
ORACLE®

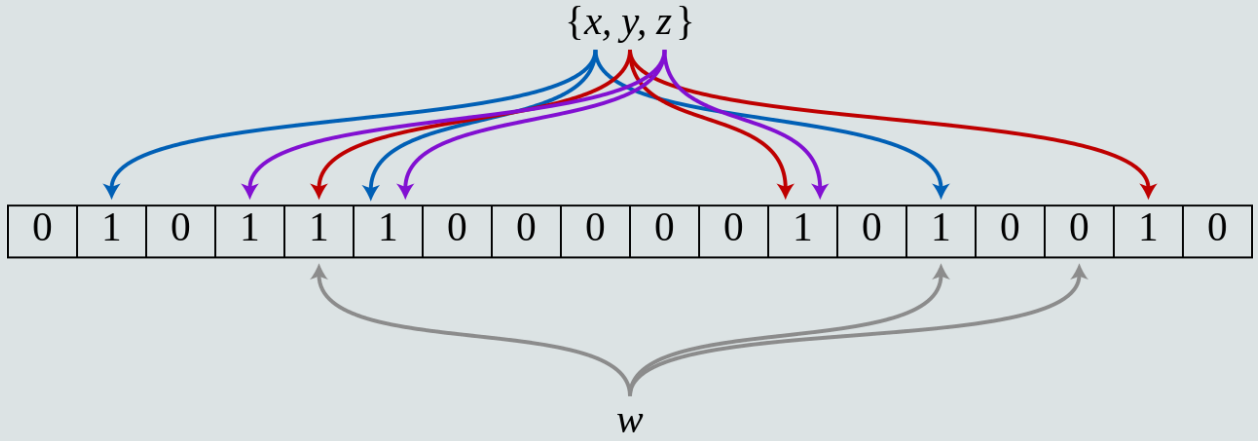
# Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

# Bloom filters

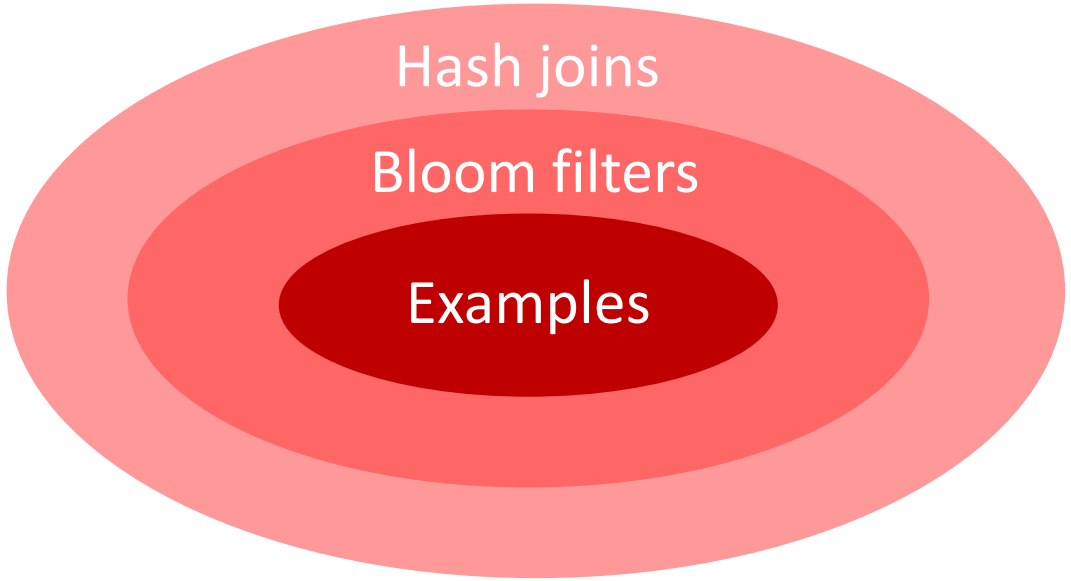
**Toon Koppelaars**

**Real World Performance team**
**Database Server Technologies**




ORACLE

# 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 created from dimension tables and applied to fact table during scan
  - To prevent unnecessary row-transport between parallel query slave sets

- As of 11.2.0.4 BF’s may also show up in serial plans

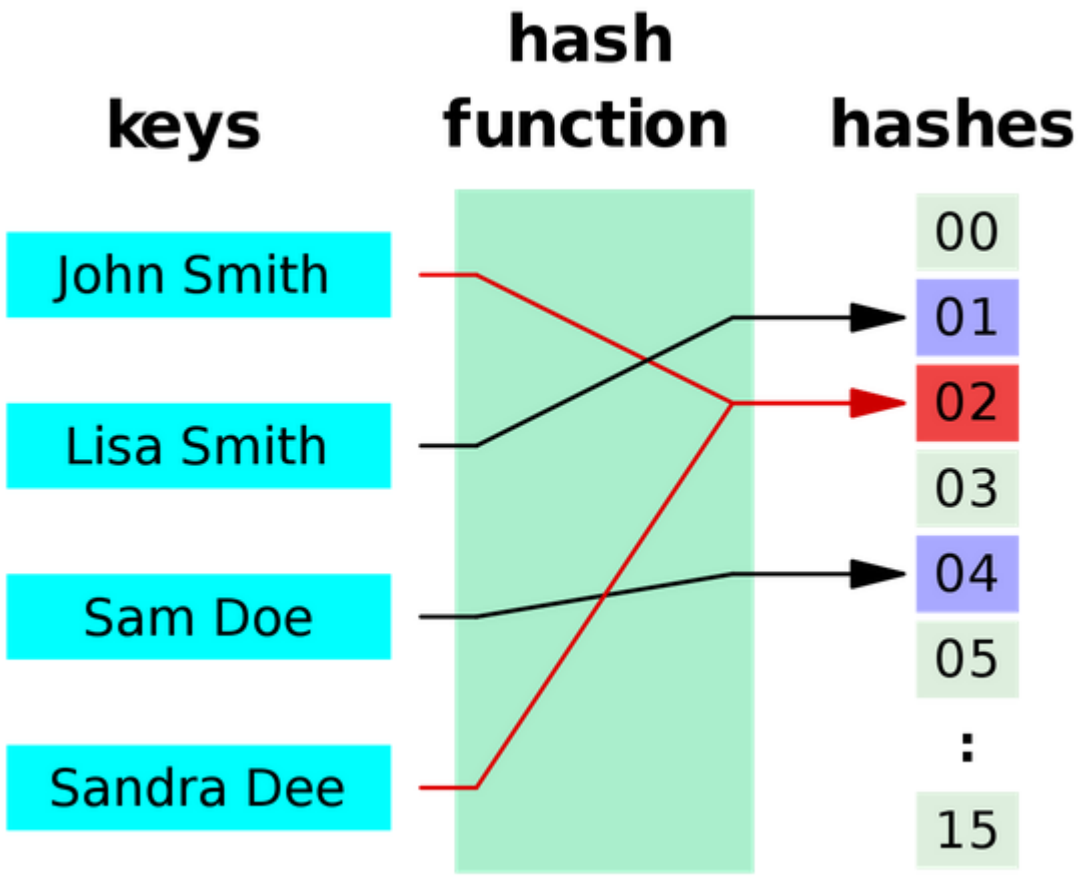
# Hash joins explained

```
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 takes arbitrary value as input domain
  - Digital data (byte string) of any length
- Hash function computes what's called a "hash value" for given input
  - Hash value = numerical offset in range of function
  - Range is typically [0..some-power-of-2 minus 1]
- Hash values are typically "evenly distributed" in range
  - Irrespective of distribution of input values
- Collisions can occur (#domain > #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 memory

# Phase 2: probe phase (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")
```

# Phase 2: probe phase (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")
```

scan

| 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 |

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

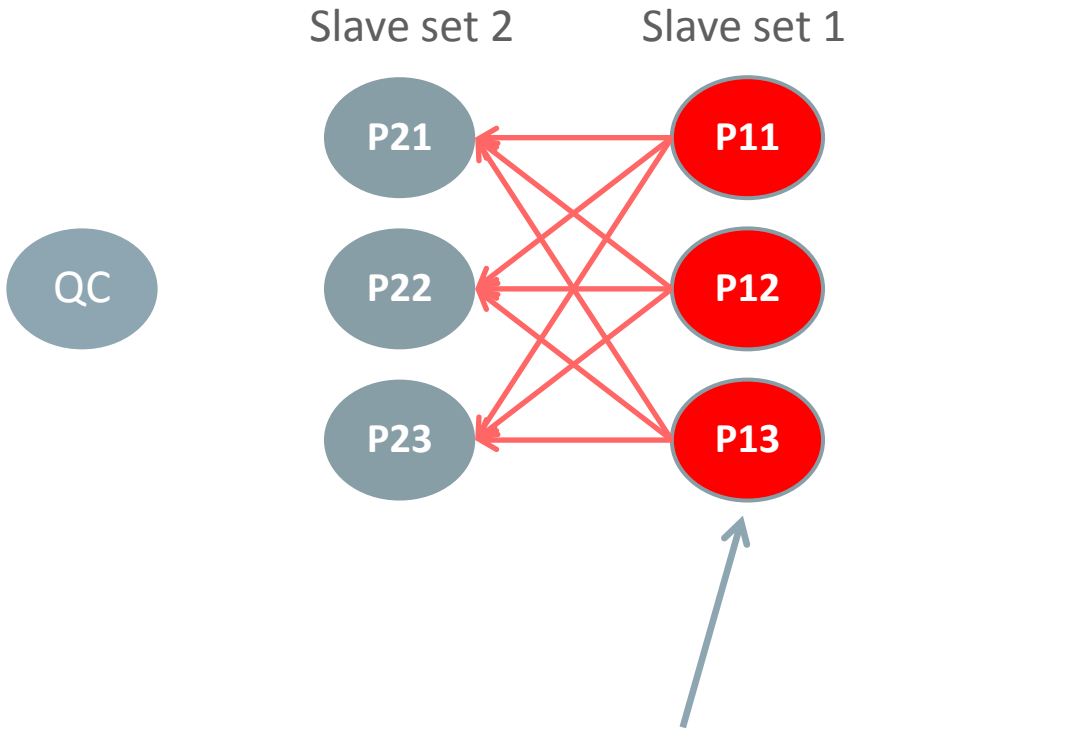
Probe phase

# Hash joins parallel execution plan

```
-------------------------------------------------------------------------------
| 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 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")
```

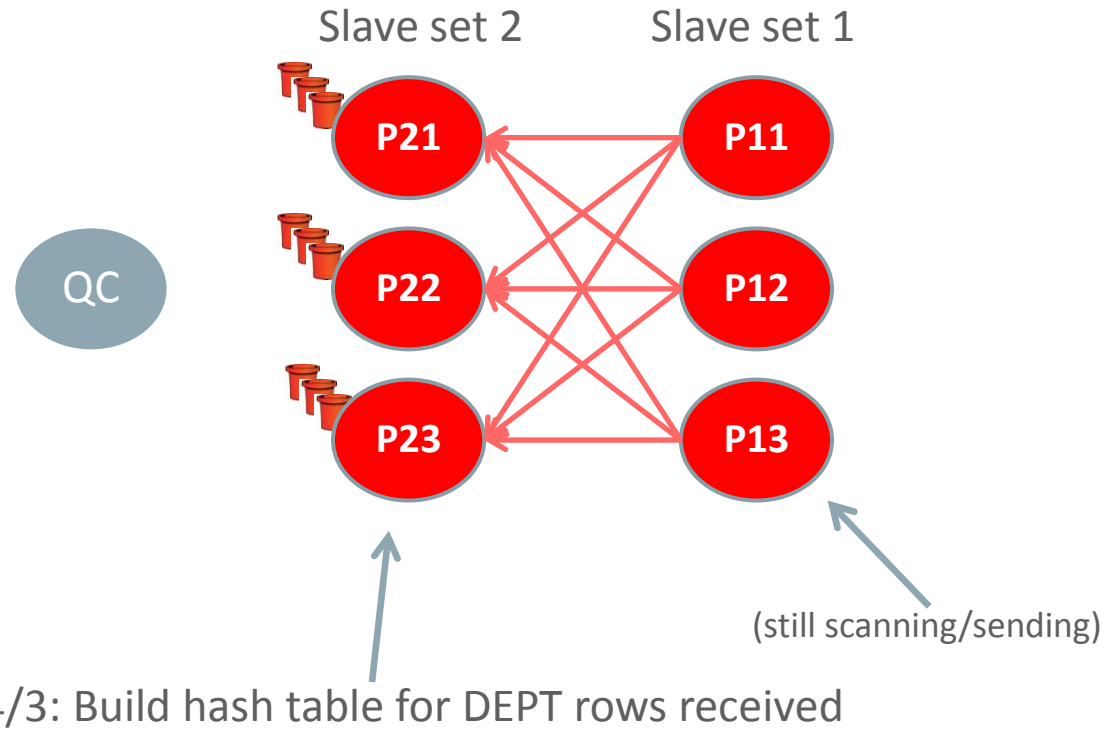



Lines 7,6,5: Scan DEPT, hash send to SS2

Hash PQ Distrib:
simply distributes rows based on
mod(hash(deptno),3)

# 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")
```



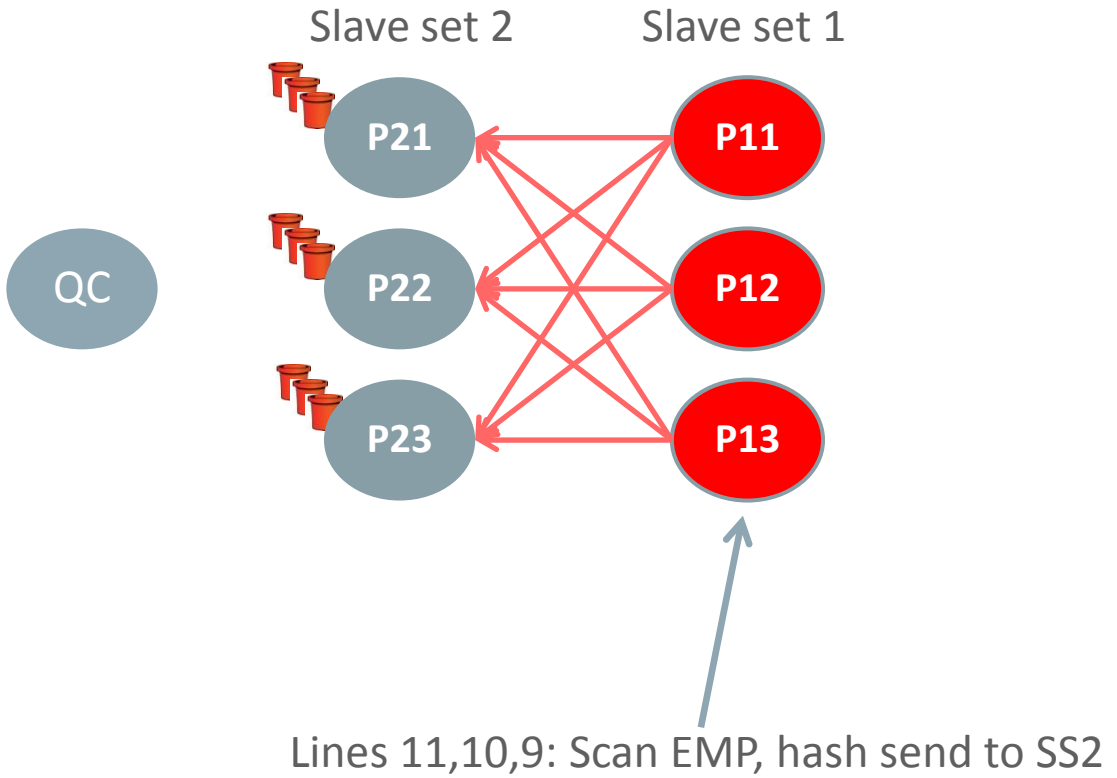
# Hash joins parallel execution plan

```
-------------------------------------------------------------------------------
| 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

# Hash joins parallel execution plan

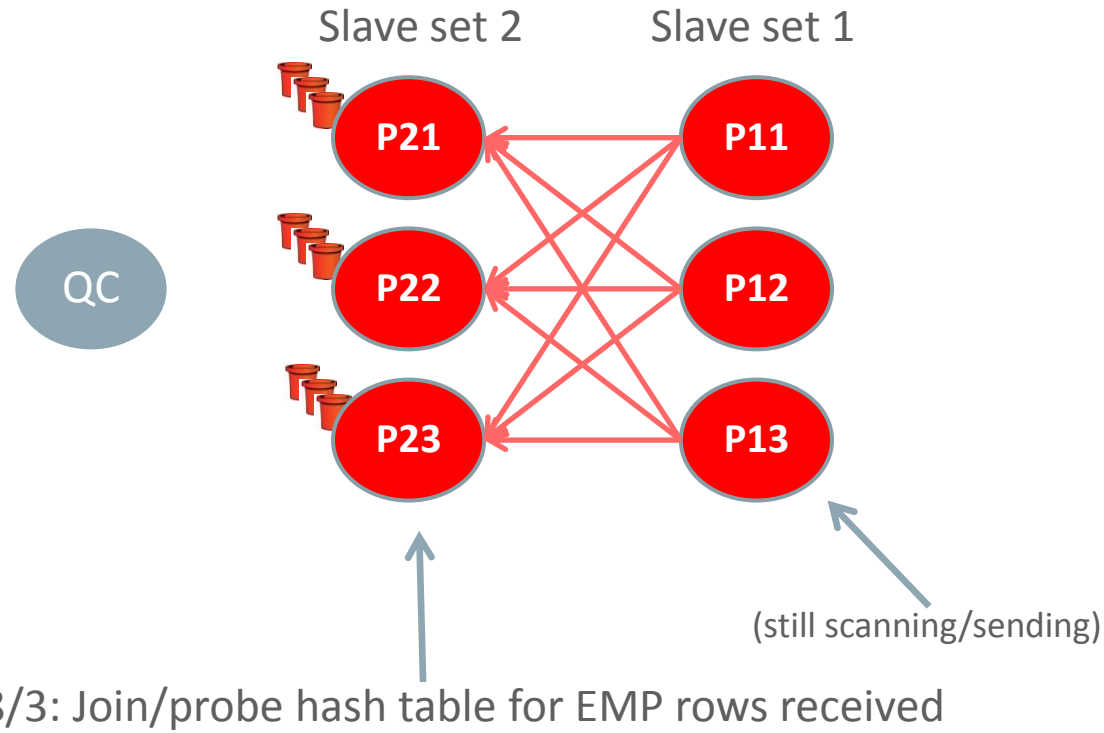
```
-------------------------------------------------------------------------------
| Id  | Operation                | Name     |    TQ  |IN-OUT| PQ Distrib |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |          |        |      |            |
|   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")
```




Same mod(hash(deptno),3) distribution used

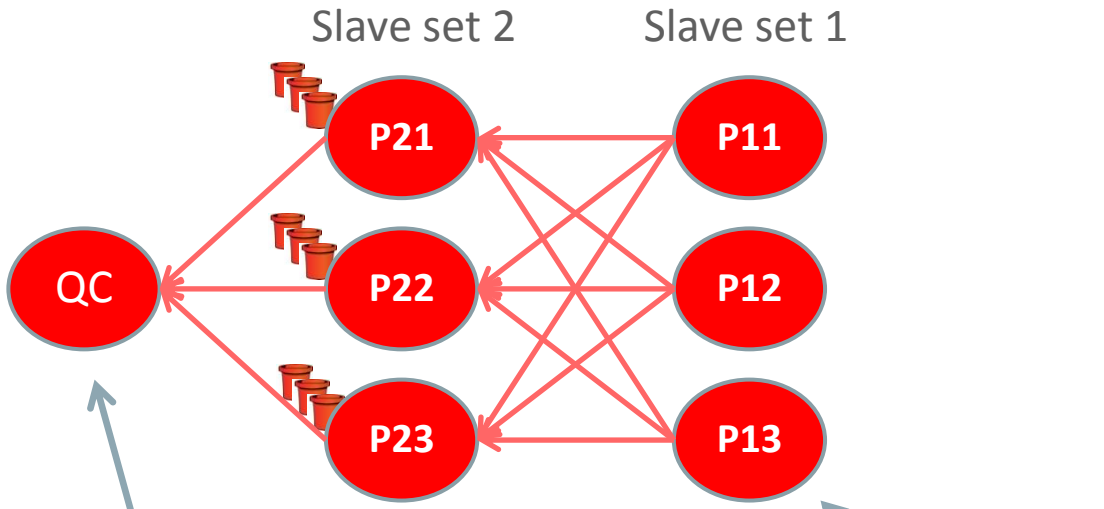
# Hash joins parallel execution plan

```
-------------------------------------------------------------------------------
| 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")
```



# 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")
```



# 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    | Starts | A-Rows |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |         |      1 |     84 |
|*  1 |  HASH JOIN                            |         |      1 |     84 |
|   2 |   TABLE ACCESS BY INDEX ROWID BATCHED | DEPT    |      1 |     21 |
|*  3 |    INDEX RANGE SCAN                   | DEPT_PK |      1 |     21 |
|*  4 |   TABLE ACCESS FULL                   | EMP     |      1 |     84 | <== 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))
```

# 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
  - Probably yes

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

Conceived by Burton Howard Bloom in 1970

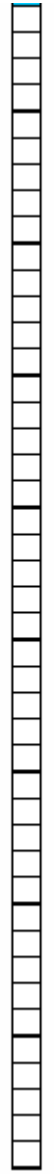
# Bloom filter concept

- BF(k,m)
  - k = number of hash functions used to encode value ($hf_1$,…,$hf_k$)
  - m = size of memory structure in bits (array of m bits)

  Hash-functions produce hash-value >=m, then moduled between 0..m-1

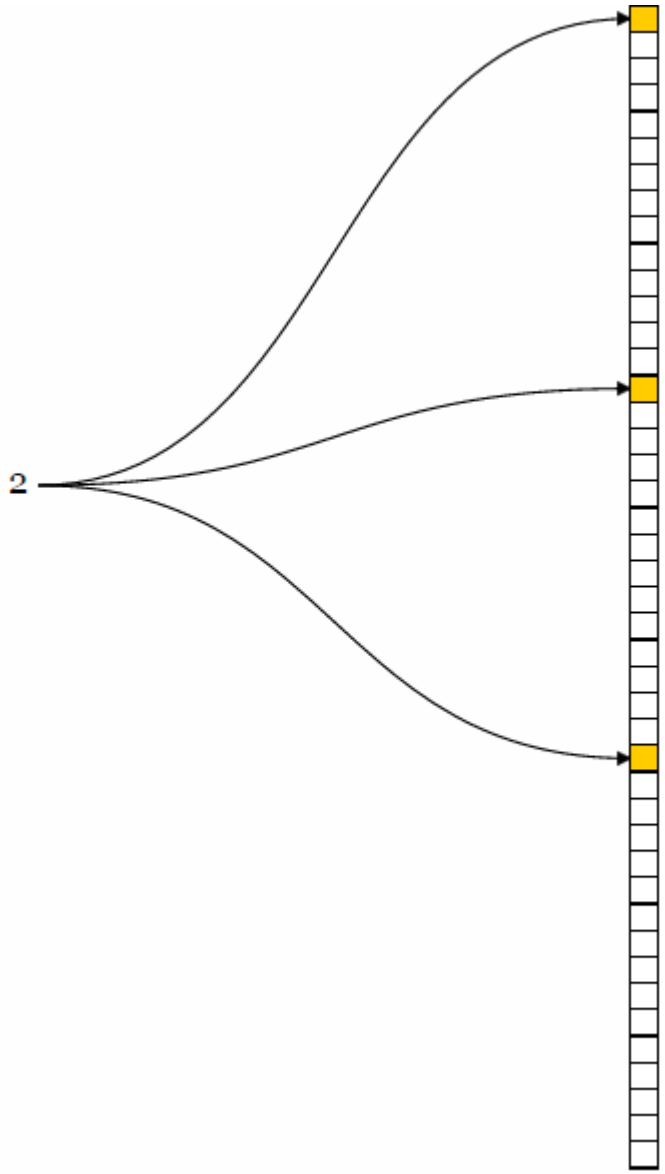
  Bit array initially filled with 0's

- 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

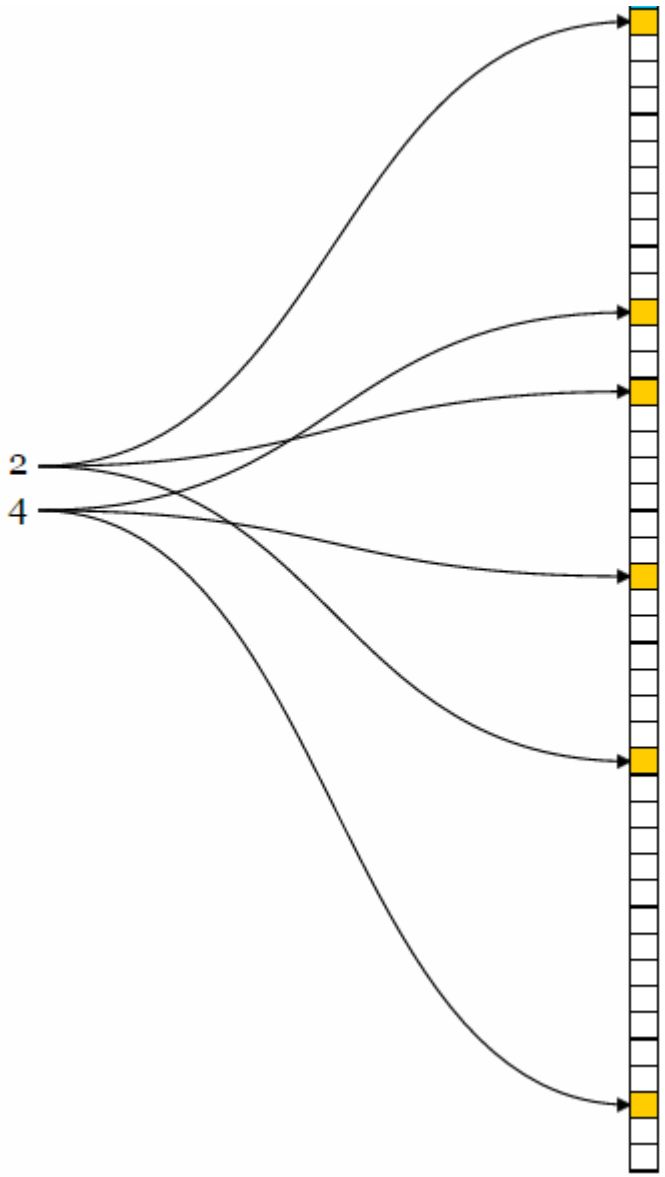


# Bloom filter: adding value 2

Assuming 3 hash functions used

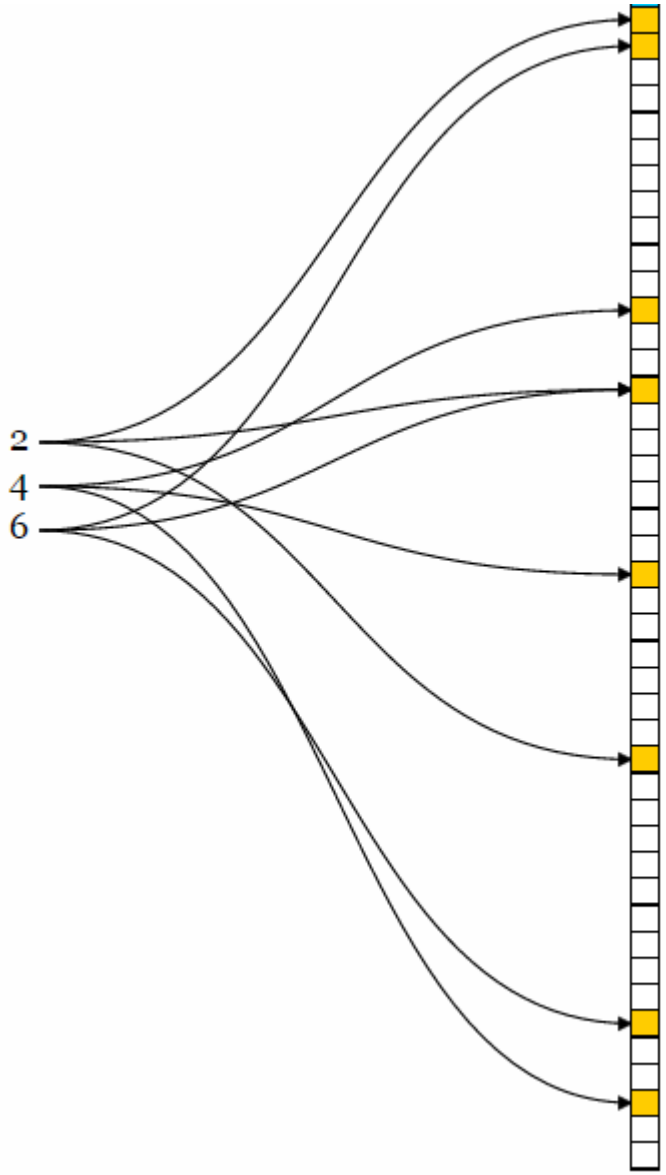


# Bloom filter: adding value 4



# Bloom filter: adding value 6

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

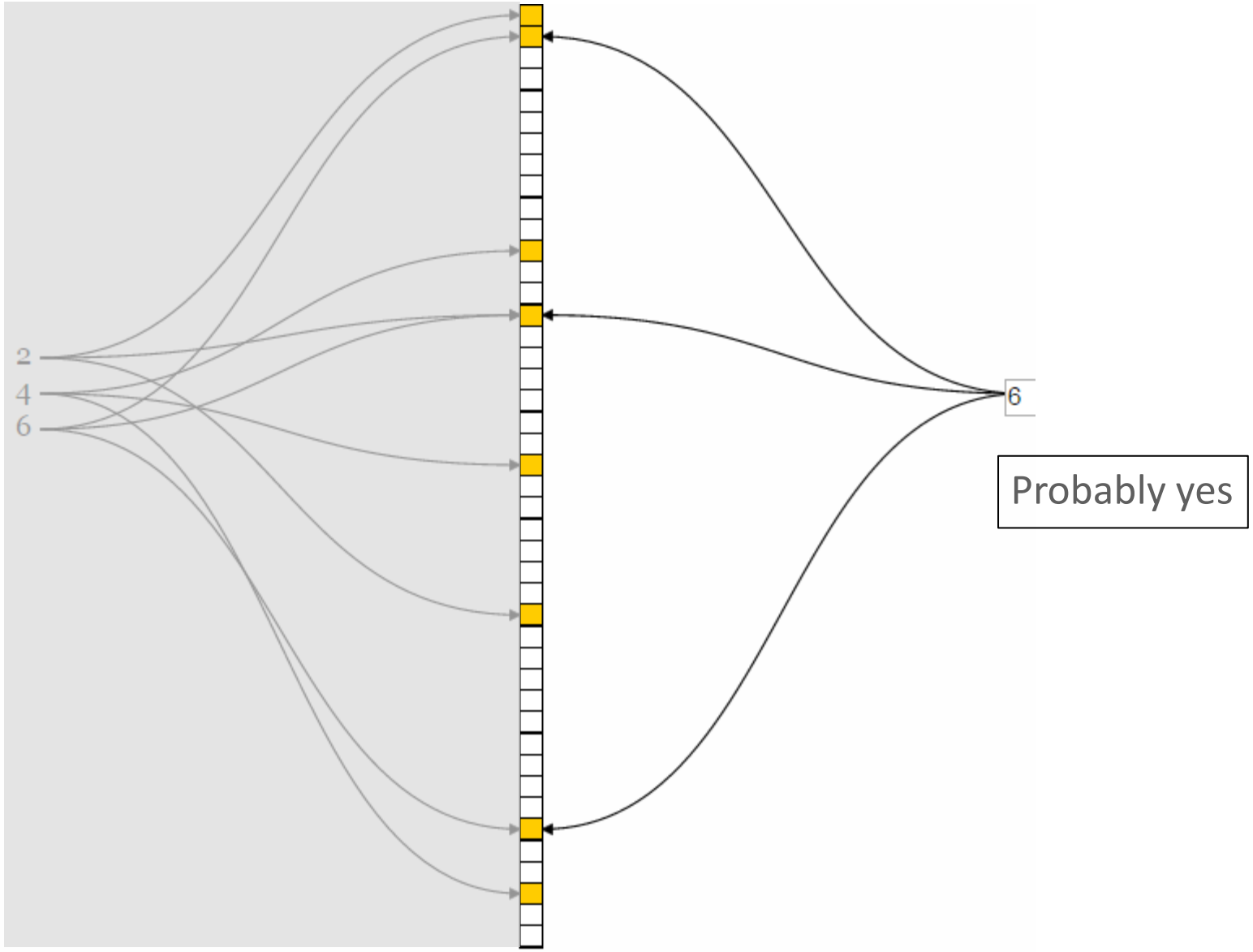


# Bloom filter: testing for membership

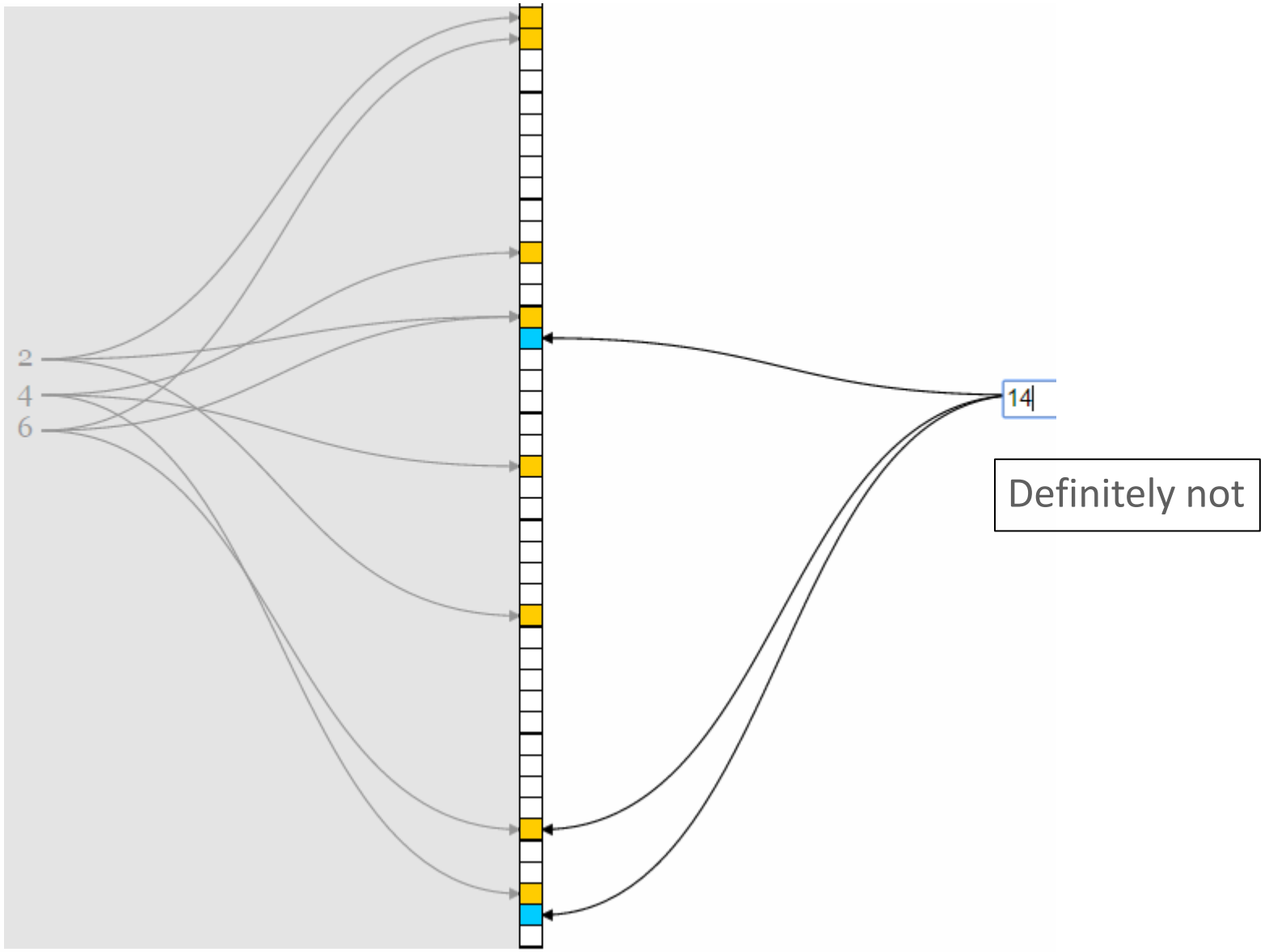
- Once bloom filter is populated with complete set, we can test for membership of any given value
  - By just applying the hash-functions to the value
  - And checking if all positions in bit-array are filled with 1
- If any not filled ➔ then *definitely not* a member of the set
- If all filled ➔ then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

- Two possible answers
  - Definitely not
  - Probably 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 (*)

- Demo: http://hur.st/bloomfilter?n=100000&p=0.01

(*) Not true anymore (folding/unfolding)

# 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 bit set/test operations
3. Enable early filtering (definitely not an element of set)
   - As we'll see shortly

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

# How can Bloom filter help filter early?

- BF use-cases in hash-join are of the form:
  - In BUILD stage of execution plan: BF CREATE
  - Then during PROBE scan stage of execution plan: BF USE

# Bloom filter use cases

- 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'
```

False positives caught in line 3

What is saved?
- Row distr. SS2 → SS1
- Bit vs byte comparisons

:BF0000 =
bit signature of join condition

```
-----------------------------------------
| Id  | Operation                | Name     |
-----------------------------------------
|   0 | SELECT STATEMENT         |          |
|   1 |  PX COORDINATOR          |          |
|   2 |   PX SEND QC (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"))
```

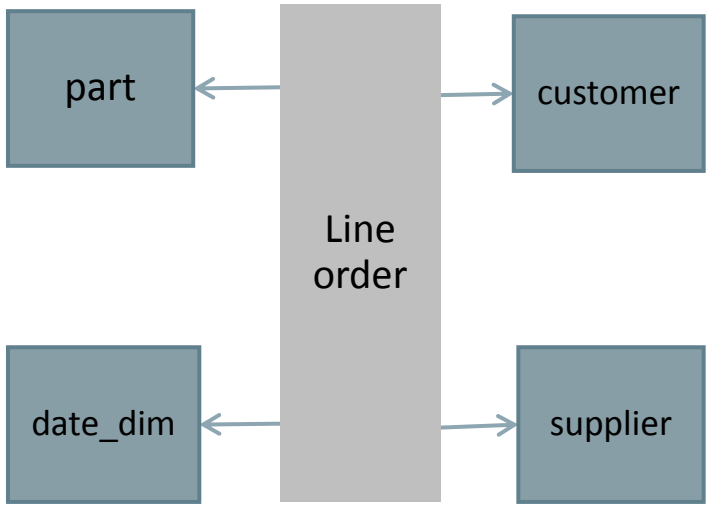
# Serial subquery filter

```
select d.dname,e.disnm
 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 GB

```
-----------------------------------------------------------------------
| 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"))
```

# Mutliple 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)

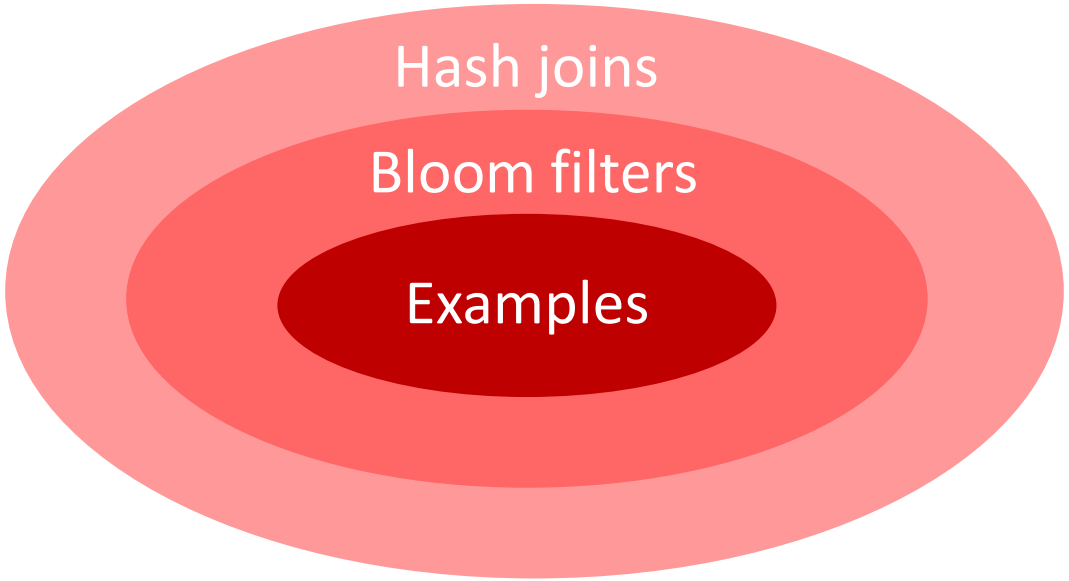
- Shows up as "*Partition* join filter create"
  - When joining from DIM to FACT and join-column is input for partitioning scheme of FACT, then
  - Partition function is applied to DIM column values to find P#'s of FACT that would hold such value
  - These P#'s are added to Bloom filter
  - Bloom filter is used when probing FACT
    - Only partitions that 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 can be pushed down to storage cells



# Questions?

Email: toon.koppelaars@oracle.com

Twitter: @toonkoppelaars

# Hardware and Software

ORACLE

# Engineered to Work Together