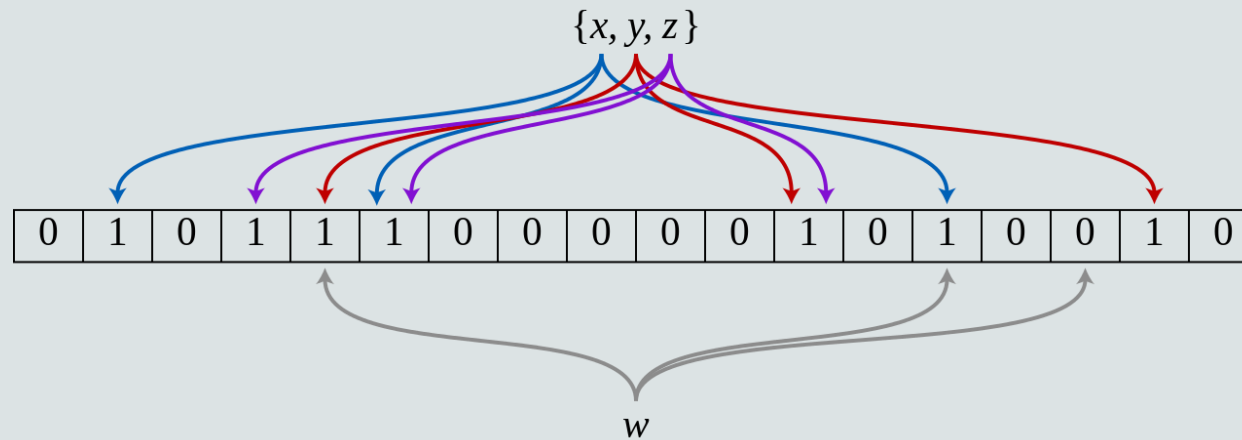


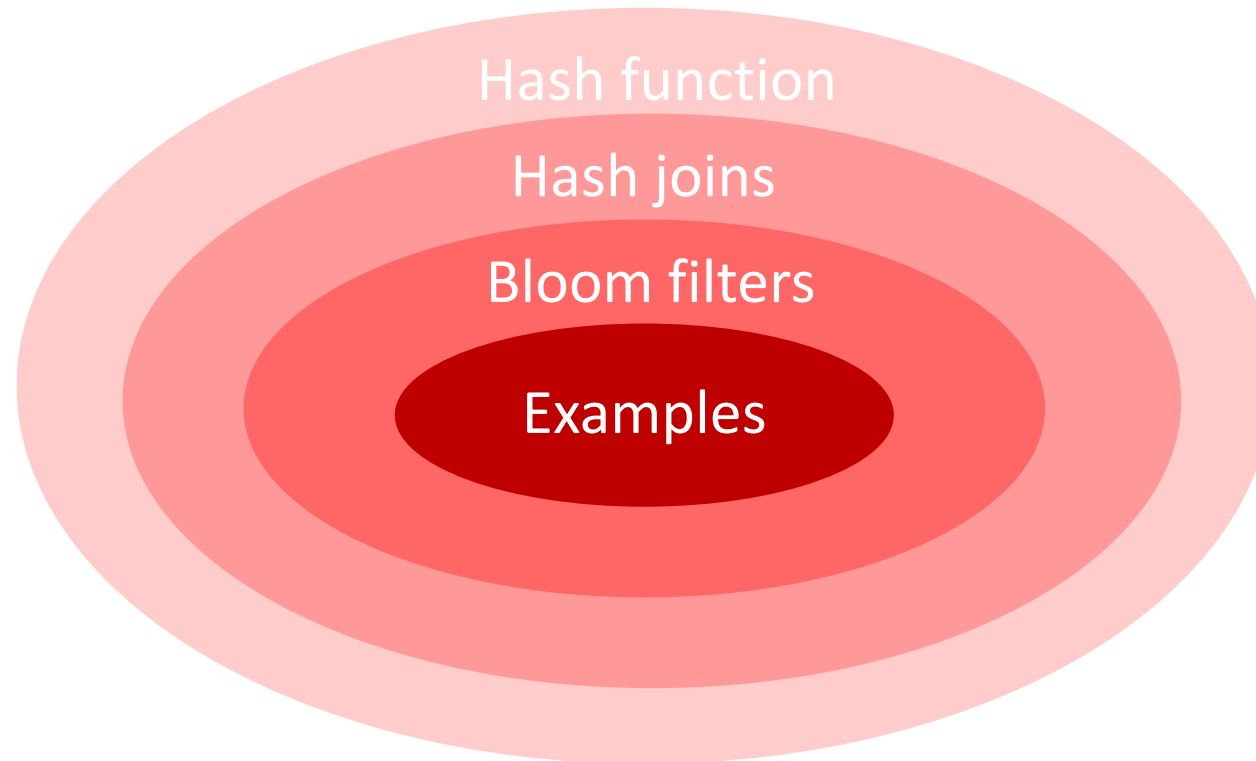
Bloom filters

Toon Koppelaars

Real World Performance team
Database Server Technologies



Agenda



Hash joins and Bloom filters

- Bloom filters (BF) introduced in 10gR2
 - Enable CBO to “filter early” in hash joins
 - Used automatically, visible in execution plan
- Implemented to speed up large parallel hash joins (DWH/BI)
 - BF’s prevent unnecessary row-transport between parallel query slave sets
- As of 11.2.0.4 BF’s may also show up in serial plans

Hash joins explained

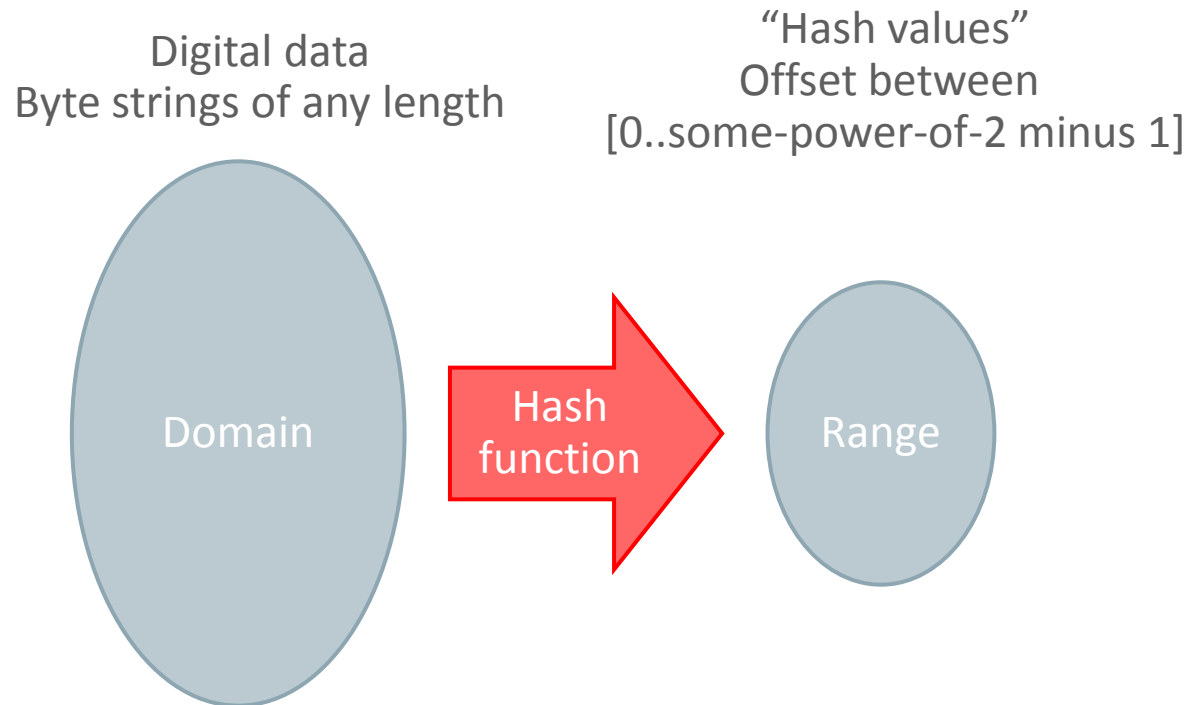
```
select *  
from emp e, dept d  
where d.deptno = e.deptno
```

Id	Operation	Name
0	SELECT STATEMENT	
* 1	HASH JOIN	
2	TABLE ACCESS FULL	DEPT
3	TABLE ACCESS FULL	EMP

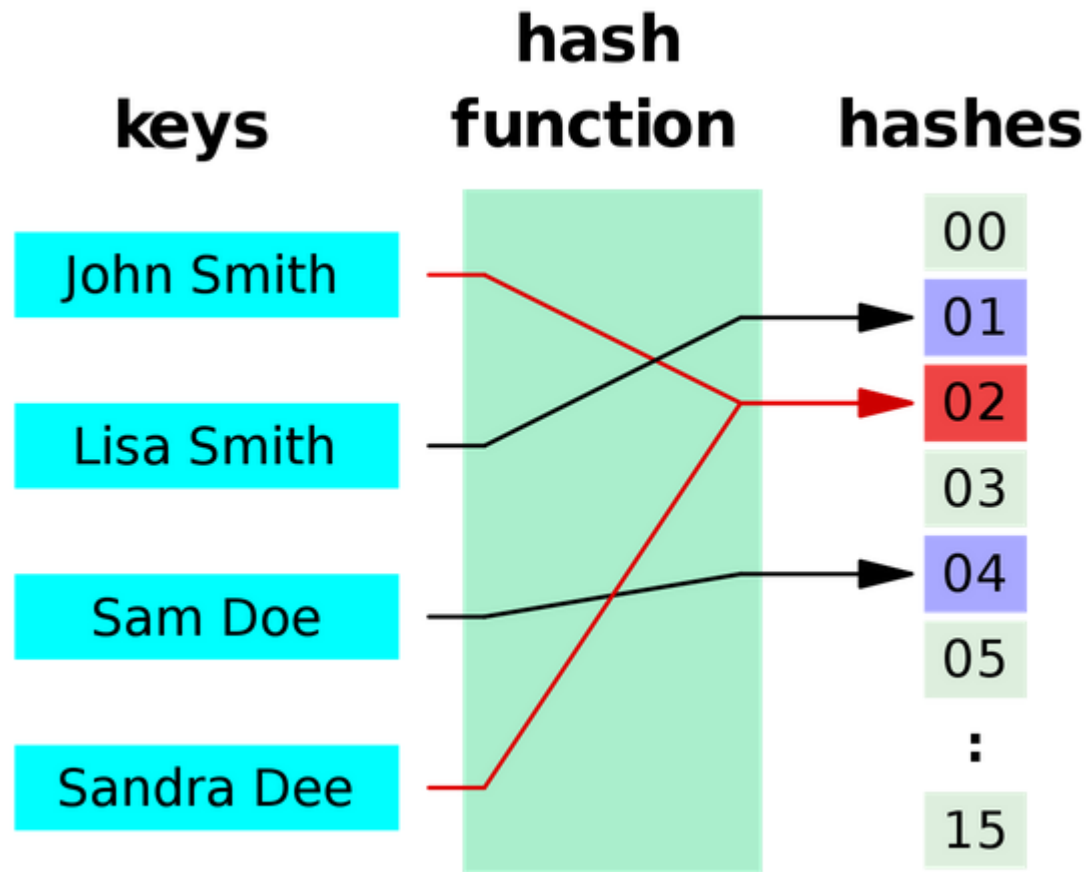
Predicate Information

1 - access("D"."DEPTNO"="E"."DEPTNO")

What is a hash-function?



- Hash function application is cheap
- Hash values are typically “evenly distributed” in range
 - Irrespective of distribution of input values
- Collisions can occur
 - Two domain values map to same hash value
 - Since $\#domain \gg \#range$



Phase 1: build hash table (DEPT)

```
select *  
from emp e, dept d  
where d.deptno = e.deptno
```

Id	Operation	Name
0	SELECT STATEMENT	
* 1	HASH JOIN	
2	TABLE ACCESS FULL	DEPT
3	TABLE ACCESS FULL	EMP

Predicate Information

1 - access("D"."DEPTNO"="E"."DEPTNO")

Scan ↓

DEPTNO	DNAME	LOC
1	ACCOUNTING	NEW YORK
2	RESEARCH	DALLAS
3	SALES	CHICAGO
4	OPERATIONS	BOSTON

Apply hash function to DEPT.DEPTNO to find in which bucket to store

Build phase



Bucketized hash-table in PGA memory



Phase 1 done

```
select *  
from emp e, dept d  
where d.deptno = e.deptno
```

Id	Operation	Name
0	SELECT STATEMENT	
* 1	HASH JOIN	
2	TABLE ACCESS FULL	DEPT
3	TABLE ACCESS FULL	EMP

Predicate Information

1 - access("D"."DEPTNO"="E"."DEPTNO")



Phase 2: probe 2nd table (EMP)

```
select *
from emp e, dept d
where d.deptno = e.deptno
```

Id	Operation	Name
0	SELECT STATEMENT	
* 1	HASH JOIN	
2	TABLE ACCESS FULL	DEPT
3	TABLE ACCESS FULL	EMP

Predicate Information

1 - access("D"."DEPTNO"="E"."DEPTNO")

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	SMITH	CLERK	7902	17-DEC-80	800		2
2	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	3
3	WARD	SALESMAN	7698	22-FEB-81	1250	500	4
4	JONES	MANAGER	7839	02-APR-81	2975		5
5	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	6
6	BLAKE	MANAGER	7839	01-MAY-81	2850		7
7	CLARK	MANAGER	7839	09-JUN-81	2450		8
8	SCOTT	ANALYST	7566	19-APR-87	3000		9
9	KING	PRESIDENT		17-NOV-81	5000		10
10	TURNER	SALESMAN	7698	08-SEP-81	1500	0	11
11	ADAMS	CLERK	7788	23-MAY-87	1100		12
12	JAMES	CLERK	7698	03-DEC-81	950		13
13	FORD	ANALYST	7566	03-DEC-81	3000		14
14	MILLER	CLERK	7782	23-JAN-82	1300		15

Scan

Apply hash function to EMP.DEPTNO to find in which bucket to probe for matching DEPT record
If match found: join and return row

Probe phase



Hash joins parallel execution plan

Assuming DOP = 3

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT				
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10002	Q1,02	P->S	QC (RAND)
* 3	HASH JOIN		Q1,02	PCWP	
4	PX RECEIVE		Q1,02	PCWP	
5	PX SEND HASH	:TQ10000	Q1,00	P->P	HASH
6	PX BLOCK ITERATOR		Q1,00	PCWC	
7	TABLE ACCESS FULL	DEPT	Q1,00	PCWP	
8	PX RECEIVE		Q1,02	PCWP	
9	PX SEND HASH	:TQ10001	Q1,01	P->P	HASH
10	PX BLOCK ITERATOR		Q1,01	PCWC	
11	TABLE ACCESS FULL	EMP	Q1,01	PCWP	

Predicate Information (identified by operation id):

3 - access("D"."DEPTNO"="E"."DEPTNO")



Slave set 2



Slave set 1



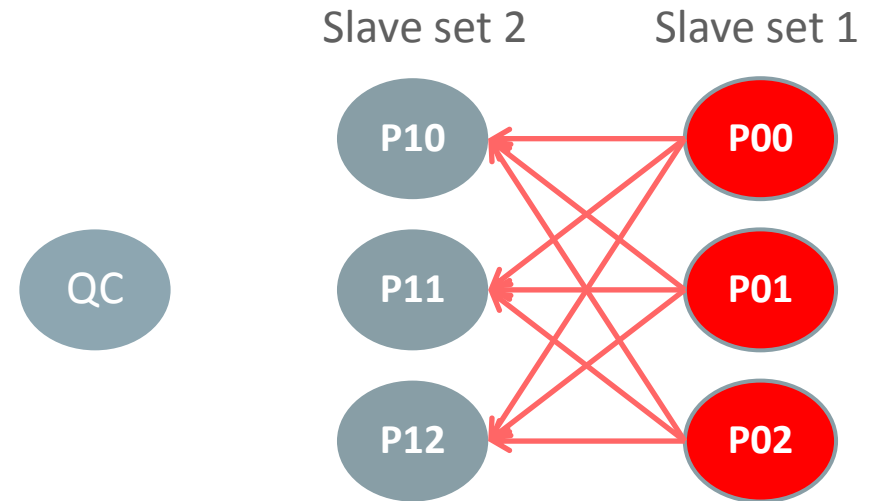
Build phase

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT				
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10002	Q1,02	P->S	QC (RAND)
* 3	HASH JOIN		Q1,02	PCWP	
4	PX RECEIVE		Q1,02	PCWP	
5	PX SEND HASH	:TQ10000	Q1,00	P->P	HASH
6	PX BLOCK ITERATOR		Q1,00	PCWC	
7	TABLE ACCESS FULL	DEPT	Q1,00	PCWP	
8	PX RECEIVE		Q1,02	PCWP	
9	PX SEND HASH	:TQ10001	Q1,01	P->P	HASH
10	PX BLOCK ITERATOR		Q1,01	PCWC	
11	TABLE ACCESS FULL	EMP	Q1,01	PCWP	

Predicate Information (identified by operation id):

3 - access("D"."DEPTNO"="E"."DEPTNO")

Hash PQ Distrib:
conceptually distributes rows based on
 $\text{mod}(\text{hash}(\text{deptno}), 3)$



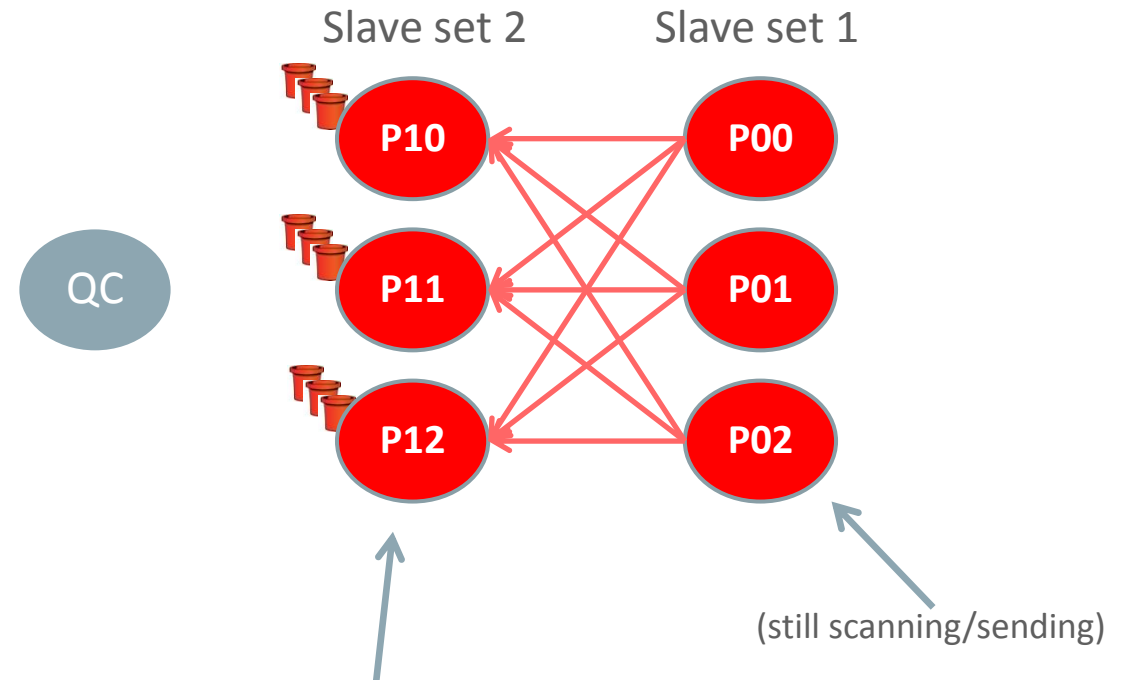
Lines 7,6,5: Scan DEPT, hash send to SS2

Build phase

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT				
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10002	Q1,02	P->S	QC (RAND)
* 3	HASH JOIN		Q1,02	PCWP	
4	PX RECEIVE		Q1,02	PCWP	
5	PX SEND HASH	:TQ10000	Q1,00	P->P	HASH
6	PX BLOCK ITERATOR		Q1,00	PCWC	
7	TABLE ACCESS FULL	DEPT	Q1,00	PCWP	
8	PX RECEIVE		Q1,02	PCWP	
9	PX SEND HASH	:TQ10001	Q1,01	P->P	HASH
10	PX BLOCK ITERATOR		Q1,01	PCWC	
11	TABLE ACCESS FULL	EMP	Q1,01	PCWP	

Predicate Information (identified by operation id):

3 - access("D"."DEPTNO"="E"."DEPTNO")



Line 4/3: Build hash table for DEPT rows received

Hash joins parallel execution plan

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT				
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10002	Q1,02	P->S	QC (RAND)
* 3	HASH JOIN		Q1,02	PCWP	
4	PX RECEIVE		Q1,02	PCWP	
5	PX SEND HASH	:TQ10000	Q1,00	P->P	HASH
6	PX BLOCK ITERATOR		Q1,00	PCWC	
7	TABLE ACCESS FULL	DEPT	Q1,00	PCWP	
8	PX RECEIVE		Q1,02	PCWP	
9	PX SEND HASH	:TQ10001	Q1,01	P->P	HASH
10	PX BLOCK ITERATOR		Q1,01	PCWC	
11	TABLE ACCESS FULL	EMP	Q1,01	PCWP	

Predicate Information (identified by operation id):

3 - access("D"."DEPTNO"="E"."DEPTNO")



Build phase ready

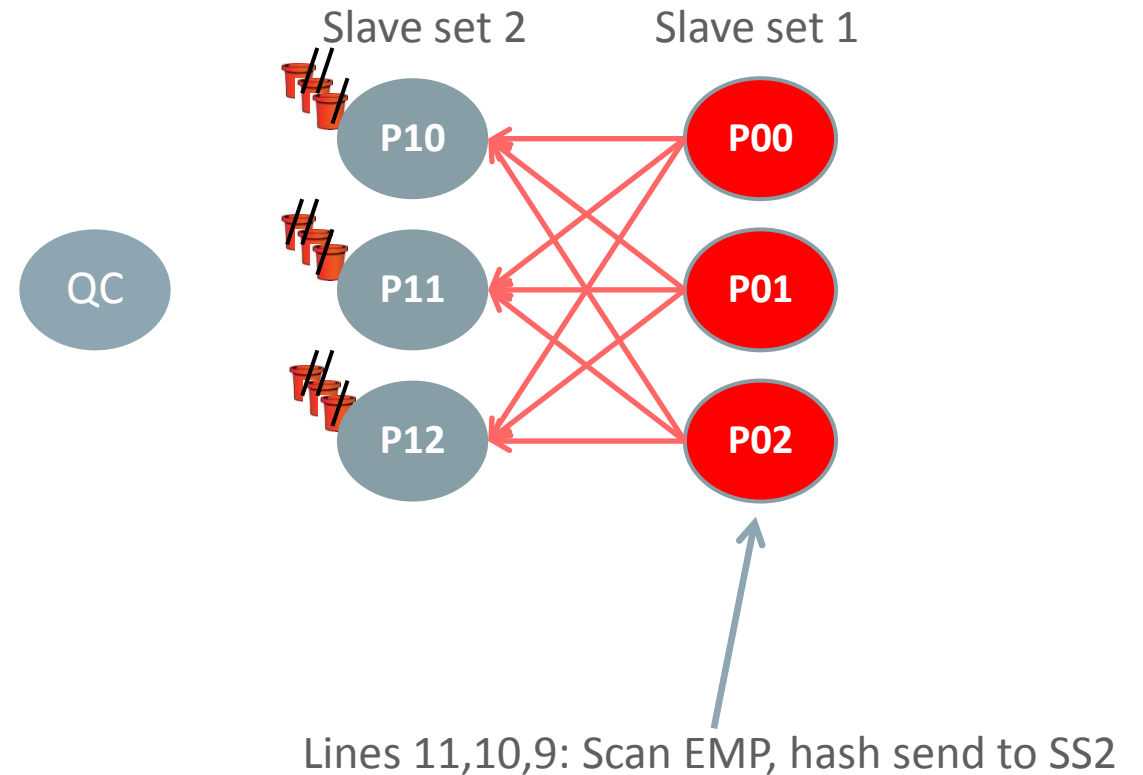
Probe phase

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT				
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10002	Q1,02	P->S	QC (RAND)
* 3	HASH JOIN		Q1,02	PCWP	
4	PX RECEIVE		Q1,02	PCWP	
5	PX SEND HASH	:TQ10000	Q1,00	P->P	HASH
6	PX BLOCK ITERATOR		Q1,00	PCWC	
7	TABLE ACCESS FULL	DEPT	Q1,00	PCWP	
8	PX RECEIVE		Q1,02	PCWP	
9	PX SEND HASH	:TQ10001	Q1,01	P->P	HASH
10	PX BLOCK ITERATOR		Q1,01	PCWC	
11	TABLE ACCESS FULL	EMP	Q1,01	PCWP	

Predicate Information (identified by operation id):

3 - access("D"."DEPTNO"="E"."DEPTNO")

Same $\text{mod}(\text{hash}(\text{deptno}), 3)$ distribution used

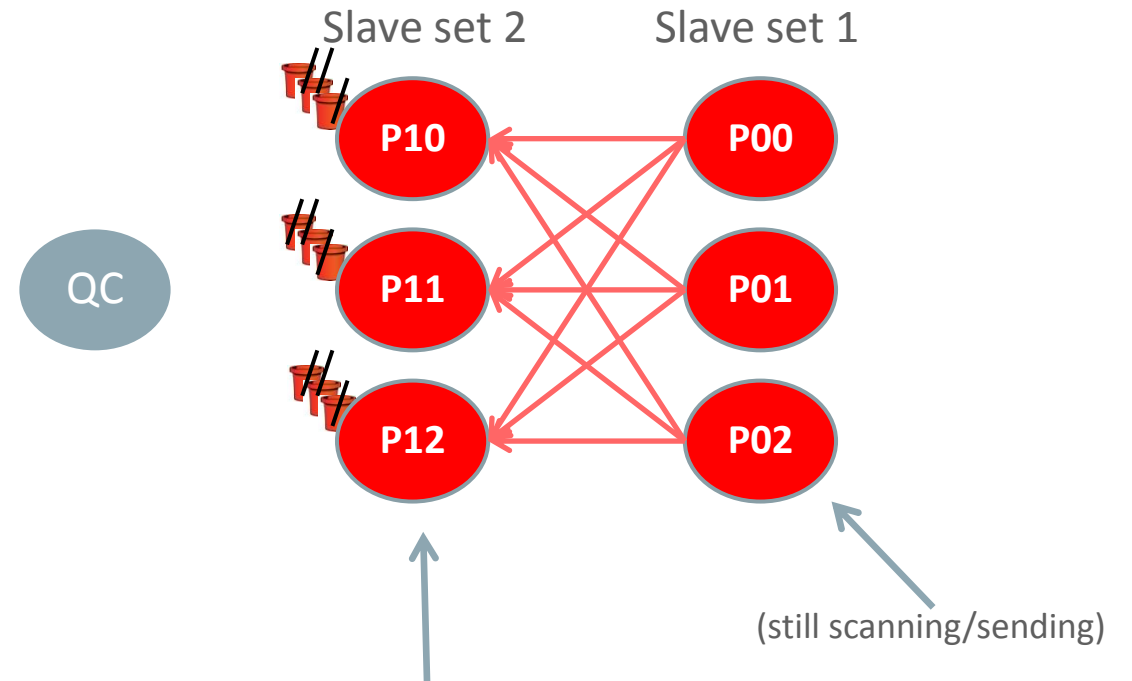


Probe phase

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT				
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10002	Q1,02	P->S	QC (RAND)
* 3	HASH JOIN		Q1,02	PCWP	
4	PX RECEIVE		Q1,02	PCWP	
5	PX SEND HASH	:TQ10000	Q1,00	P->P	HASH
6	PX BLOCK ITERATOR		Q1,00	PCWC	
7	TABLE ACCESS FULL	DEPT	Q1,00	PCWP	
8	PX RECEIVE		Q1,02	PCWP	
9	PX SEND HASH	:TQ10001	Q1,01	P->P	HASH
10	PX BLOCK ITERATOR		Q1,01	PCWC	
11	TABLE ACCESS FULL	EMP	Q1,01	PCWP	

Predicate Information (identified by operation id):

3 - access("D"."DEPTNO"="E"."DEPTNO")



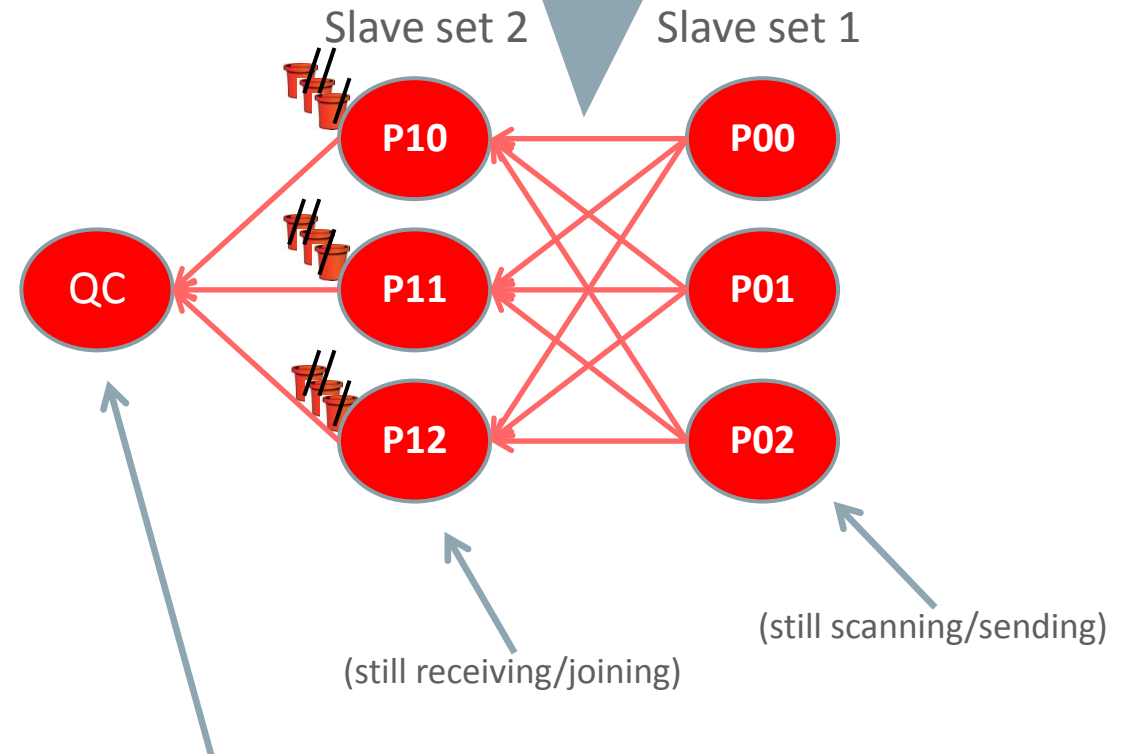
Line 8/3: Join/probe hash table for EMP rows received

Probe phase

Id	Operation	Name	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT				
1	PX COORDINATOR				
2	PX SEND QC (RANDOM)	:TQ10002	Q1,02	P->S	QC (RAND)
* 3	HASH JOIN		Q1,02	PCWP	
4	PX RECEIVE		Q1,02	PCWP	
5	PX SEND HASH	:TQ10000	Q1,00	P->P	HASH
6	PX BLOCK ITERATOR		Q1,00	PCWC	
7	TABLE ACCESS FULL	DEPT	Q1,00	PCWP	
8	PX RECEIVE		Q1,02	PCWP	
9	PX SEND HASH	:TQ10001	Q1,01	P->P	HASH
10	PX BLOCK ITERATOR		Q1,01	PCWC	
11	TABLE ACCESS FULL	EMP	Q1,01	PCWP	

Predicate Information (identified by operation id):

3 - access("D"."DEPTNO"="E"."DEPTNO")



Lines 2/1: Receive resulting joined rows

Filter early

- What do we mean by “filter early”?
 - In the hierarchical execution plan tree
 - Eliminate rows as soon (early) as possible
 - Preventing their (expensive) flow through parent execution stages in plan

Filter early: join example

Remember this plan
BF's operate in same way

```
select d.dname,e.ename
from dept d, emp e
where d.deptno = e.deptno
and d.deptno between 130 and 150
```

Id	Operation	Name	
0	SELECT STATEMENT		
* 1	HASH JOIN		
2	TABLE ACCESS BY INDEX ROWID BATCHED	DEPT	
* 3	INDEX RANGE SCAN	DEPT_PK	
* 4	TABLE ACCESS FULL	EMP	<== Filter!


Predicate Information (identified by operation id):

- 1 - access("D"."DEPTNO"="E"."DEPTNO")
- 3 - access("D"."DEPTNO">=130 AND "D"."DEPTNO"<=150)
- 4 - filter(("E"."DEPTNO">=130 AND "E"."DEPTNO"<=150)) <== Filter!

Bloom filter concept

Bloom filter concept

- Memory structure representing a *set of values* (say S)
 - “representing” → member values are encoded in specific manner
- Memory structure can be used to answer:
 - Is a given value x , member of set S ?
- Two possible answers
 - Definitely not
 - Possibly yes



Two operators on structure:
1) Add value x
2) Test membership

Conceived by Burton Howard Bloom in 1970

Bloom filter concept

- BF(m,k)

- m = size of memory structure in bits → array of m bits

- k = number of hash functions used to encode value (hf_1, \dots, hf_k)

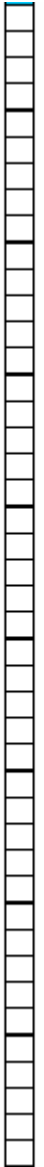
- Range of hash functions is: [0..m]

- Bit array initially filled with 0's

- Adding (encoding) a value x:

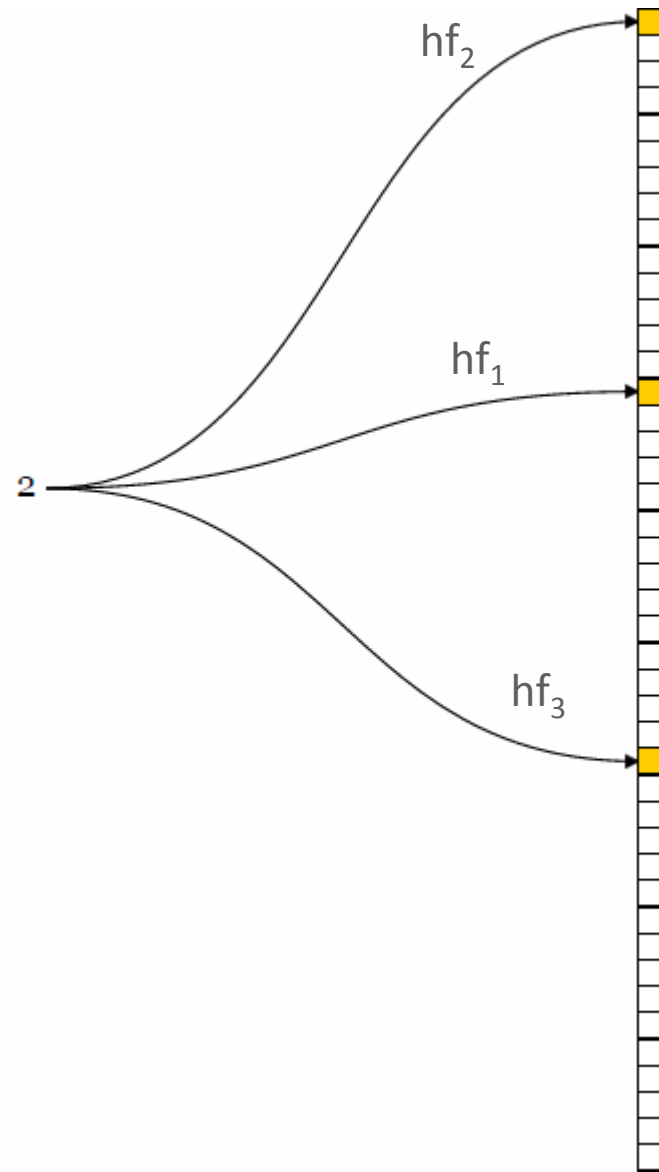
- Apply each hash-function to value x → this produces k hash-values

- Set bit in array at each hash-value's position

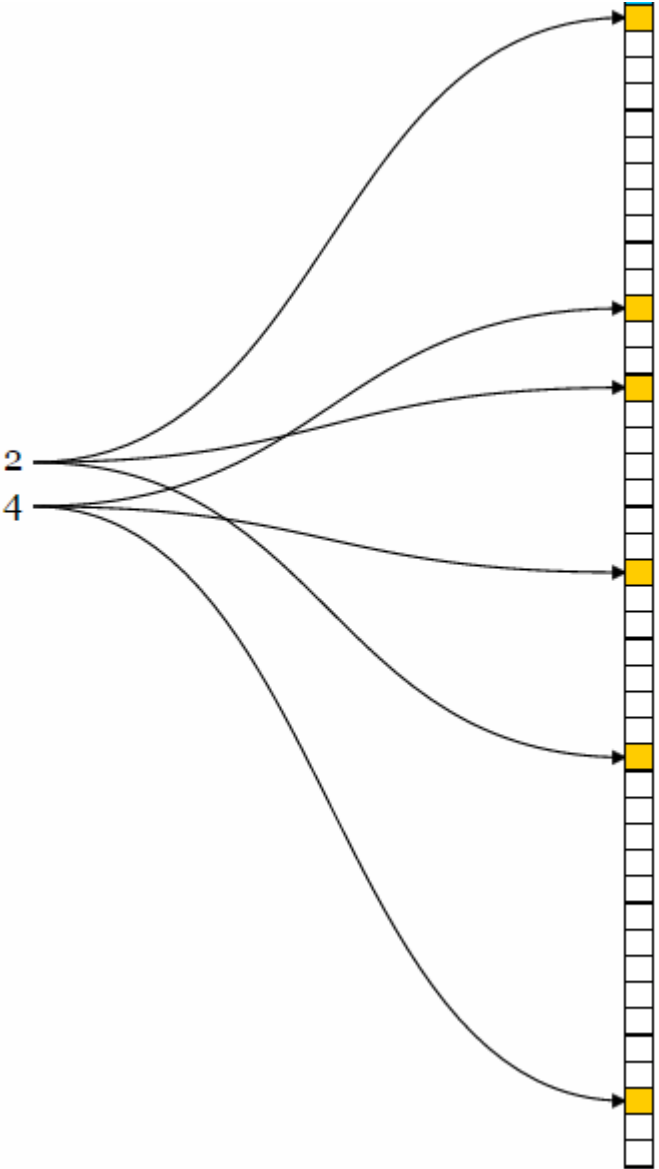


Bloom filter: adding value 2

Assuming 3 hash functions used

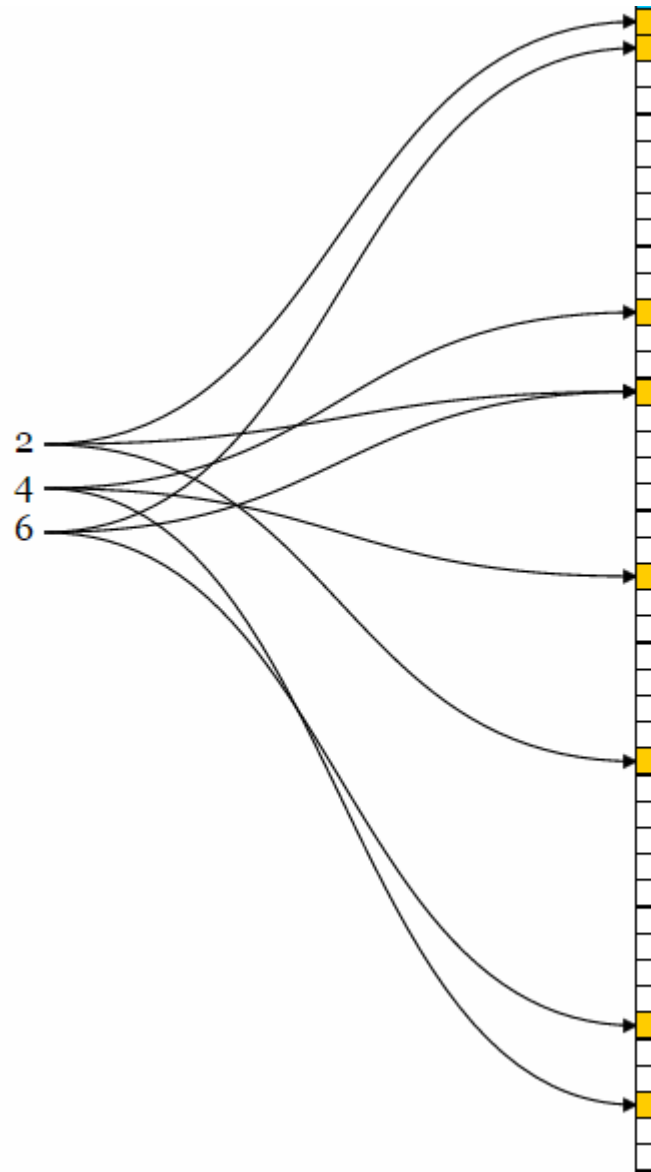


Bloom filter: adding value 4



Bloom filter: adding value 6

Note:
one of hashes of 6 is same as
one of hashes of 2

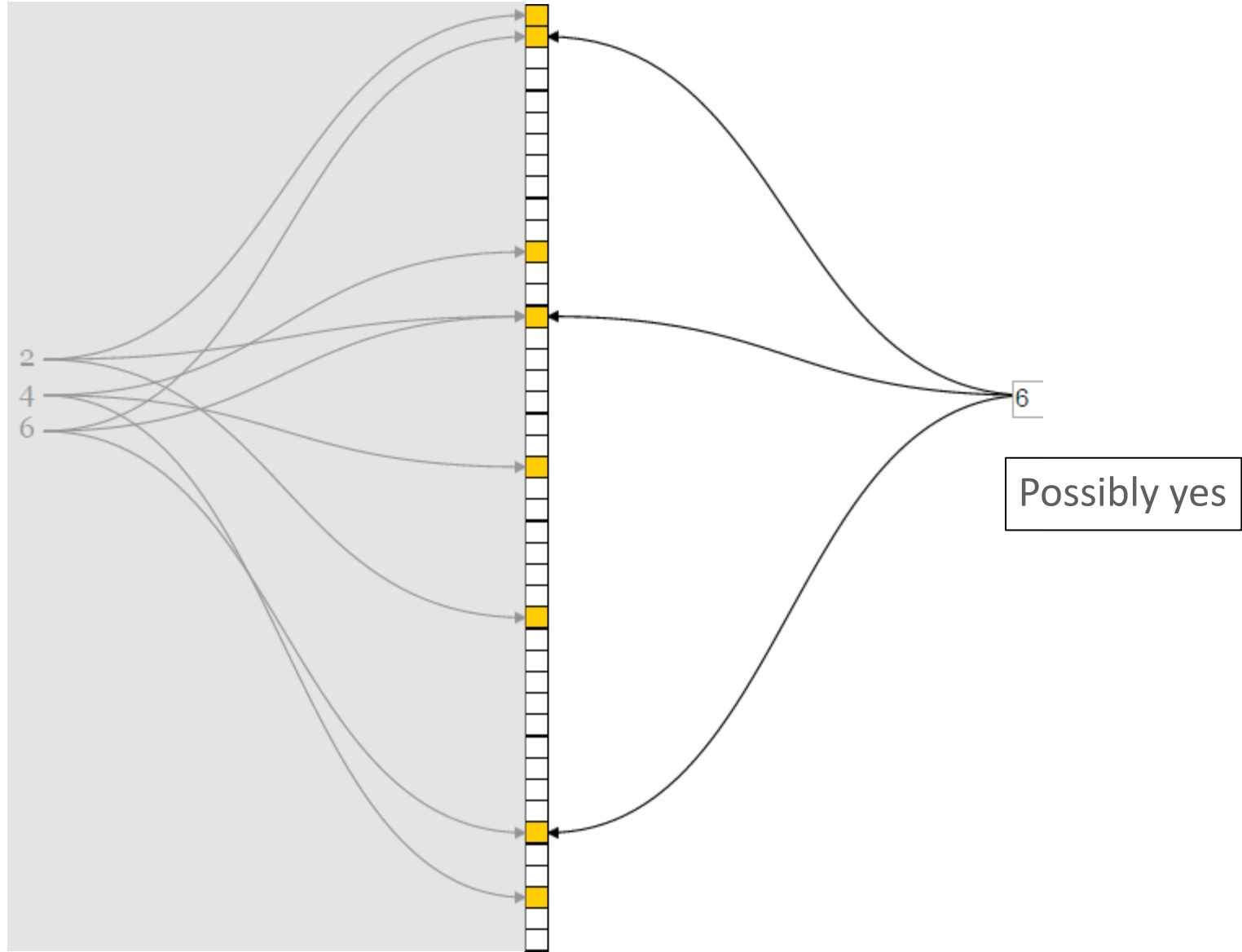


Bloom filter: testing for membership

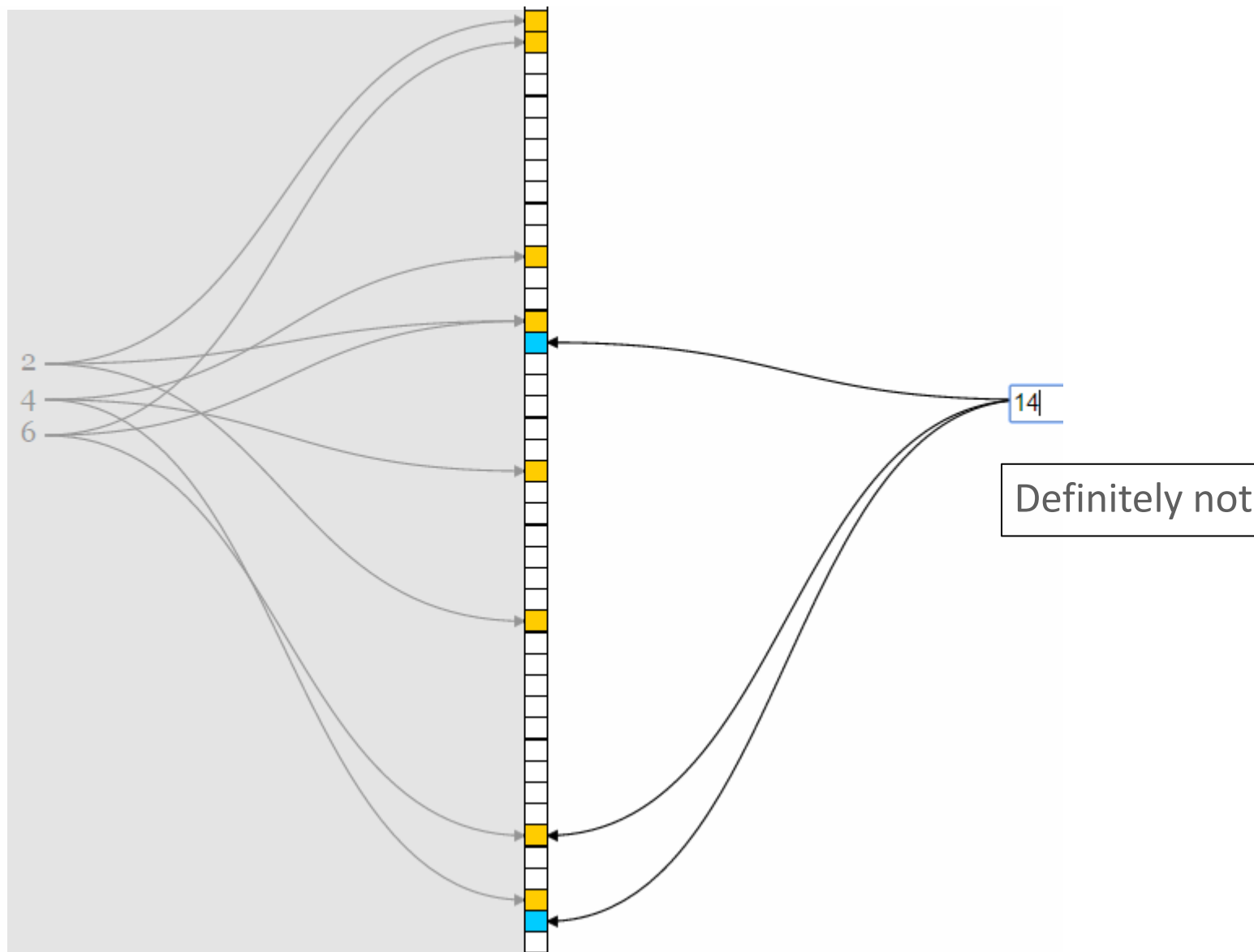
- Once bloom filter is populated with complete set, we can test for membership of any given value
 - By just applying hash-functions to given value
 - And checking if all hashed positions in bit-array are filled with 1
- If any not filled → then *definitely not* a member of the set
- If all filled → then *possibly (!)* member of the set

- Two possible answers
 - Definitely not
 - Possibly yes

Testing value 6



Testing value 14



False positives possible

- Likelihood increases when bit-array is sized too small
- Demo: <https://www.jasondavies.com/bloomfilter/>

Bloom filter sizing

- CBO uses estimated NDV to size Bloom filter
 - Number of Distinct Values
- Demo: <http://hur.st/bloomfilter?n=1000000&p=0.01>

Why use Bloom filters?

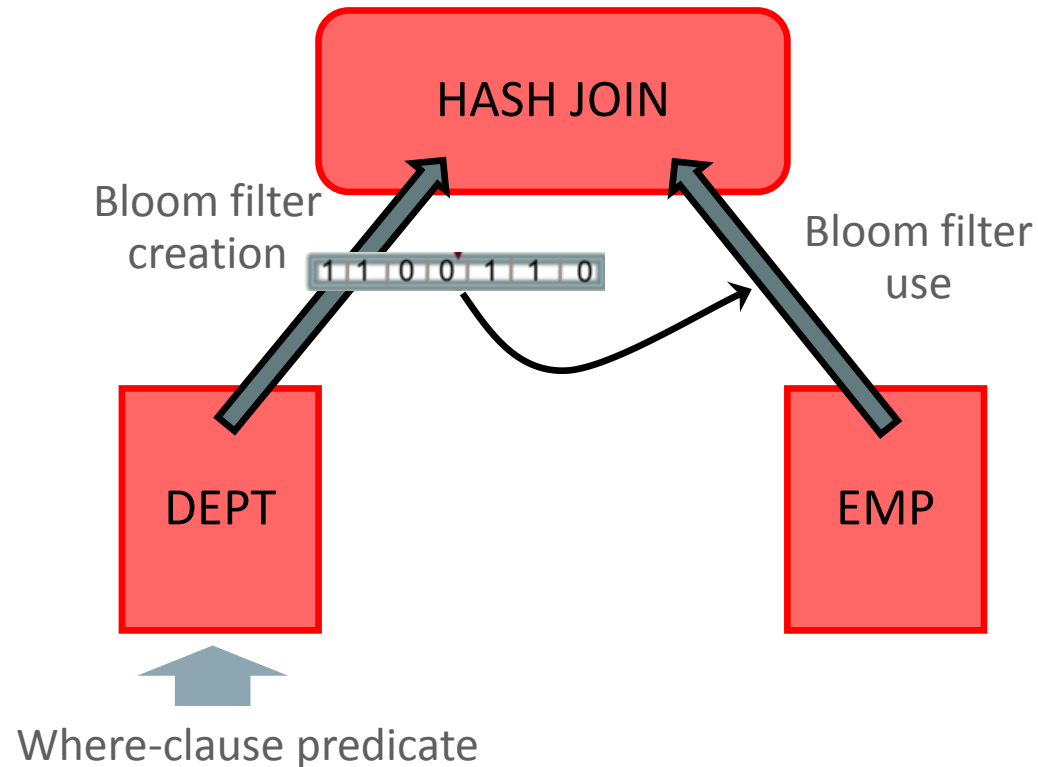
1. Memory structure is small
 - Around 9-10 bits per (distinct) element required for a 1% false positive probability
2. Add and test operators are fast
 - Cheap hash + fast set/test bit-operations

➔ Bloom filters are cheap to create, use and dispose of, on-the-fly

How can Bloom filter help filter early in hash joins?

- BF use-cases in hash-join are of the form:
 - In BUILD stage of execution plan: BF CREATE
 - All join-column values of build-table added to BF
 - BF filter = bit signature of join condition
 - Then during PROBE stage of execution plan: BF USE
 - Every join-column value of probe-table tested against BF
 - BF filter used to filter ‘definitely not member’ cases

Early filtering



- Bloom filter encodes subset of DEPT PK join-values
- Used during scan of EMP FK join-values
- Bloom filter prevents flow of EMP rows to Hash-Join operator for which there would not be matching DEPT row

Bloom filter examples

- Parallel join filter
- Serial subquery filter
- Multiple BF's (rightdeep tree)
- Bloom pruning filter

Parallel join filter

```
select d.dname,e.ename
from dept d, emp e
where d.deptno = e.deptno
      and d.loc = 'UTRECHT'
```

Filter on non-join column (LOC)
used to 'early filter' EMP rows

False positives caught in line 3

What is saved?

1. Less rows distributed SS1 → SS2 during probe phase
2. Bit vs byte-string operations

Id	Operation	Name
0	SELECT STATEMENT	
1	PX COORDINATOR	
2	PX SEND OC (RANDOM)	:TQ10002
* 3	HASH JOIN	
4	JOIN FILTER CREATE	:BF0000
5	PX RECEIVE	
6	PX SEND HASH	:TQ10000
7	PX BLOCK ITERATOR	
* 8	TABLE ACCESS FULL	DEPT
9	PX RECEIVE	
10	PX SEND HASH	:TQ10001
11	JOIN FILTER USE	:BF0000
12	PX BLOCK ITERATOR	
* 13	TABLE ACCESS FULL	EMP

Predicate Information (identified by operation id):

- 3 - access("D"."DEPTNO"="E"."DEPTNO")
- 8 - filter("D"."LOC"='UTRECHT')
- 13 - filter(SYS_OP_BLOOM_FILTER(:BF0000,"E"."DEPTNO"))

Serial subquery filter

```
select d.dname,e.dism
from dept d
      ,(select deptno
         ,count(distinct ename)
         as disnm
        from emp
         group by deptno) e
where d.loc='UTRECHT'
and d.deptno = e.deptno
```

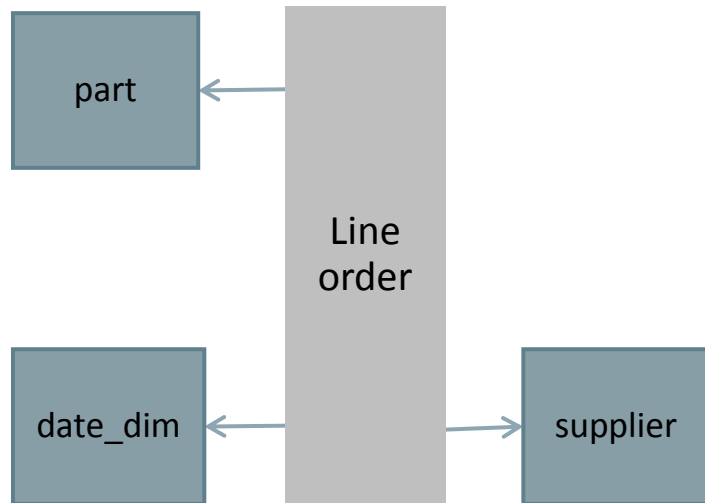
BF push through GroupBy

Id	Operation	Name	Starts	A-Rows
0	SELECT STATEMENT		1	11
1	HASH GROUP BY		1	11
* 2	HASH JOIN		1	44
3	JOIN FILTER CREATE	:BF0000	1	11
* 4	TABLE ACCESS FULL	DEPT	1	11
5	VIEW	VM_NWVW_1	1	44
6	HASH GROUP BY		1	44
7	JOIN FILTER USE	:BF0000	1	44
* 8	TABLE ACCESS FULL	EMP	1	44

Predicate Information (identified by operation id):

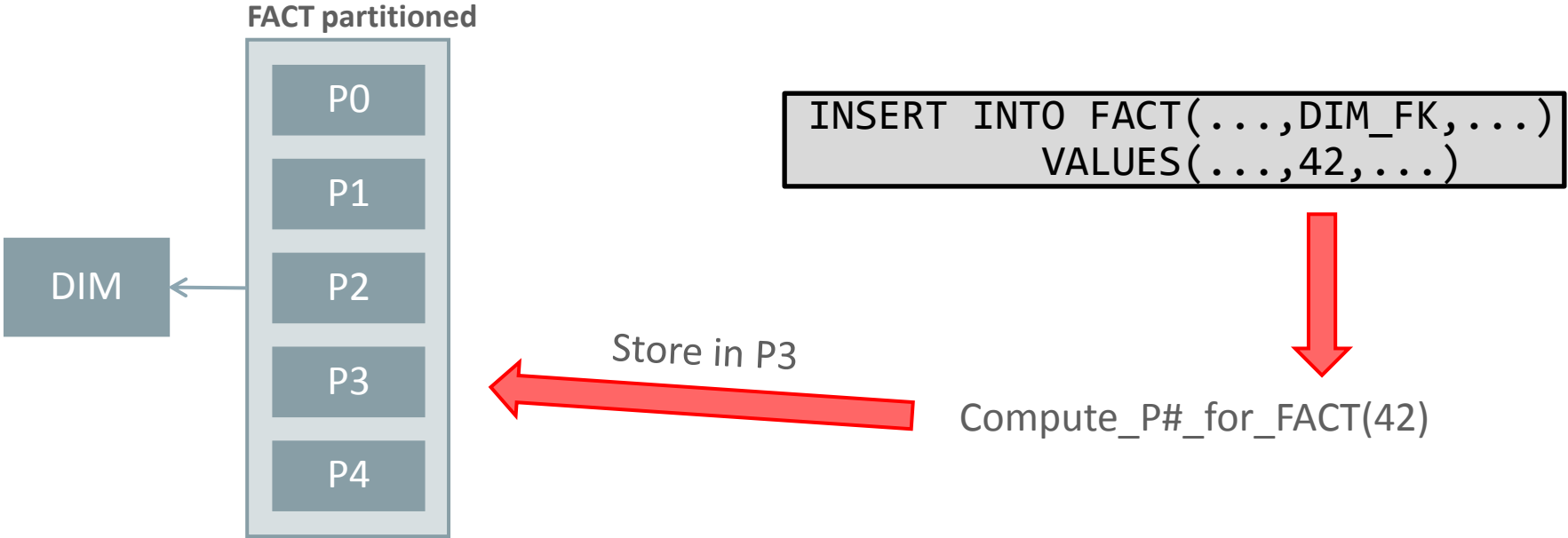
```
2 - access("D"."DEPTNO"="$vm_col_2")
4 - filter("D"."LOC"='UTRECHT')
8 - filter(SYS_OP_BLOOM_FILTER(:BF0000,"DEPTNO"))
```

Multiple Bloom filters



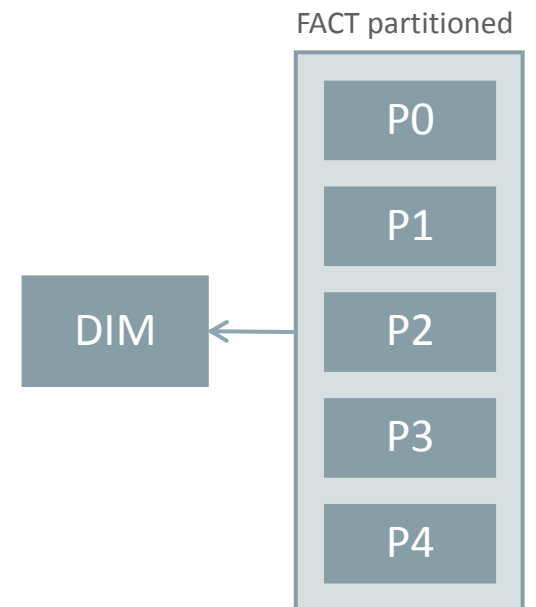
Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		63	7371
1	PX COORDINATOR			
2	PX SEND QC (ORDER)	:TQ10003	63	7371
3	SORT GROUP BY		63	7371
4	PX RECEIVE		63	7371
5	PX SEND RANGE	:TQ10002	63	7371
6	HASH GROUP BY		63	7371
* 7	HASH JOIN		430	50310
8	JOIN FILTER CREATE	:BF0000	4013	133K
9	PX RECEIVE		4013	133K
10	PX SEND BROADCAST	:TQ10000	4013	133K
11	PX BLOCK ITERATOR		4013	133K
* 12	TABLE ACCESS INMEMORY FULL	SUPPLIER	4013	133K
* 13	HASH JOIN		10589	858K
14	JOIN FILTER CREATE	:BF0001	31	1023
* 15	TABLE ACCESS INMEMORY FULL	DATE_DIM	31	1023
* 16	HASH JOIN		845K	40M
17	JOIN FILTER CREATE	:BF0002	2332	65296
18	PX RECEIVE		2332	65296
19	PX SEND BROADCAST	:TQ10001	2332	65296
20	PX BLOCK ITERATOR		2332	65296
* 21	TABLE ACCESS INMEMORY FULL	PART	2332	65296
22	JOIN FILTER USE	:BF0000	300M	6294M
23	JOIN FILTER USE	:BF0001	300M	6294M
24	JOIN FILTER USE	:BF0002	300M	6294M
25	PX BLOCK ITERATOR		300M	6294M
* 26	TABLE ACCESS INMEMORY FULL	LINEORDER	300M	6294M

Bloom pruning filter (11g)



Bloom pruning filter (11g)

- Shows up as “*Partition join filter create*”
 - When joining DIM to FACT and join-column is input for partitioning scheme of FACT
 - Build phase:
 - FACT’s partition function is applied to DIM PK values to find P#’s of FACT that would hold such values
 - These P#’s are added to Bloom filter
 - Probe phase:
 - Bloom filter is used when scanning FACT
 - Only partitions whose P#’s are in Bloom filter will be scanned



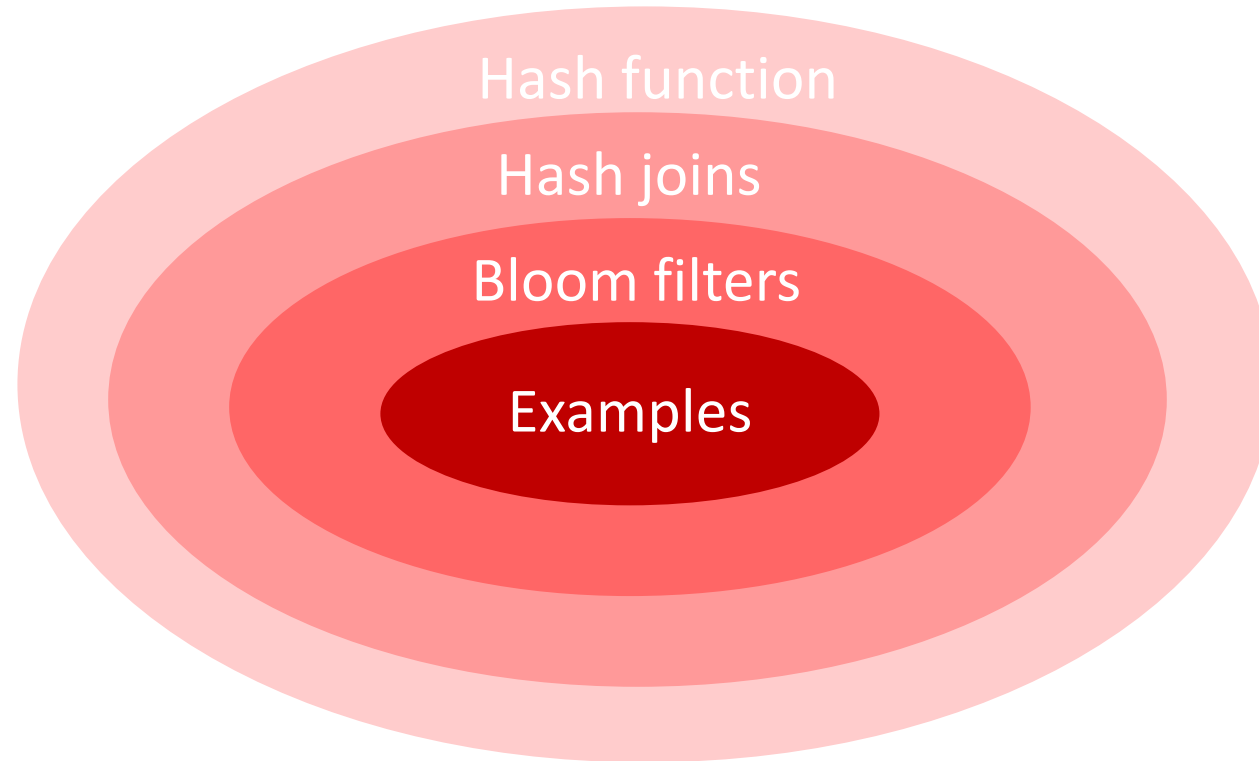
Bloom pruning filter

Id	Operation	Name	Pstart	Pstop
0	SELECT STATEMENT			
1	SORT AGGREGATE			
2	PX COORDINATOR			
3	PX SEND QC (RANDOM)	:TQ10001		
4	SORT AGGREGATE			
* 5	HASH JOIN			
6	BUFFER SORT			
7	PART JOIN FILTER CREATE	:BF0000		
8	PX RECEIVE			
9	PX SEND BROADCAST	:TQ10000		
* 10	TABLE ACCESS FULL	DATE_DIM		
11	PX BLOCK ITERATOR		:BF0000	:BF0000
* 12	TABLE ACCESS FULL	STORE_SALES	:BF0000	:BF0000

Final remarks

- Bloom filter are pushed down to storage cells and DBIM scans
- SQL Monitor report has details on Bloom filter
 - V\$sql_join_filter
- Hints px_join_filter() / no_px_join_filter()
- alter session set events 'trace[RDBMS.Bloom_filter] ...'
- Event 10128 Bloom pruning filter

Questions?



Email: toon.koppelaars@oracle.com

Twitter: @toonkoppelaars

The Oracle logo consists of the word "ORACLE" in white, uppercase, sans-serif font, centered within a solid red rectangular background.

ORACLE®

Integrated Cloud
Applications & Platform Services