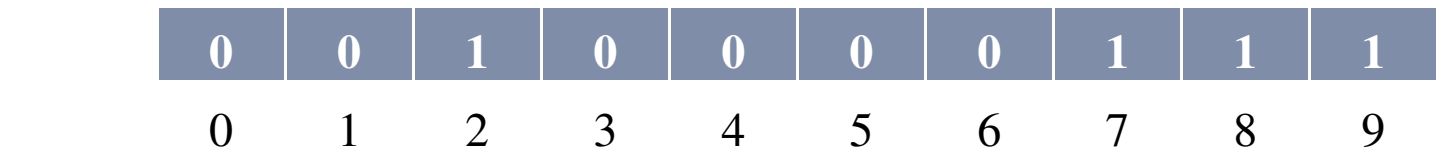
Bloom filters - Test

Test Value = {3rdval}

FNV: 0

MURMUR: 9

NOMATCH!



SET = {fval, 2ndval}

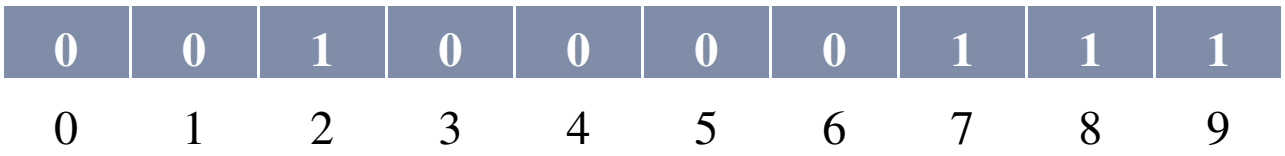
Bloom filters - Test

Test Value = {vaw}

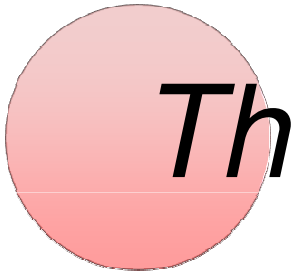
FNV: 9

MURMUR: 7

FALSE POSITIVE!



SET = {fval, 2ndval}



Thank you!