performing databases

Roam With The Beast
-
Let the Oracle Cost-Based Optimizer Run!

Martin Klier 

Performing Databases GmbH
Mitterteich / Germany

Speaker

- Martin Klier
- Solution Architect and Database Expert
- My focus:
  - Performance + Tuning
  - Highly available systems
  - Cluster and Replication
- Linux since 1997
- Oracle Database since 2003

Oracle ACE Director

SYMPOSIUM42
Proud Member of symposium42

… it‘s all about Community!

_SYMPOSIUM

Stay tuned!

Performing Databases

- Three Experts for Database technology
  - Concepts and Project Competence
  - Architecture- and System planning
  - Licensing
  - Implementation and Troubleshooting

- Get in touch
  - Performing Databases GmbH
    Wiesauer Strasse 27
    95666 Mitterteich // Germany
  - http://www.performing-databases.com
  - Twitter: @PerformingDB

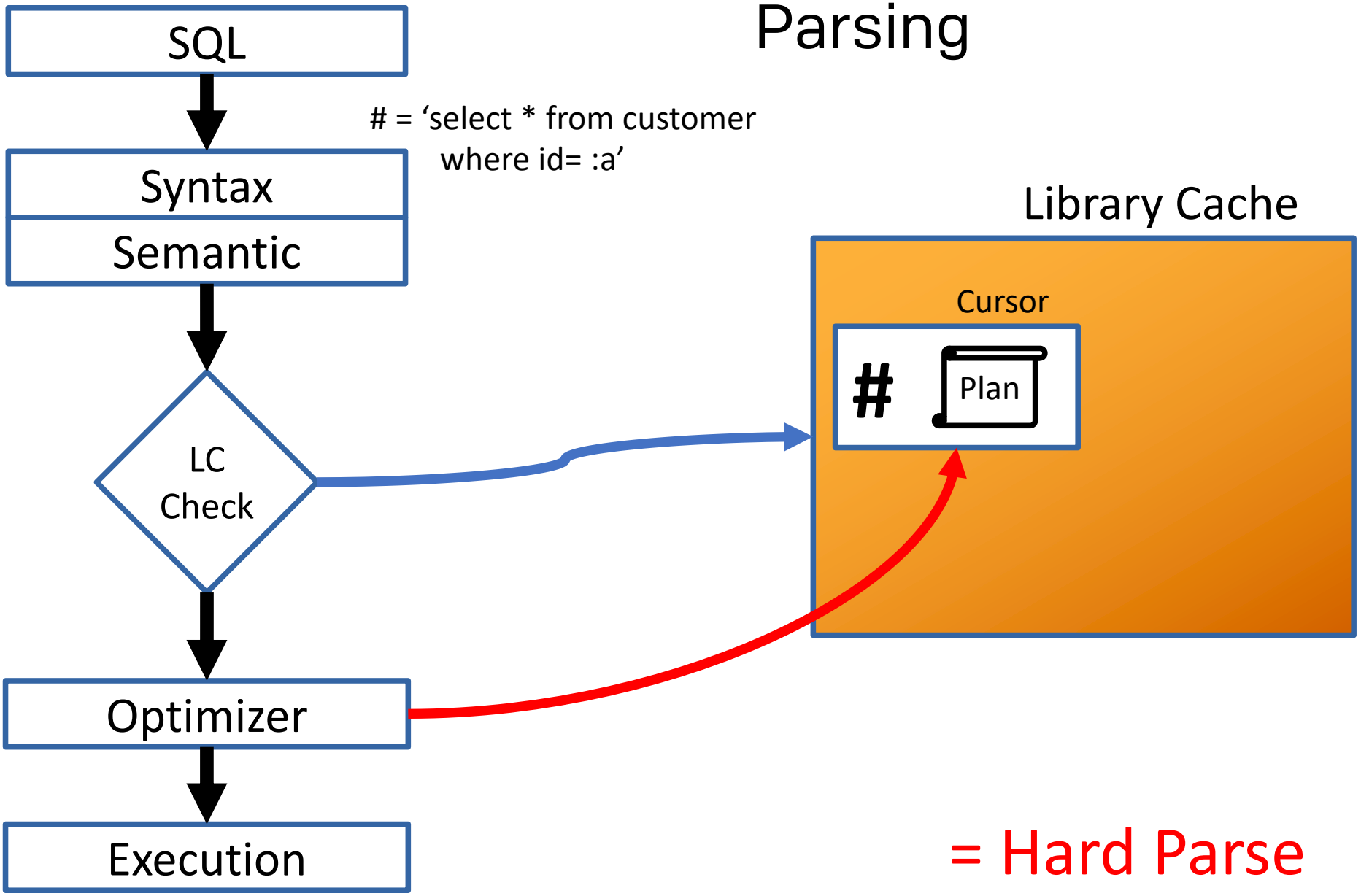
A strong impression that something must be done, is the origin of many bad measures.

~ Daniel Webster (1782-1852)
Public Performance Specialist

Lawyer, U.S. Senator, 14th & 19th Secretary of State

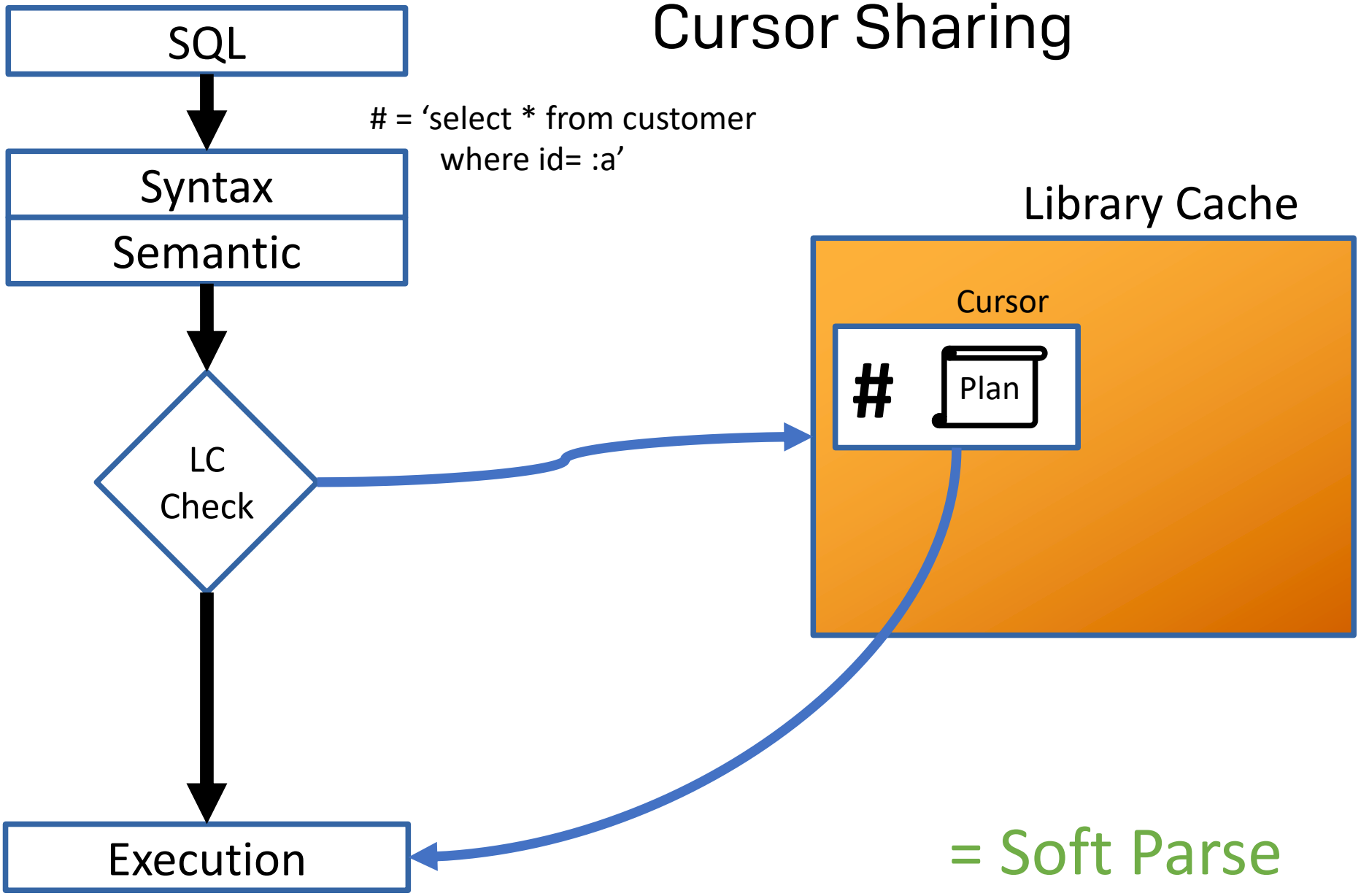
Theory

Parsing



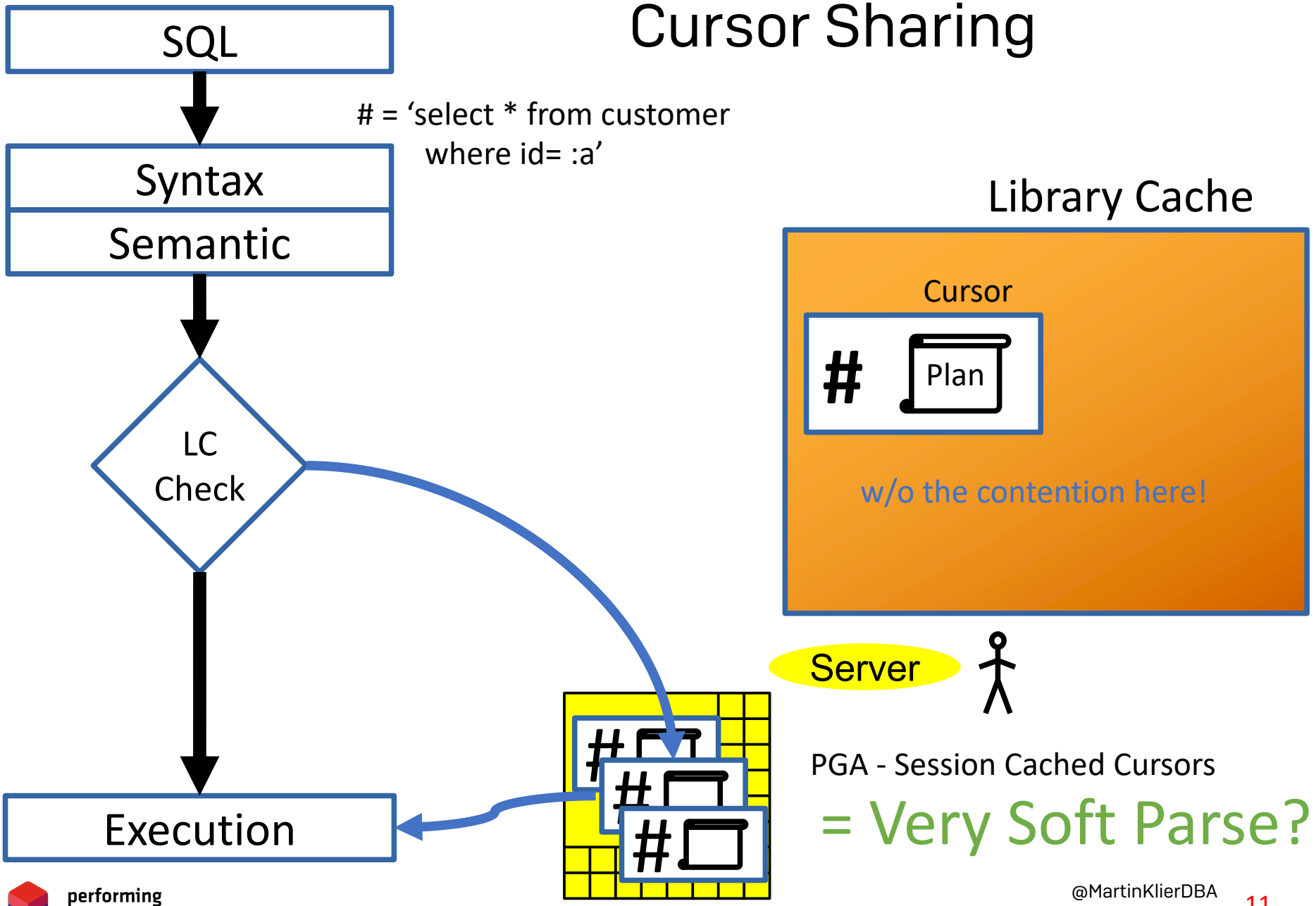
= Hard Parse

Cursor Sharing

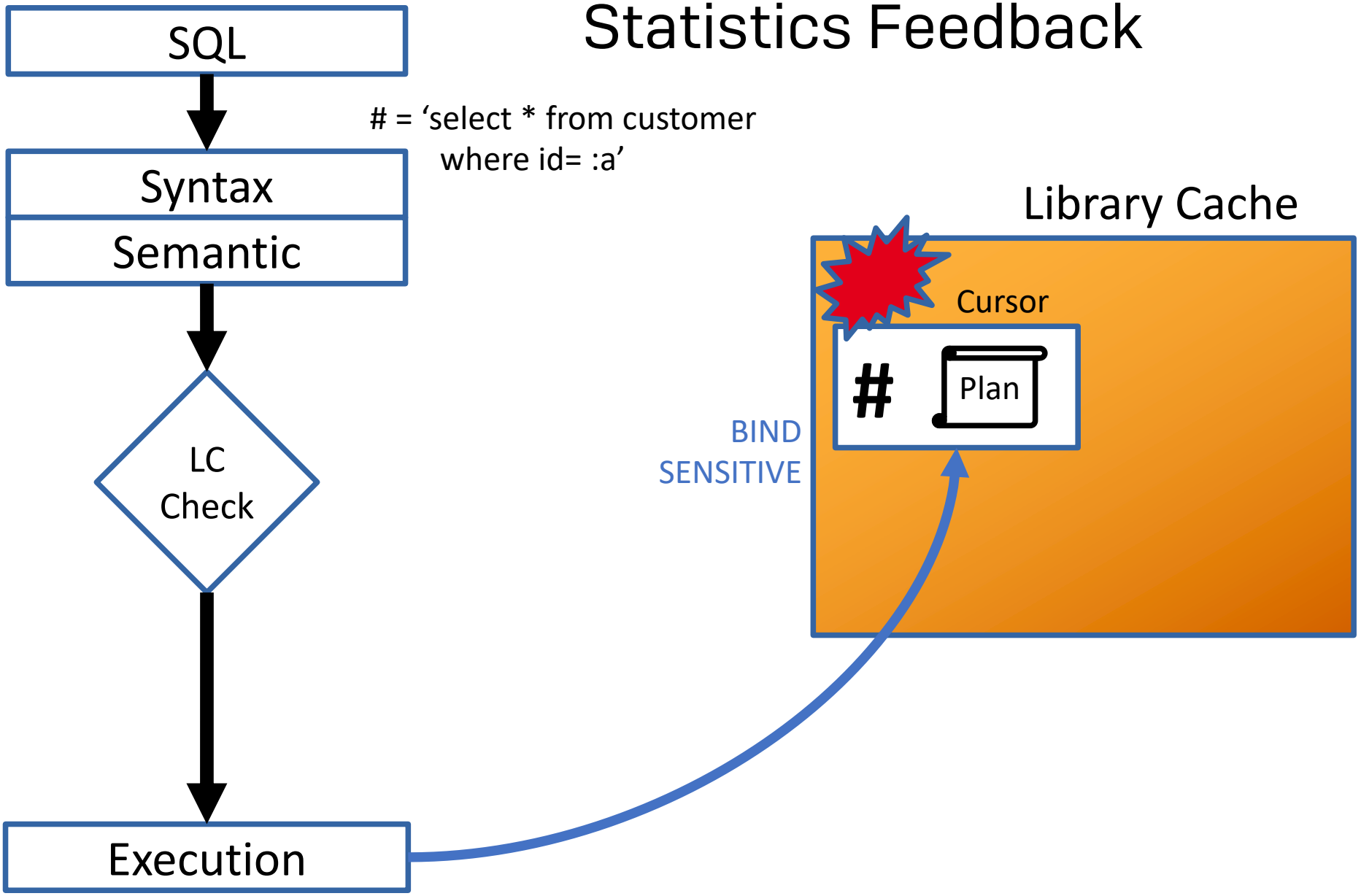


= Soft Parse

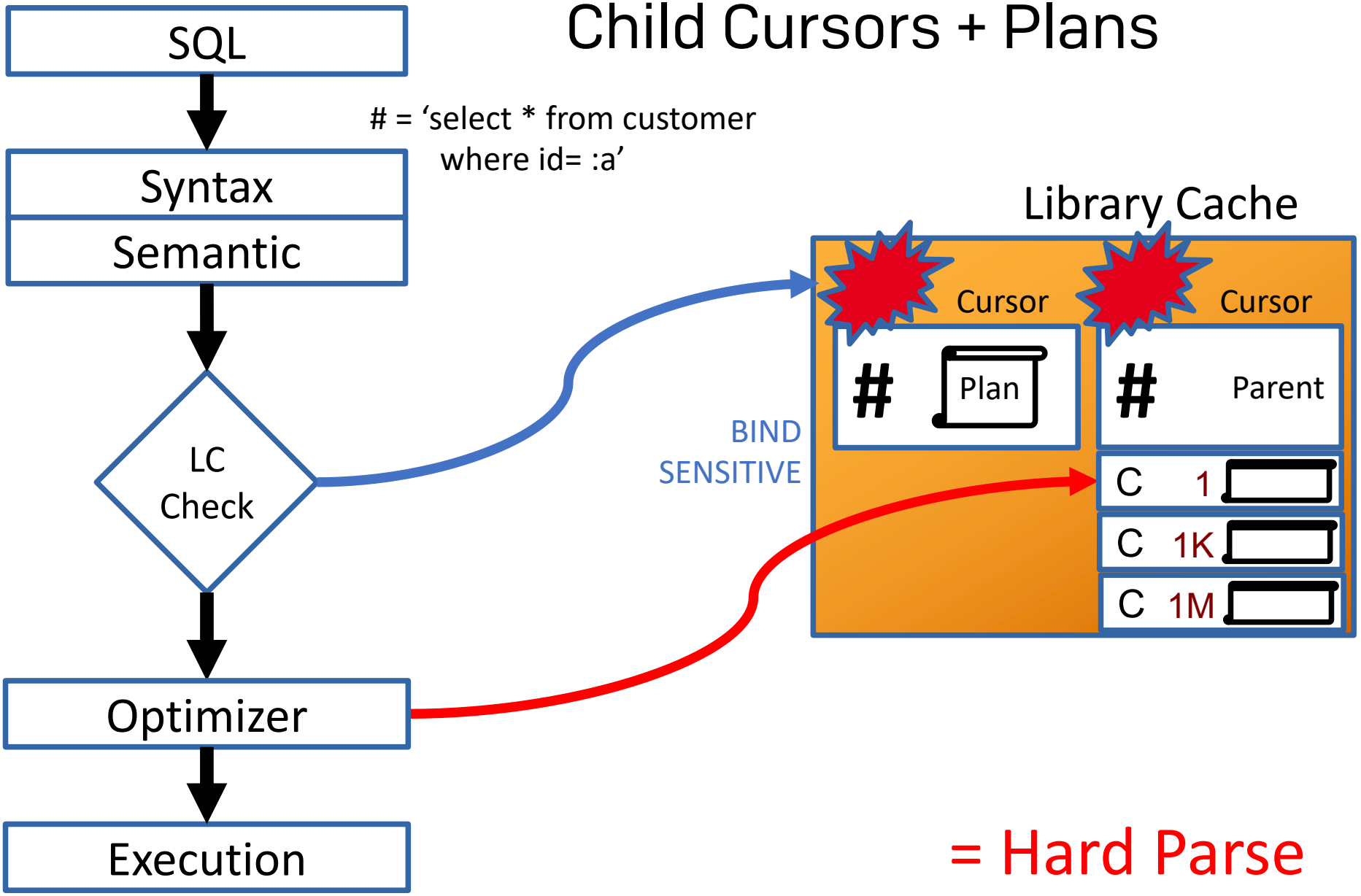
Cursor Sharing



Statistics Feedback

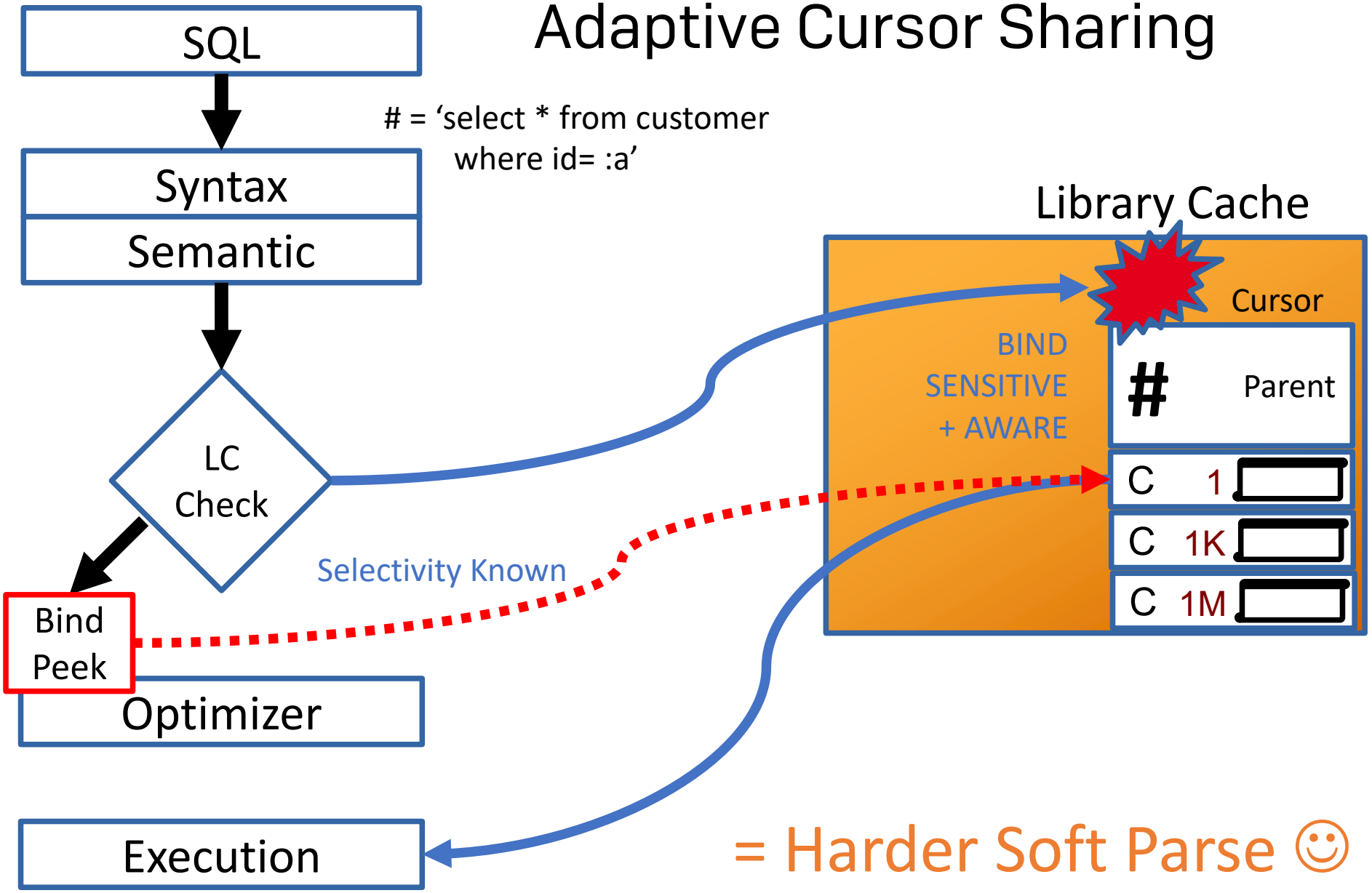


Child Cursors + Plans



= Hard Parse

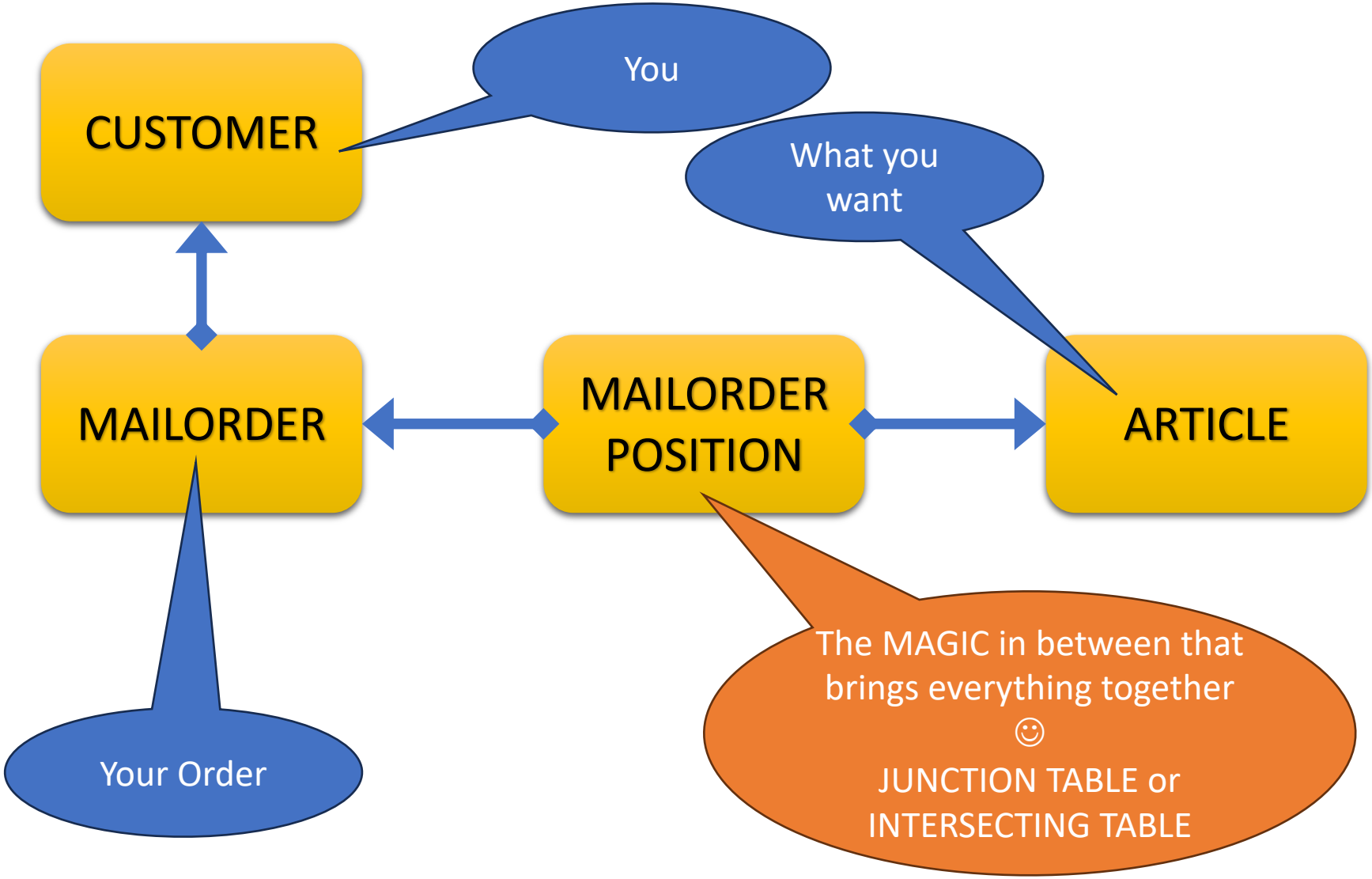
Adaptive Cursor Sharing



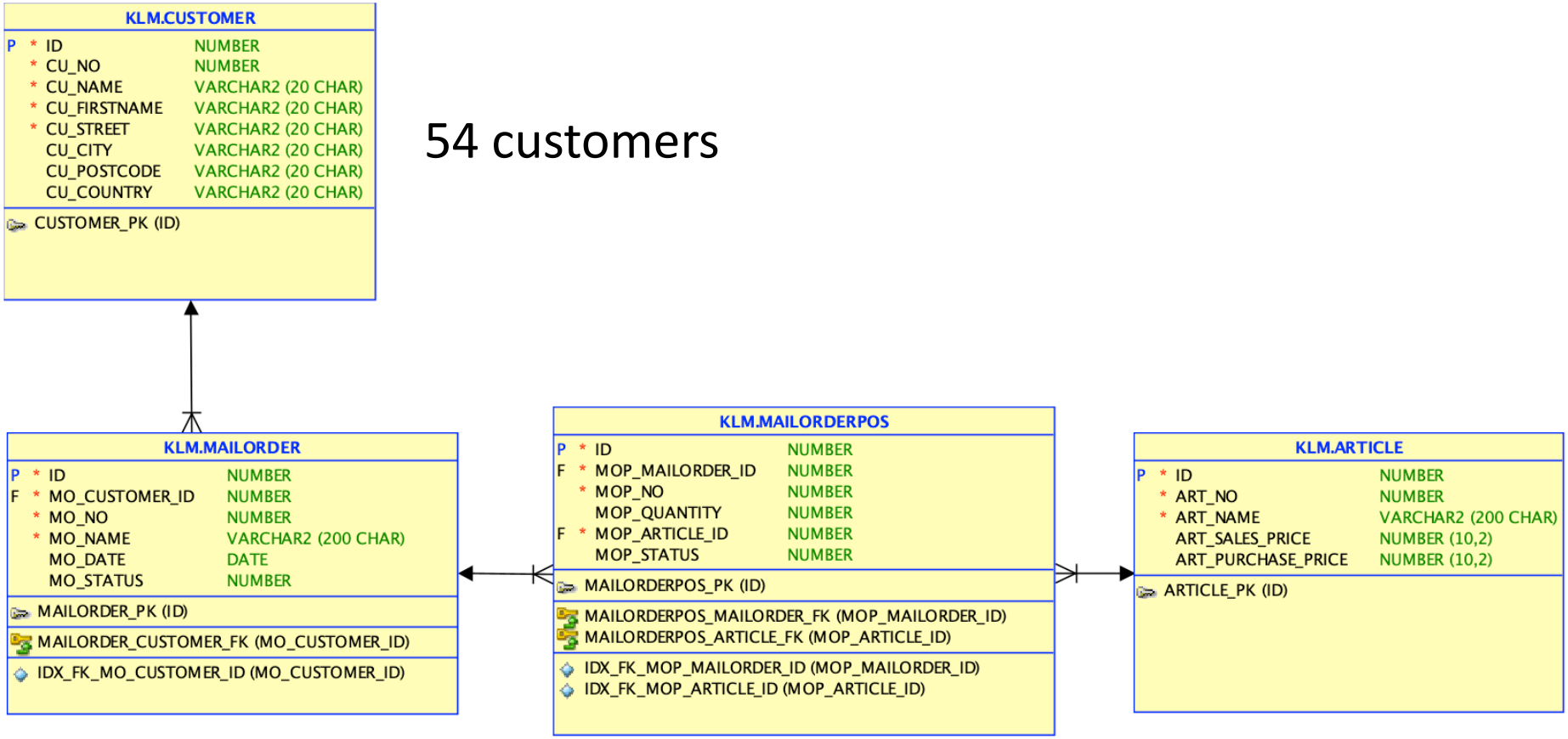
= Harder Soft Parse ☺

Lab Data Model

Lab Data Model



Lab Data Model



54 customers

1.000 mailorders

1.100.000 mailorderpos’s

101 articles

Simple Cases
Advanced Problems

TABLE ACCESS FULL

```
select * from klm.article;
```

50 Zeilen abgerufen in 0,049 Sekunden

	ID	ART_NO	ART_NAME
1	169	3001	Digital Camera
2	170	3002	Wireless Headphones
3	171	3003	Smartwatch
4	172	3004	Laptop Backpack
5	173	3005	Bluetooth Speaker
6	174	3006	Portable Charger
7	175	3007	Gaming Mouse
8	176	3008	Fitness Tracker

```
1  select * from klm.article;
2
```

Abfrageergebnis | Autotrace

SQL HotSpot | 2,546 Sekunden

OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
PX COORDINATOR		
PX SEND	SYS.:TQ10000	QC (RANDOM)
PX BLOCK		ITERATOR
TABLE ACCESS	ARTICLE	FULL
Access Predicates		
AND		
:Z>=:Z		
:Z<=:Z		

```
1  select /*+noparallel*/ * from klm.article;
2
```

Abfrageergebnis | Autotrace

SQL HotSpot | 2,329 Sekunden

OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	ARTICLE	FULL

We stick with NOPARALLEL for a while

TABLE ACCESS FULL vs. INDEX+TABLE

```
6  select /*+noparallel*/ *
7  from mailorderpos
8  where MOP_NO=8306
9  ;
```

One out of 10k+

Autotrace

SQL HotSpot | 2,426 Sekunden

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS	LAST_ELAPSED_TIME
SELECT STATEMENT				375	1382	3030
TABLE ACCESS	MAILORDERPOS	FULL	1	375	1382	3030
Filter Predicates						
MOP_NO=8306						

No Index on MOP_NO

```
1  select /*+noparallel*/ *
2  from mailorderpos
3  where ID=368602
4  ;
```

Autotrace

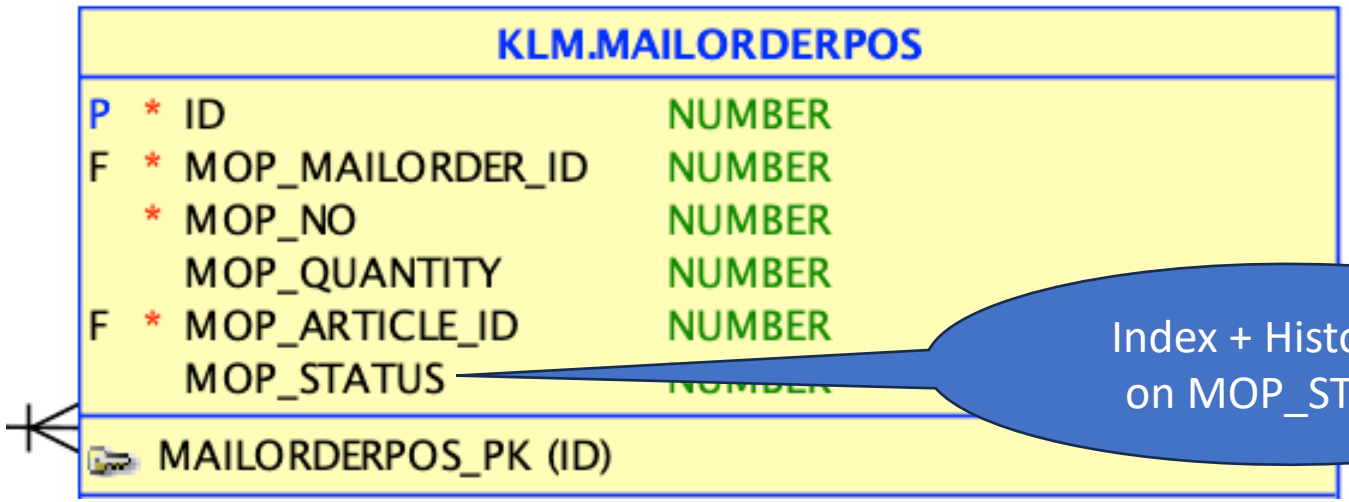
SQL HotSpot | 2,423 Sekunden

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS	LAST_ELAPSED_TIME
SELECT STATEMENT				2	3	26
TABLE ACCESS	MAILORDERPOS	BY INDEX ROWID	1	2	3	26
INDEX	MAILORDERPOS_PK	UNIQUE SCAN	1	1	2	16
Access Predicates						
ID=368602						

PK Index on (MOP) ID

Why NOT to force the optimizer

Data Distribution in MAILORDERPOS



Nearly All "DONE"

Some "OPEN"

Very Few "IN WORK"

MOP_STATUS	X
99	1093272
1	6727
50	1

Data Distribution in MAILORDERPOS

```
select * from dba_tab_col_statistics
where owner='KLM'
and table_name='MAILORDER'
and column_name='MO_STATUS'
;
```

DBA_TAB_COL_STATISTICS

Abfrageergebnis 1

Alle Zeilen abgerufen:1 in 0,362 Sekunden

OWNER	TABLE_NAME	COLUMN_NAME	NUM_DISTINCT	LOW_VALUE	HIGH_VALUE	DENSITY	NUM_NULLS	NUM_BUCKETS	LAST_ANALYZED	SAMPLE_SIZE	GLOBAL_STATS	USER_STATS	NOTES	AVG_COL_LEN	HISTOGRAM
KLM	MAILORDER	MO_STATUS	3	C102	C164	0,0005	0	3	15.03.2024 17:15:26	1000	YES	NO	HYPERLOGLOG	3	FREQUENCY

FREQUENCY based histogram on MOP_STATUS

DBA_TAB_HISTOGRAMS

```
select * from dba_tab_histograms
where owner='KLM'
    and table_name='MAILORDER'
    and column_name='MO_STATUS'
order by column_name
;
```

Three Buckets show SKEWED value distribution

Abfrageergebnis 1

Alle Zeilen abgerufen:3 in 0,042 Sekunden

OWNER	TABLE_NAME	COLUMN_NAME	ENDPOINT_NUMBER	ENDPOINT_VALUE	ENDPOINT_ACTUAL_VALUE	ENDPOINT_ACTU
KLM	MAILORDER	MO_STATUS	6	1	1	C102
KLM	MAILORDER	MO_STATUS	7	50	50	C133
KLM	MAILORDER	MO_STATUS	1000	99	99	C164

Conflict by Cursor Sharing

Query for the Mailorder Position
IN WORK

```
EXECUTE :a := 50
select *
      from MAILORDERPOS
      where MOP_STATUS=:a;
```

1 row matches

INDEX RANGE SCAN
IDX_MOP_STATUS
(non-unique column)
+
TABLE ACCESS BY INDEX ROWID BATCHED
MAILORDERPOS
(we want to see all columns)

VS.

Query for the Mailorder Position
DONE

```
EXECUTE :a := 99
select *
      from MAILORDERPOS
      where MOP_STATUS=:a;
```

1.000.000 rows match

TABLE ACCESS FULL
MAILORDERPOS
(we want to see all columns)

Adaptive Cursor Sharing

After a few executions of both, cursor becomes a PARENT CURSOR

IS_BIND_AWARE = Y
IS_BIND_SENSITIVE=Y

GV$SQL

SQL_ID	CHILD_NUMBER	PLAN_HASH_VALUE	OPTIMIZER_COST	EXECUTIONS	IS_BIND_SENSITIVE	IS_BIND_AWARE	INST_ID	SQL_TEXT
d95h4wsmrywsb	1	2142752586	1307	6	Y	Y	2	select * from mailorder
d95h4wsmrywsb	2	4102084357	4	5	Y	Y	2	select * from mailorder
d95h4wsmrywsb	3	4102084357	4	4	Y	N	2	select * from mailorder

CHILD 1 = TABLE ACCESS FULL
CHILD 2 = INDEX + TABLE

Adaptive Cursor Sharing

DBMS_XPLAN.DISPLAY_CURSOR('<SQL_ID>',null,'COST,IOSTATS,LAST,ADVANCED,ADAPTIVE')

```
SQL_ID  d95h4wsmrywsb, child number 1
-------------------------------------
select * from mailorderpos where mop_status=:a

Plan hash value: 2142752586

-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name         | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |              |      1 |        |       |  1307 (100)|          |   1093K|00:00:00.12 |    6963 |
|*  1 |  TABLE ACCESS FULL | MAILORDERPOS |      1 |   1093K|    27M|  1307   (1)| 00:00:01 |   1093K|00:00:00.12 |    6963 |
-----------------------------------------------------------------------------------------------------------------------------------
```

CHILD 1

TABLE ACCESS FULL

6963 Buffer Gets

```
SQL_ID  d95h4wsmrywsb, child number 2
-------------------------------------
select * from mailorderpos where mop_status=:a

Plan hash value: 4102084357

-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name          | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |               |      1 |        |       |     4 (100)|          |      1 |00:00:00.01 |       4 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| MAILORDERPOS  |      1 |      1 |    26 |     4   (0)| 00:00:01 |      1 |00:00:00.01 |       4 |
|*  2 |   INDEX RANGE SCAN                  | IDX_MOP_STATUS|      1 |      1 |       |     3   (0)| 00:00:01 |      1 |00:00:00.01 |       3 |
-----------------------------------------------------------------------------------------------------------------------------------
```

CHILD 2

INDEX RANGE SCAN + TABLE ACC.

4 Buffer Gets

My Bug or Your Feature?

Somebody meant to help …

(Got rid of some TABLE ACCESS FULL by forcing the CBO into our STATUS index)

```
select /*+ INDEX(MAILORDERPOS IDX_MOP_STATUS) */ *
      from MAILORDERPOS
      where MOP_STATUS=:a;
```

But if :a = 99
-> 1.000.000 rows match

Hint forces the CBO into:

INDEX RANGE SCAN
IDX_MOP_STATUS
+
TABLE ACCESS BY INDEX ROWID BATCHED
MAILORDERPOS

=> ???

Without tampering:

TABLE ACCESS FULL
MAILORDERPOS

=> 6963 Buffer Gets

Somebody meant to help …

(Got rid of some TABLE ACCESS FULL by forcing the CBO into our STATUS index)

```
SQL_ID  06yfmbwymd86p, child number 0
-------------------------------------
select /*+ INDEX(MAILORDERPOS IDX_MOP_STATUS) */ * from mailorderpos
where mop_status=:a

Plan hash value: 4102084357

-----------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name           | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                |      1 |        |       | 10635 (100)|          |   1093K|00:00:00.35 |   11380 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| MAILORDERPOS   |      1 |   1093K|    27M| 10635   (1)| 00:00:01 |   1093K|00:00:00.35 |   11380 |
|*  2 |   INDEX RANGE SCAN                  | IDX_MOP_STATUS |      1 |   1093K|       |  2279   (1)| 00:00:01 |   1093K|00:00:00.13 |    4450 |
-----------------------------------------------------------------------------------------------------------------------------------------------
```

DBMS_XPLAN.DISPLAY_CURSOR()

INDEX RANGE SCAN
+ TABLE ACC.

11.380
Buffer Gets

Increases Buffer Gets **+60%**

6.963 vs. 11.380

Detour via Index is inefficient here.

More unlikely, but worse

(Got rid of some inefficient INDEX ACCESS by forcing the CBO into TABLE ACC. FULL)

/*+ NO_INDEX(MAILORDERPOS) */

DBMS_XPLAN.DISPLAY_CURSOR()

Id	Operation	Name	Starts	E-Rows	E-Bytes	Cost (%CPU)	E-Time	A-Rows	A-Time	Buffers
0	SELECT STATEMENT		1			1307 (100)		1	00:00:00.03	4787
* 1	TABLE ACCESS FULL	MAILORDERPOS	1	1	26	1307 (1)	00:00:01	1	00:00:00.03	4787

TABLE ACCESS FULL

4787
Buffer Gets

Increases Buffer Gets **Factor 1.000**

4 vs. 4787

TABLE ACCESS FULL is inefficient here.

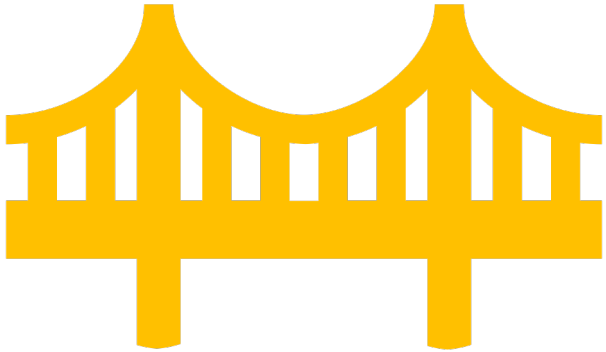
Let the optimizer run!

It's way more flexible than

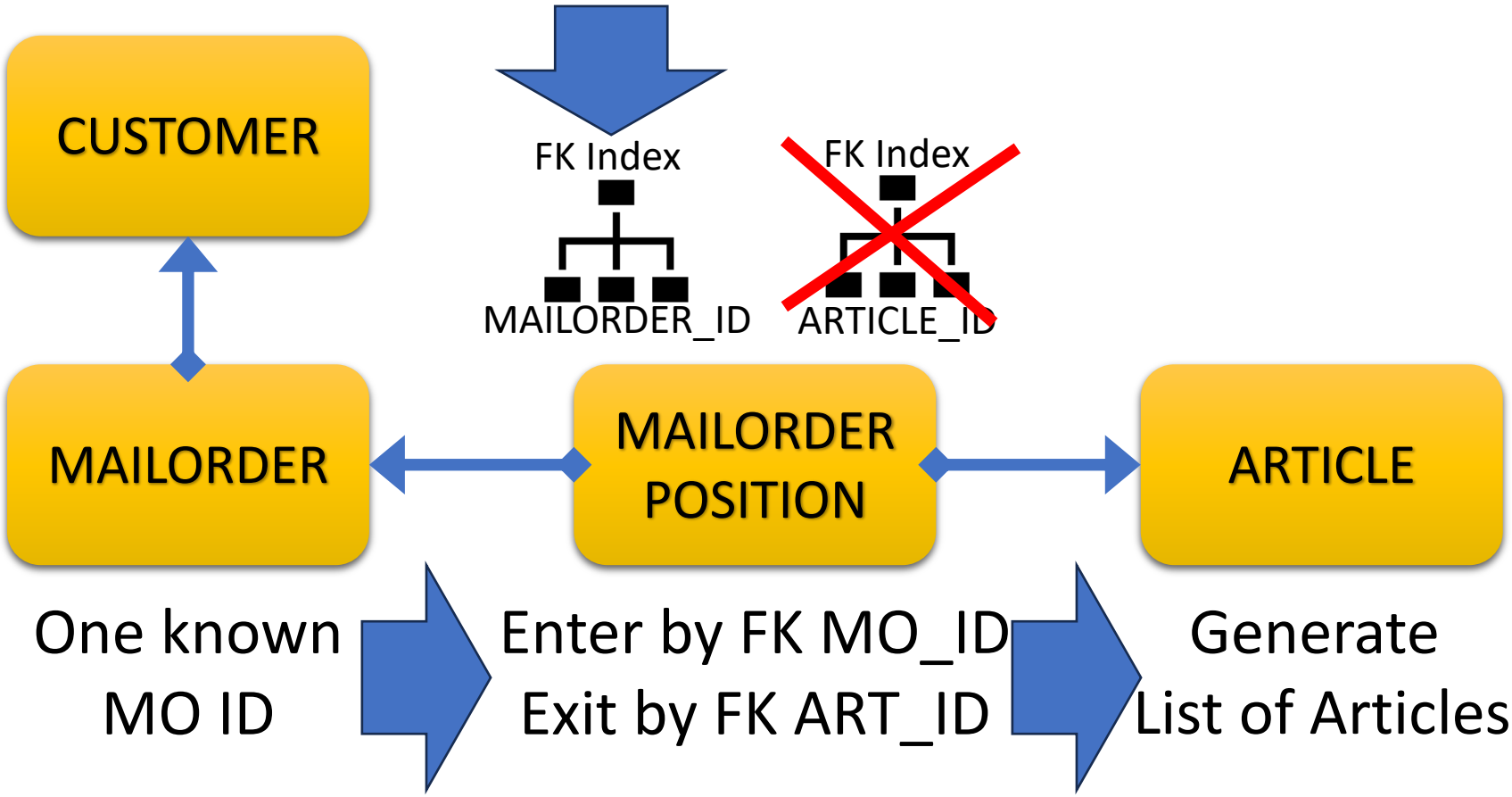
~~Hints~~
~~Stored Outlines~~
~~SQL Profiles~~

Goldene Brücken

offering the CBO a graceful exit



Query All Articles of One Order



BUT:

Only one Index at a time!

ART_ID comes from table

Query All Articles of One Order

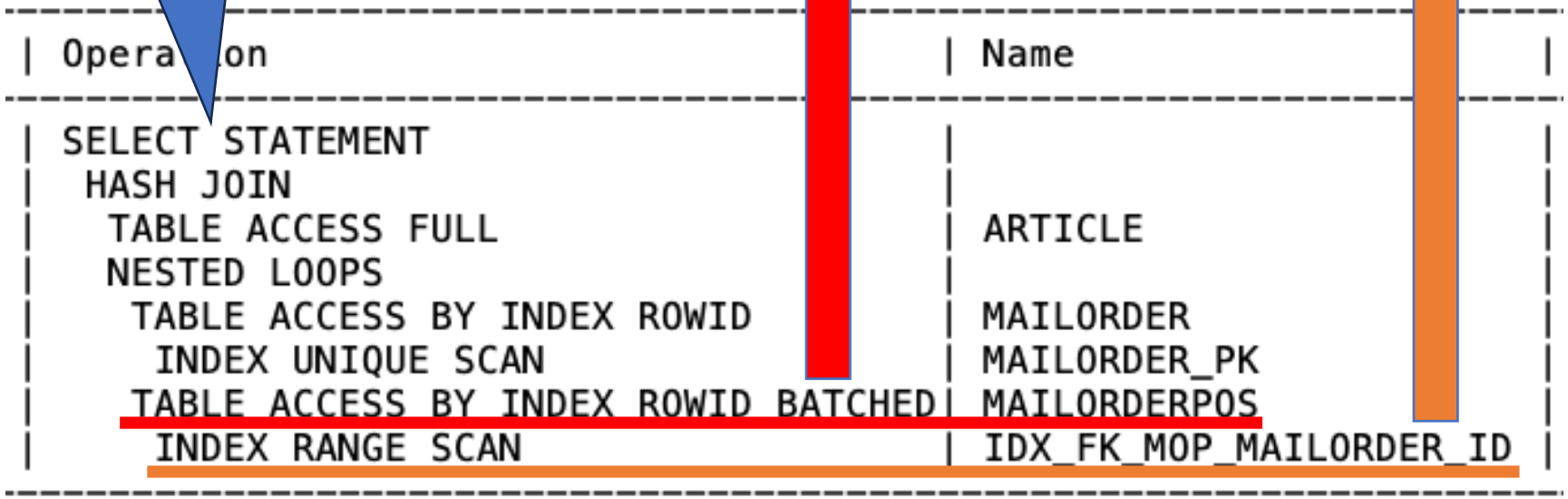
```
select /* SHOWCASE1 */
       mo.id as MO_ID, mo.mo_no, a.id as A_ID, a.art_name
from
       mailorder mo, mailorderpos mop, article a
where
       mop.mop_mailorder_id = mo.id
  and mop.mop_article_id = a.id
  and mo.id = 1739
```

Two Segments
Many large IOs
(only 1 Index used)

4800 Buffer Gets

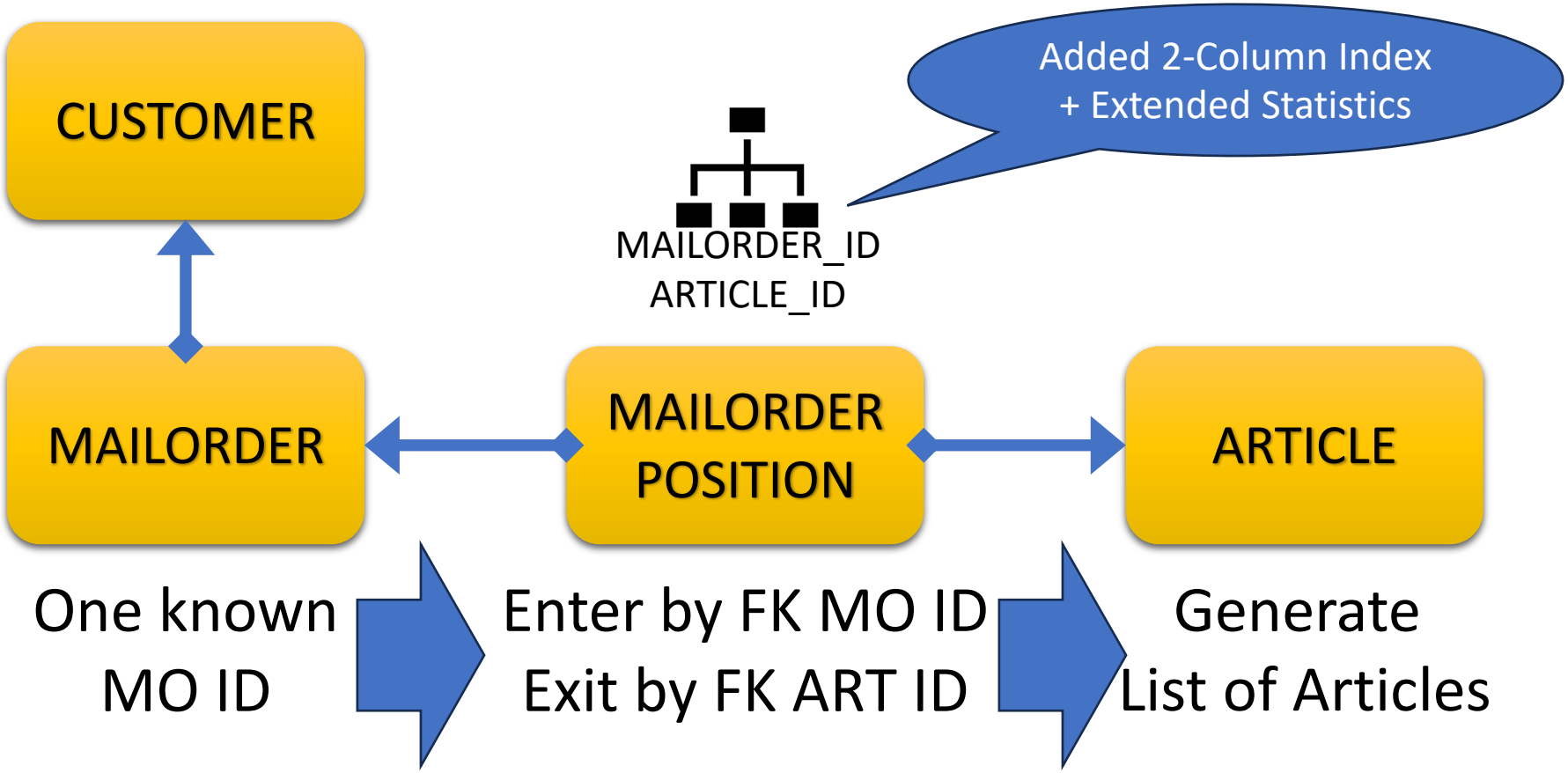
from Table

from Index



Operation	Name
SELECT STATEMENT	
HASH JOIN	
TABLE ACCESS FULL	ARTICLE
NESTED LOOPS	
TABLE ACCESS BY INDEX ROWID	MAILORDER
INDEX UNIQUE SCAN	MAILORDER_PK
TABLE ACCESS BY INDEX ROWID BATCHED	MAILORDERPOS
INDEX RANGE SCAN	IDX_FK_MOP_MAILORDER_ID

Query All Articles of One Order



Now:

ART_ID comes from Entrance Index

Query All Articles of One Order

```
select /* SHOWCASE1-2 */
       mo.id as MO_ID, mo.mo_no, a.id as A_ID, a.art_name
from
       mailorder mo, mailorderpos mop, article a
where
       mop.mop_mailorder_id = mo.id
   and mop.mop_article_id = a.id
   and mo.id = 1739
```

19 Buffer Gets

Operation	Name
SELECT STATEMENT	
HASH JOIN	
TABLE ACCESS FULL	ARTICLE
NESTED LOOPS	
TABLE ACCESS BY INDEX ROWI	MAILORDER
INDEX UNIQUE SCAN	MAILORDER_PK
INDEX RANGE SCAN	IDX_MOP_MOID_ARTID

from Index

from Index

MAILORDERPOS vanished from plan + cost

Buffer Gets down from 4.800 to 19

Adaptive Plans

Count MAILORDERs of One Customer

```
select /* SHOWCASE2 */ cu.cu_no, count(*) x
      from      klm.customer cu,
                klm.mailorder mo
      where     mo.mo_customer_id=cu.id
            and cu.id=73
group by cu.cu_no
order by x desc;
```

Bad (no) Statistics
lead to severe misestimates
Adaptive Plan corrects
=> **4 Buffer Gets**

```
-------------------------------------------------------------------------
|   Id  | Operation                                | Name                |
-------------------------------------------------------------------------
|     0 | SELECT STATEMENT                         |                     |
|     1 |  SORT ORDER BY                           |                     |
|     2 |   HASH GROUP BY                          |                     |
|- *  3 |    HASH JOIN                             |                     |
|     4 |     NESTED LOOPS                         |                     |
|-    5 |      STATISTICS COLLECTOR                |                     |
|     6 |       VIEW                               | VW_GBF_7            |
|     7 |        TABLE ACCESS BY INDEX ROWID       | CUSTOMER            |
|   * 8 |         INDEX UNIQUE SCAN                | CUSTOMER_PK         |
|   * 9 |       INDEX RANGE SCAN                   | IDX_FK_MO_CUSTOMER_ID |
|-   10 |     TABLE ACCESS FULL                    | MAILORDER           |
-------------------------------------------------------------------------
```

Changes
HASH JOIN for NESTED LOOP

Buffer Gets 4 instead of ~400

Collects Cardinality Info

Out: TABLE ACC FULL on MO
In: RANGE SCAN on FK INDEX

Let the optimizer run!

Cheap, but Effective I

- Indexes (also Hidden)
  - Multi-Column
  - Function-Based
  - Careful w/ removing Indexes!
- Statistics
  - Extended (think Zodiac vs. Birthday)
  - Hidden Statistics
- Constraints

Cheap, but Effective II

- Gathering good statistics

```
begin
 dbms_stats.gather_table_stats(ownname=>'KLM',
                    tabname=>'MAILORDERPOS',
                    method_opt=>'FOR ALL COLUMNS SIZE SKEWONLY',
                    estimate_percent=>dbms_stats.AUTO_SAMPLE_SIZE,
                    cascade=>true,
                    no_invalidate=>dbms_stats.auto_invalidate,
                    degree=>4);
end;
/
```

you WANT histograms

re-parsing is key

use the power

get HYBRID, too

My 2 Cents

- Do **not rule** – it's the past
- Switching plans is what we want!
- **Better statistics** make better plans
  - Use **histograms**: FREQUENCY, HYBRID, HEIGHT in this order of preference
  - Old stats are old news:
    Refresh statistics **during normal operations**
    Give the machine the **resources** to do so

Careful!

New Statistics
New Plans

Stand By, Observe, Fix

Let the optimizer run!

The best thing
about the future is
that it comes
one day at a time.

~ Abraham Lincoln, Speaker and Murder Victim

Meet & Greet

martin.klier@performing-db.com
www.performing-databases.com

Many national // international events

Next:

DOAG Deutsche ORACLE-Anwendergruppe e.V.

DOAG Conference Nuremberg
November 19-22, 2024

This year:

ORACLE CloudWorld

OUGN Oracle User Group Norway

utoug utah oracle users group

Meetup

performing databases