DELPHIX

Visual SQL Tuning (VST)

Kyle Hailey
kylehailey.com
kyle@delphix.com

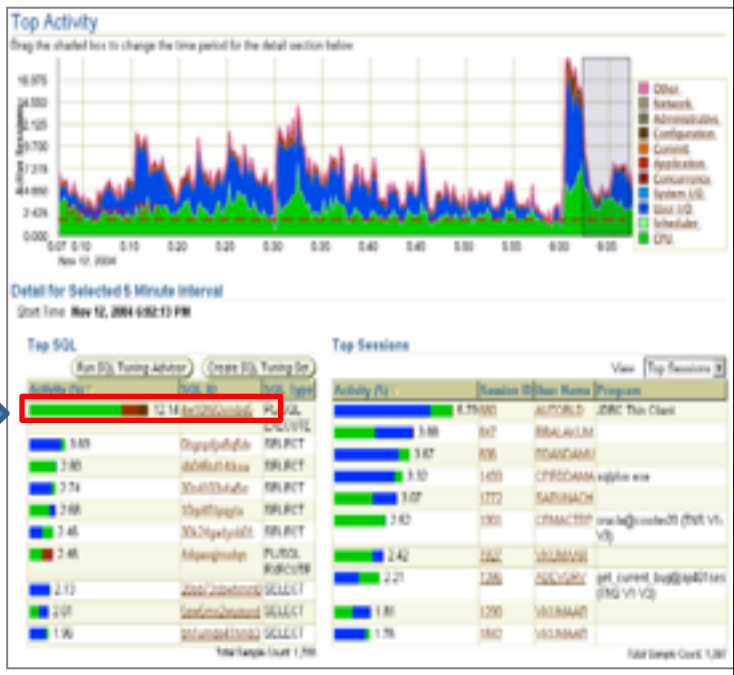
SQL Tuning

Methodology

1. Find: Problem SQL
2. Study: SQL Execution Plan
3. Fix: How ?

Step 1 : Find “bad” SQL

1. Users complain
2. High Resources
3. Monitoring



Step 2: Get Explain Plan

Trace fie

```
PARSING IN CURSOR #2 len=53 dep=0 uid=61 oct=3
lid=61 tim=1151519905950403 hv=2296704914
ad='4e50010c'
SELECT 'Hello, world; today is ' || SYSDATE FROM dual
END OF STMT
PARSE
#2:c=4000,e=1540,p=0,cr=0,cu=0,mis=1,r=0,dep=0,og=1 ti
m=1151519905950397
BINDS #2:
EXEC
#2:c=0,e=58,p=0,cr=0,cu=0,mis=0,r=0
51519906034782
WAIT #2: nam='SQL*Net m
id=1650815232 #bytes
tim=115151990603
FETCH
#2:c=0
                                                    215
                                    0,r=0,dep=0,og=0,tim=115
 Net message to client' ela= 1 driver
 #bytes=1 p3=0 obj#=-1
19906035188
 #2: nam='SQL*Net message from client' ela= 192
driver id=1650815232 #bytes=1 p3=0 obj#=-1
tim=1151519906035400
STAT #2 id=1 cnt=1 pid=0 pos=1 obj=0 op='FAST DUAL
(cr=0 pr=0 pw=0 time=3 us)
```

```
-------------------------------------------------------------------------------
          | Name                          | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------
          |                               |      1 |      1 |      1 |
          |                               |      1 |        |   1909 |
 ROWID    |                               |      1 |      1 |   3413 |
          |                               |      1 |    165 |   6827 |
          |                               |      1 |    165 |   3413 |
          |                               |      1 |    165 |   3624 |
          |                               |      1 |    242 |   2895 |
          |                               |      1 |    233 |   2897 |
          |                               |      1 |      1 |      1 |
          |                               |      1 |      1 |      1 |
          |                               |      1 |    286 |   2895 |
          |                 OPAY_EARNS    |      1 |  27456 |    122K|
          |                 ETROPAY_RQST  |      1 |  13679 |  13679 |
          |                 PS#RETROPAYPGM_TBL |   3413 |      1 |   3413 |
          |                               |   1791 |      1 |   1791 |
          |                               |   1791 |      1 |   1579 |
 /MAX)    | WB_JOB_F                      |   1791 |      1 |   1579 |
          |                               |   1539 |      1 |   1539 |
          |                               |   1539 |      1 |   1539 |
 (MIN/MAX)| WB_JOB_G                      |   1539 |      1 |   1539 |
-------------------------------------------------------------------------------


d by operation id):
-------------------

"B"."EFFSEQ"=))
"."PAY_OFF_CYCLE_CAL")
NO"="C"."RETROPAY_SEQ_NO")
EMPLID" AND "C"."EMPL_RCD#"="B"."EMPL_RCD#")
 AND "A"."PAY_CONFIRM_RUN"='N')
"COMPANY" AND "B"."PAYGROUP"="A"."PAYGROUP")
S_FLAG"='C' AND "C"."RETROPAY_LOAD_SW"='Y'))
ID"="D"."RETROPAY_PGM_ID")
ND "F"."EMPL_RCD#"=:B2 AND "F"."EFFDT"<=:B3)
ND "G"."EMPL_RCD#"=:B2 AND "G"."EFFDT"=:B3)
```

Complexity !

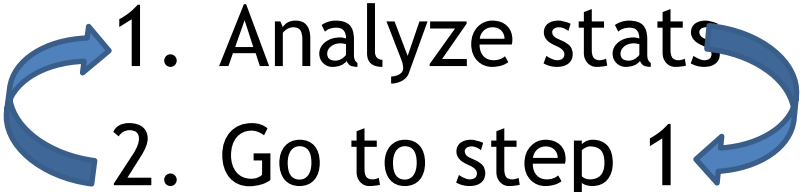
Step 3: ???

Methodology

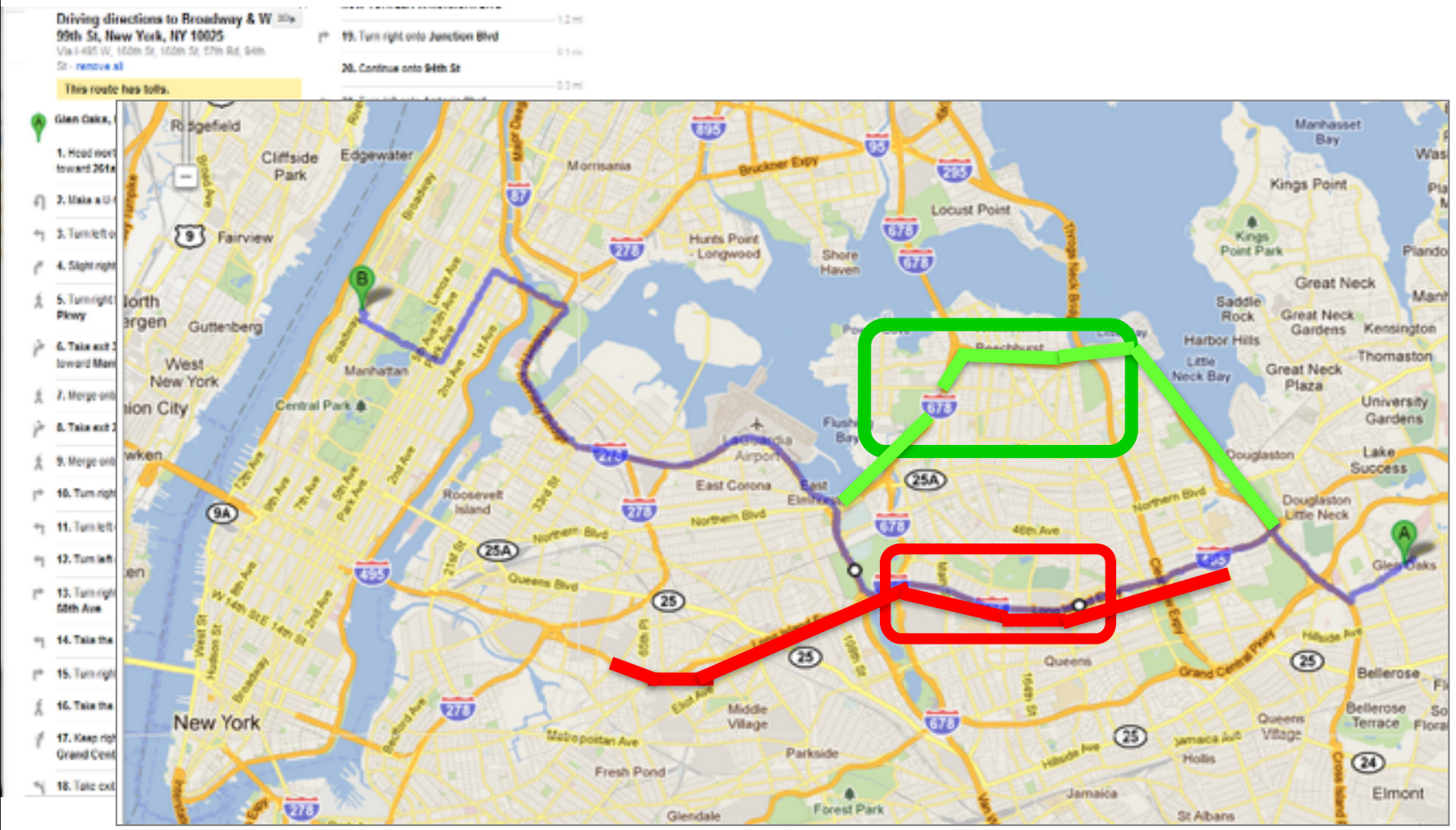
- Identify Slow Queries ✓
- Look at Execution Plar ✓
- Fix WTF?? ✗

Fix

1. Analyze stats
2. Go to step 1



Explain Plan: Like Directions



Where is the Map?

SQL Tuning

1. Two Table Join
2. Multi-Table Join
3. Create Map (VST diagram)
4. Apply Methodology (how tune)
5. Examples

DELPHIX

How to Join two tables?

1. Which table to start with (main step)
2. What Indexes to use
3. What type of join to use HJ, NL

Design options:

- Partitions
- IOT
- Bitmap indexes
- Hash cluster
- Materialized views
- etc

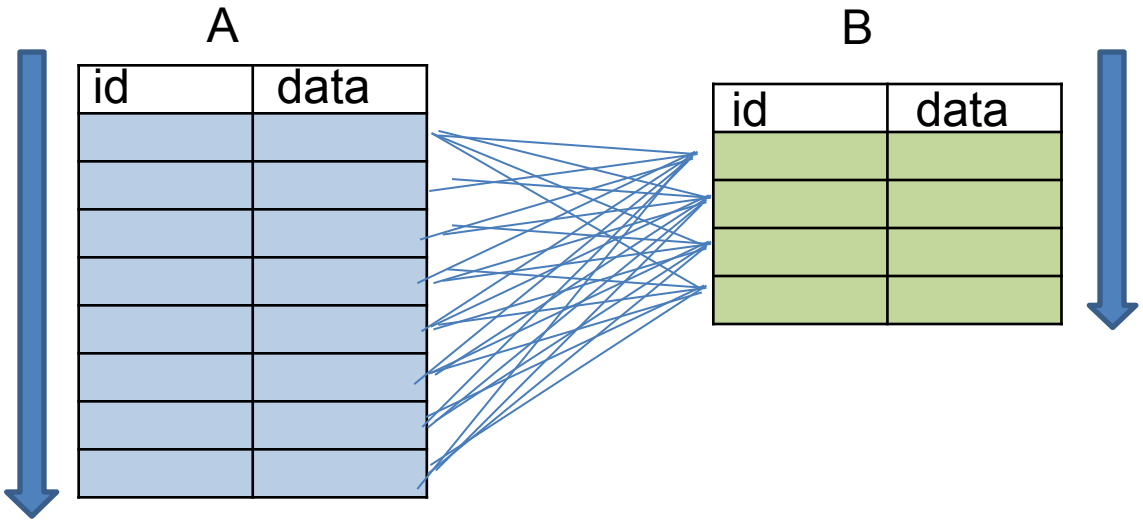
Two Table Join

1. Do Indexes exist
2. What Type of Relation between tables
   - One to One
   - One to Many
   - Many to Many
3. Special Cases
   - Outer Joins
   - Not Exists

Table Join Order

```
select    *
from      a, b
where
          a.id = b.id
```

If No Index then (NL) order doesn't matter



Every row visits Every row
Work= A-rows x B-rows (8 x 4)
HJ could optimizes => simulate index lookup

2 Table join, with indexes

```
select    *
from      a, b
where
          a.id = b.id
```

Indexes on

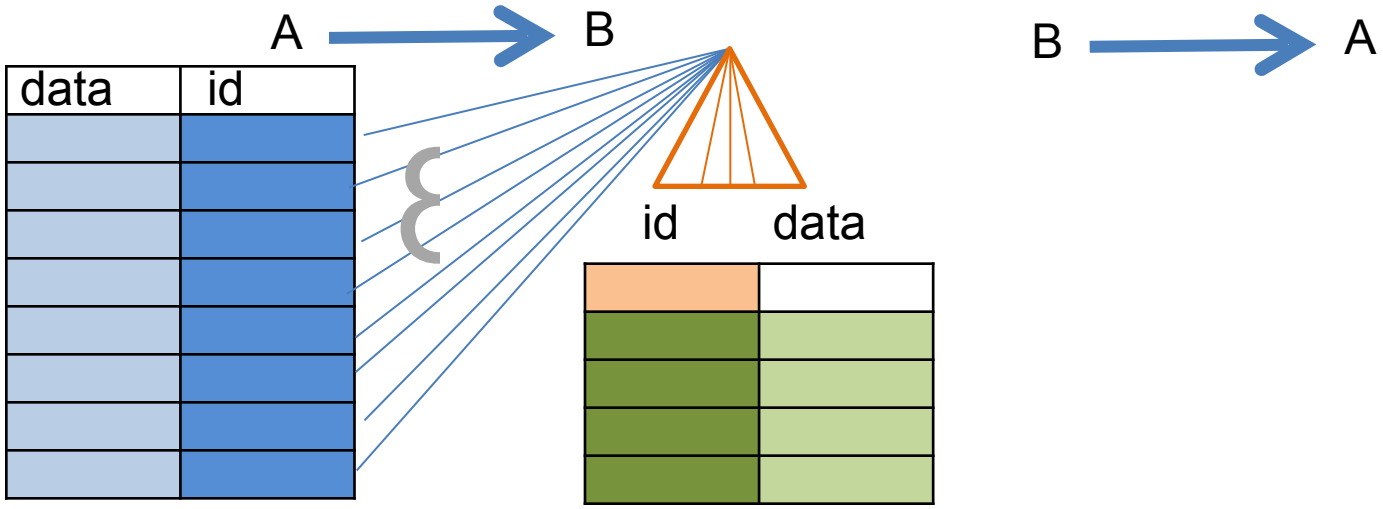
- a.id
- b.id

2 Table join, with indexes

```
select    *
from      a, b
where
          a.id = b.id
```

Indexes on

- a.id
- b.id

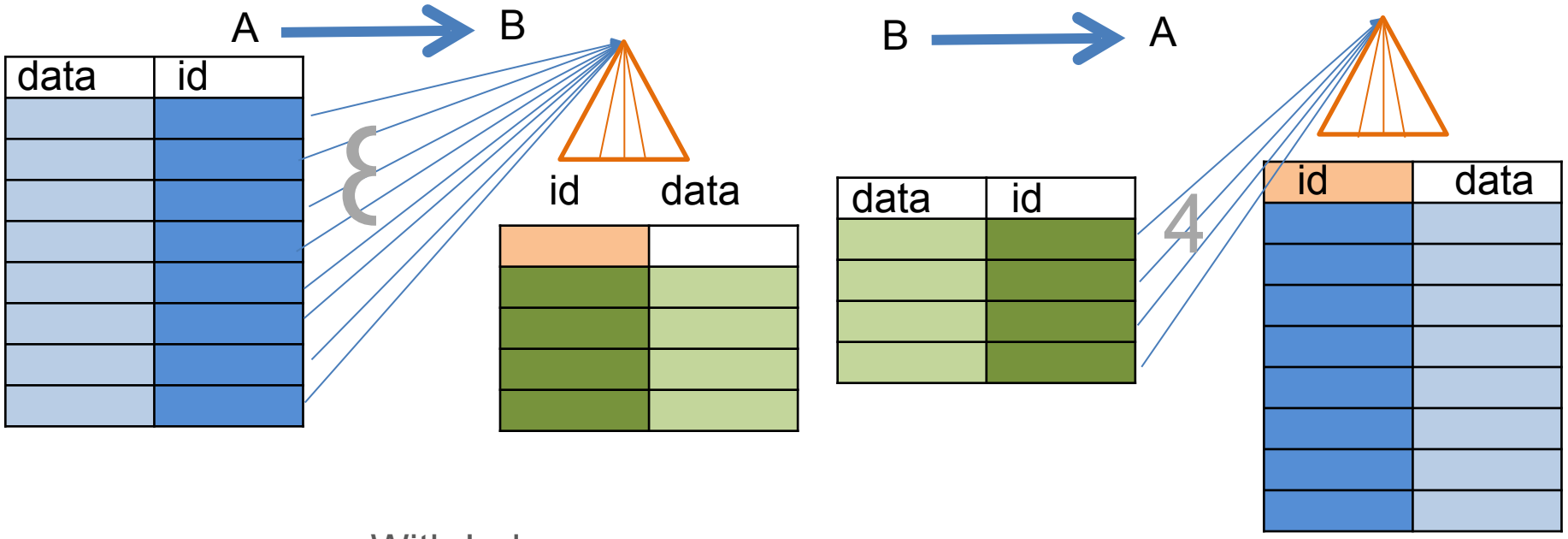


2 Table join, with indexes

```
select    *
from      a, b
where
          a.id = b.id
```

Indexes on

- a.id
- b.id

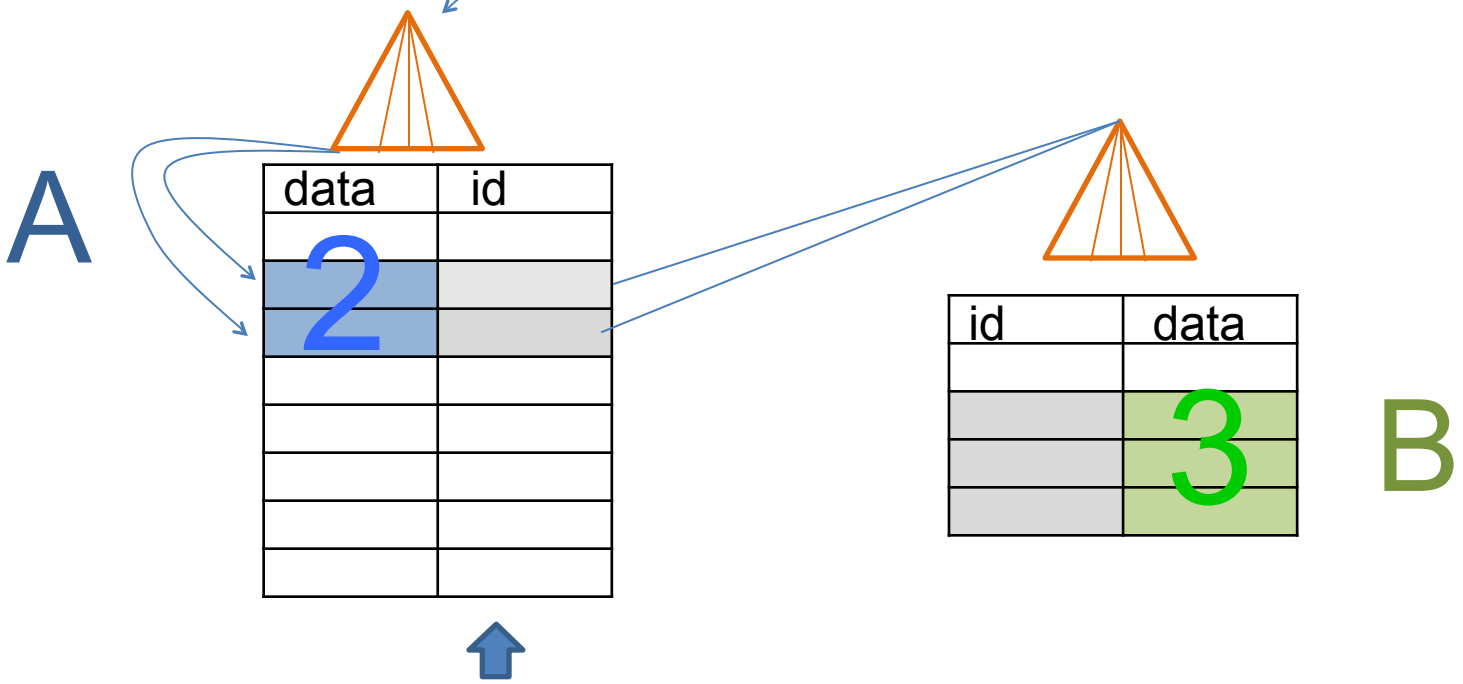


With Indexes
Start with B

Start with the least rows

2 table join - filters

```
select    *
from      a, b
where
          a.id = b.id          -- join
and       a.field = 'val a'    -- filter a
and       b.field = 'val b'    -- filter b
```



Start on table with least rows after filter

Two Table Summary

1. If no indexes & no filters

   then order doesn't matter with NL

   - Hash join will simulate index, start by hashing smaller table

2. If indexes

   then start with smaller table

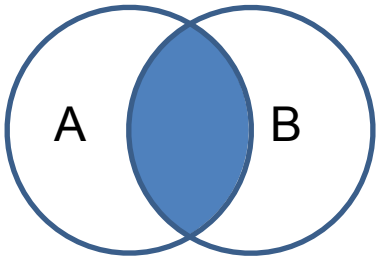
3. If filters and indexes

   then start with most filtered table

Creating Two table Join Map

- Set diagrams
- Short comings of set diagrams
- New map diagrams

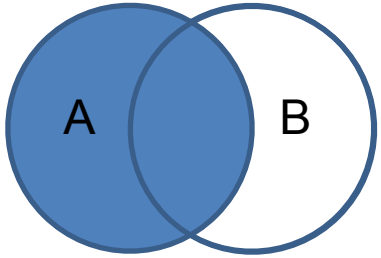
Set Diagrams: Two Table Joins – looks simple



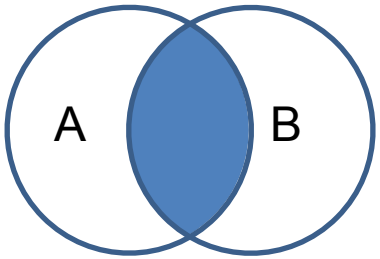
Inner Join

(blue is data returned)

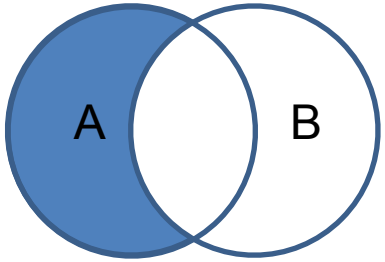
Two Table Joins



Left join B B(+)

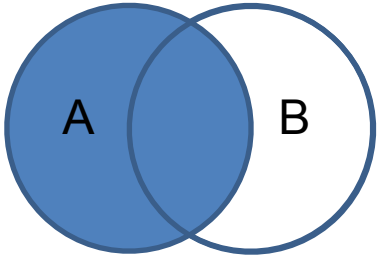


Inner Join

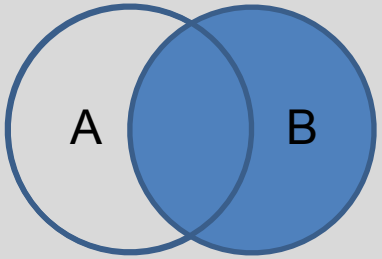


Not exists B

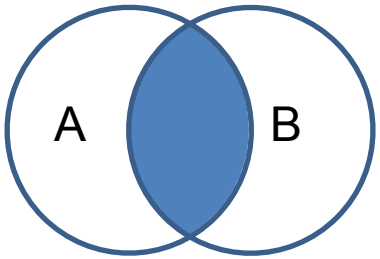
Two Table Joins



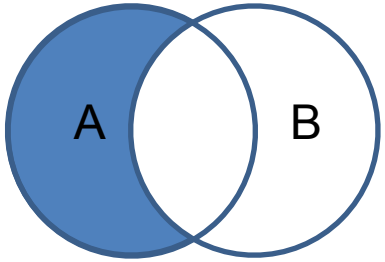
Left join B B(+)



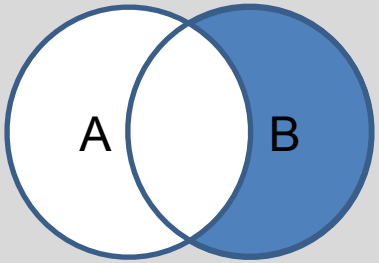
Right join B A(+)



Inner Join

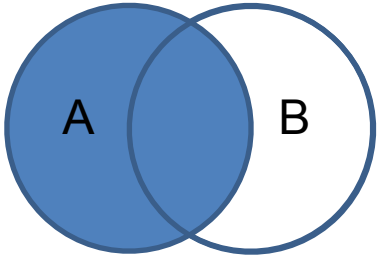


Not exists B

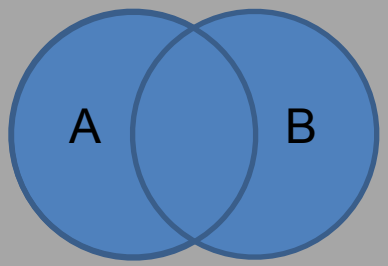


Not Exists A

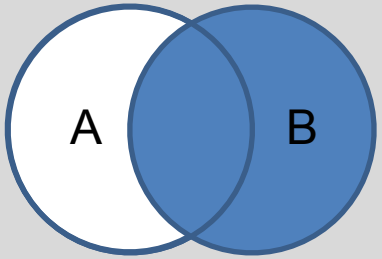
Two Table Joins (blue is data returned)



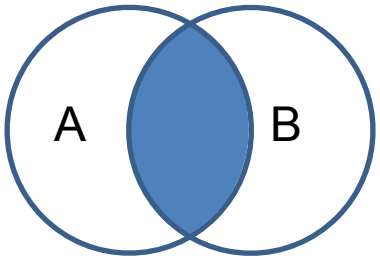
Left join B B(+)



Full outer (union)

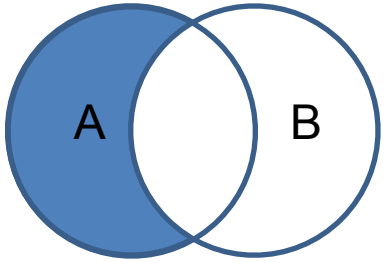


Right join B A(+)

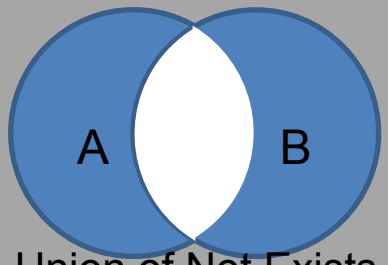


Inner Join

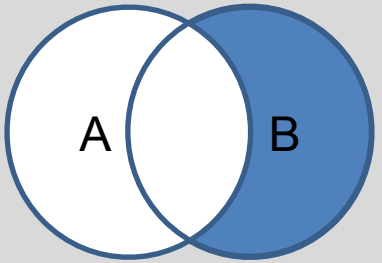
Doesn't tell the whole story
Missing « amplifications »



Not exists B

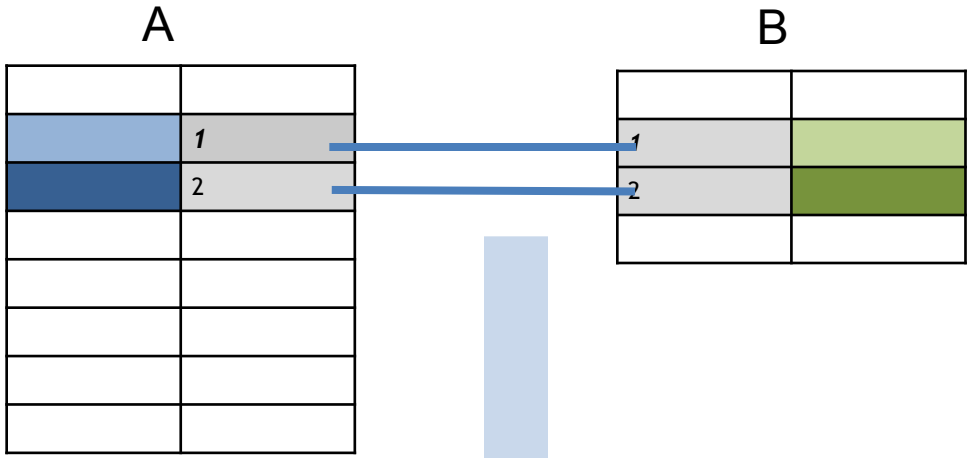


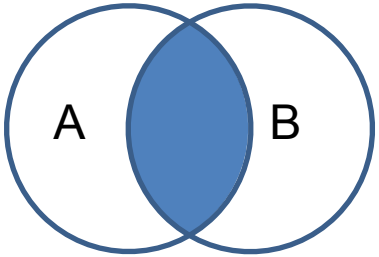
Union of Not Exists

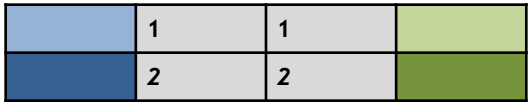


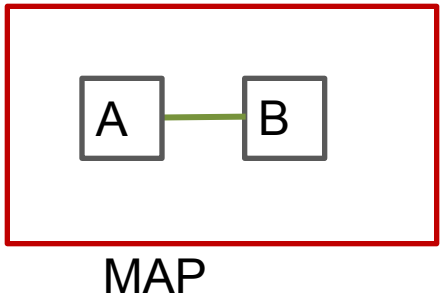
Not Exists A

One to One - intersection

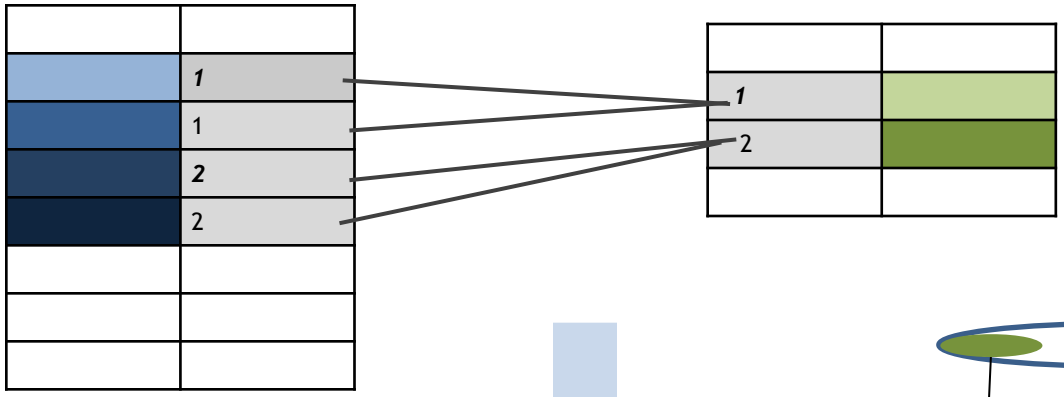


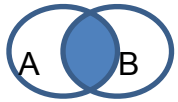


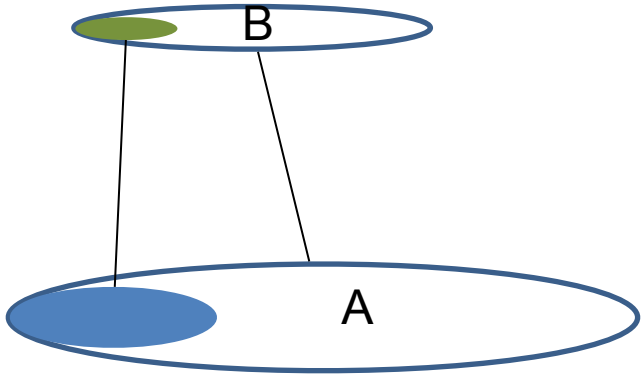


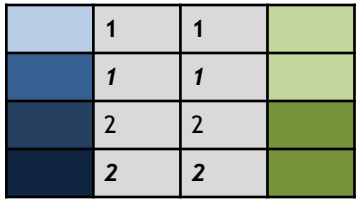


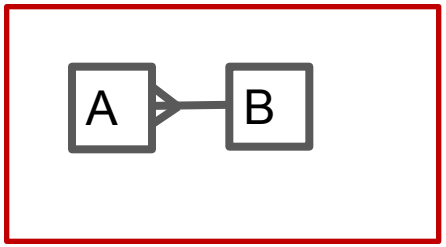
One to Many - projection





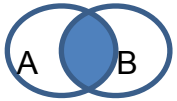


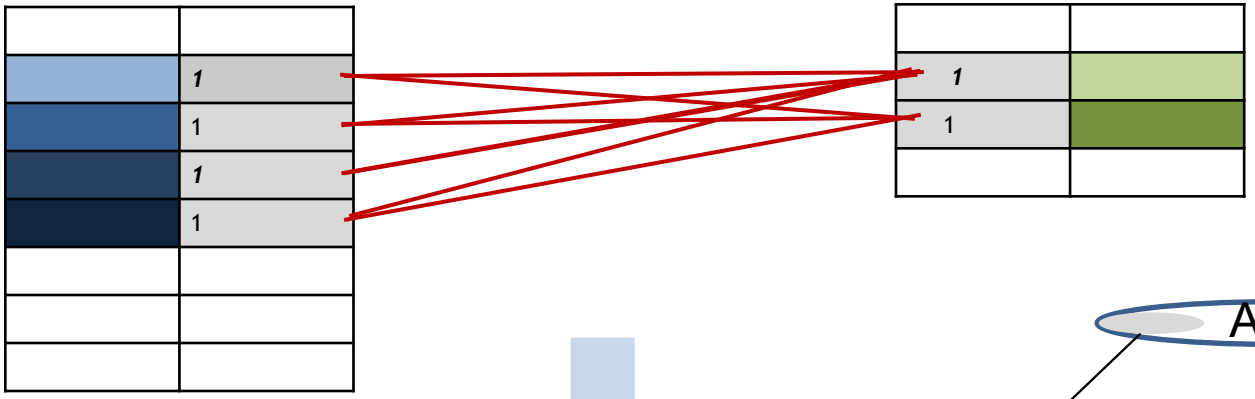


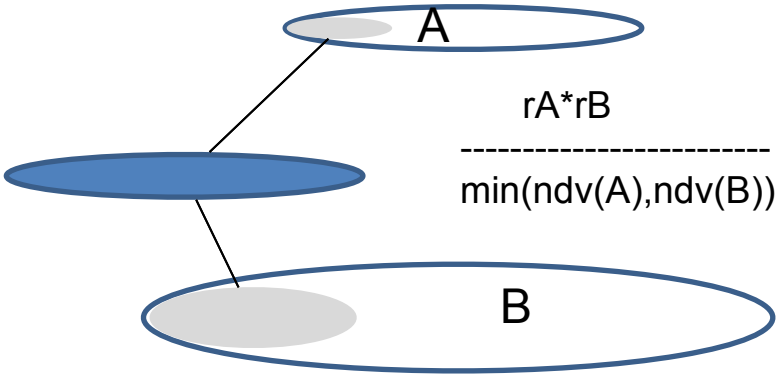


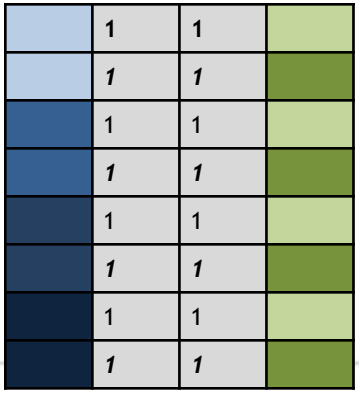
MAP

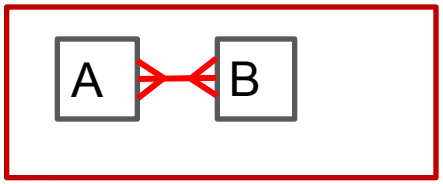
Many to Many - amplification





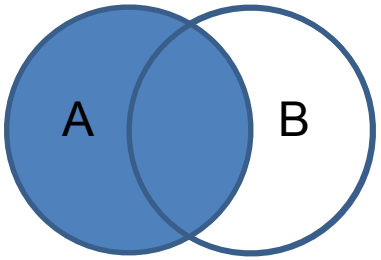






MAP

Two Table Joins



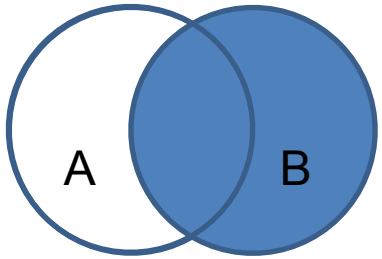
Left join B B(+)

english

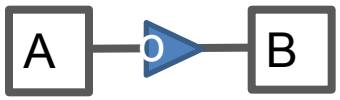
ENGLISH_TEXT	ORDINAL_ID
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

french

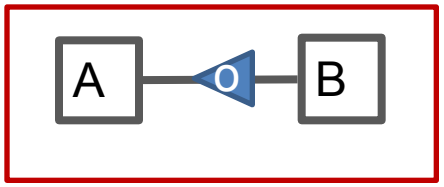
ORDINAL_ID	FRENCH_TEXT
1	Un
3	Trois
4	Quatre
5	Cinq
6	Six
7	Sept
8	Huit



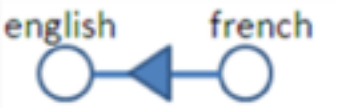
Right join B A(+)



MAP

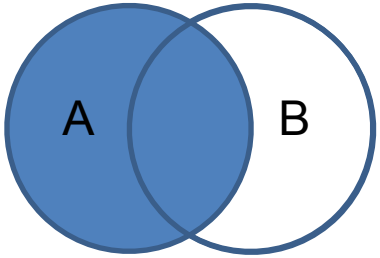


MAP

type	ANSI	ANSI 89 (Oracle)	type	type
inner join	english INNER JOIN french using (ordinal_id)	english e, french f where e.ordinal_id=f.ordinal_id		
left outer join	english LEFT JOIN french using (ordinal_id)	english e, french f where e.ordinal_id=f.ordinal_id(+)		
right outer join	english RIGHT JOIN french using (ordinal_id)	english e, french f where e.ordinal_id(+)=f.ordinal_id		

Outer Join – Attention when no columns selected

Color nodes that return no data 



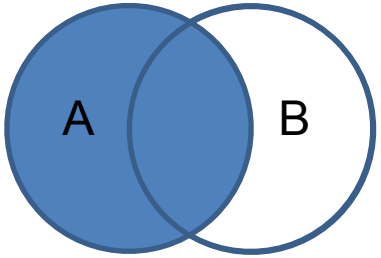
Left join B B(+)

no affect


```
Select a.* from a, b where b.field(+) = a.field;
```

Outer Join – Attention when no columns selected

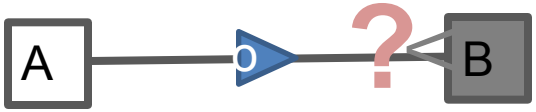
Color nodes that return no data 



Left join B B(+)

no affect


```
Select a.* from a, b where b.field(+) = a.field;
```

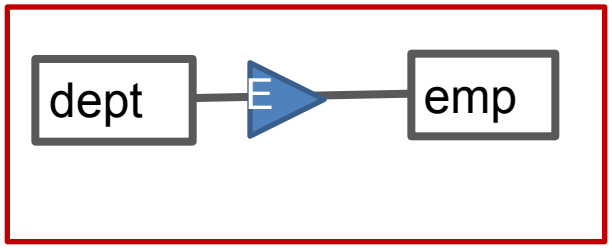


Multiplier

no affect
or
multiplies rows

Exists (in)

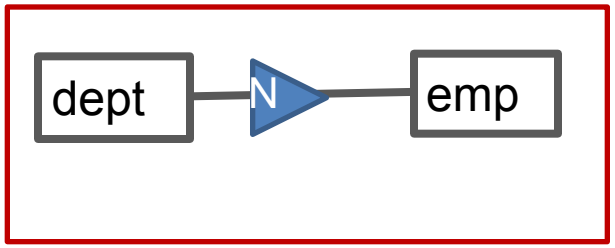
```
SELECT d.*
FROM dept d  WHERE exists
        ( SELECT null
          FROM emp e
          WHERE e.deptno=d.deptno);

SELECT d.*
FROM dept d  WHERE d.deptno in
        ( SELECT  deptno
          FROM emp e  );
```



MAP

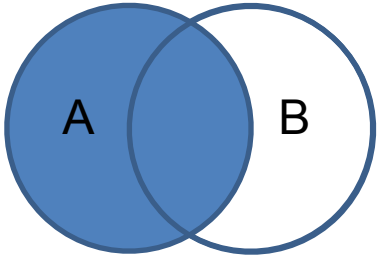
Not Exists (not in)

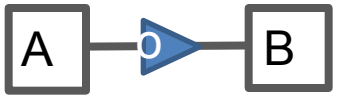
```
SELECT d.*
FROM dept d  WHERE not exists
          ( SELECT null
            FROM emp e
            WHERE e.deptno=d.deptno);

SELECT d.*
FROM dept d  WHERE d.deptno not in
         ( SELECT deptno
           FROM emp e
           where deptno is not null );
```



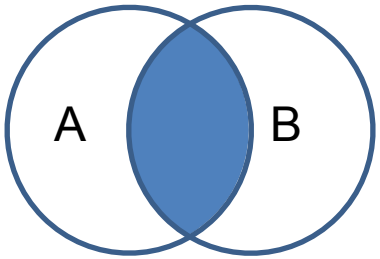
MAP

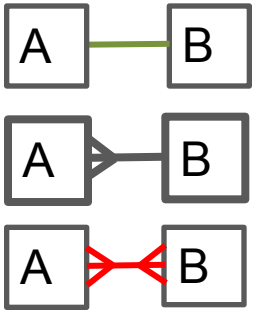
Two Table Joins : Maps



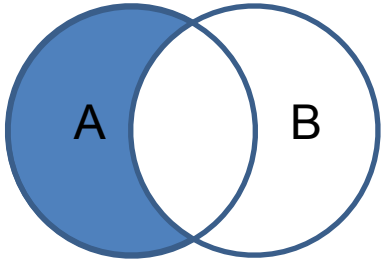


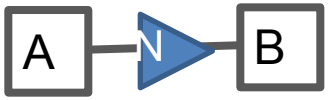
Left join B B(+)





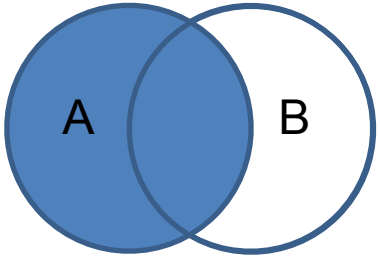
Inner Join



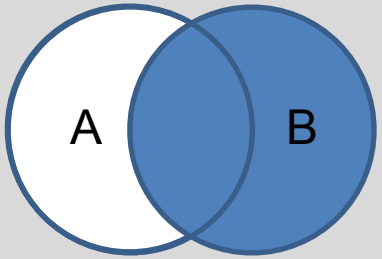


Not exists B

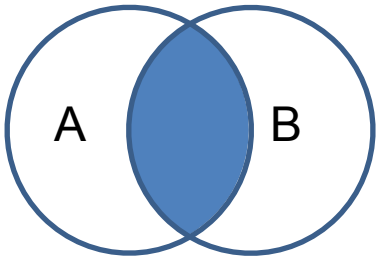
Two Table Joins : maps



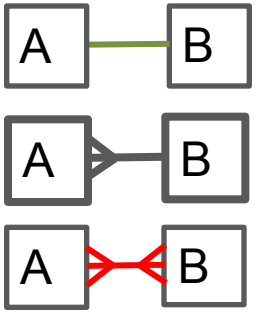
Left join B B(+)

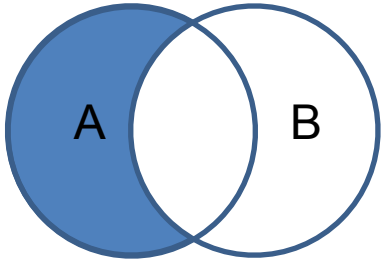


Right join B A(+)

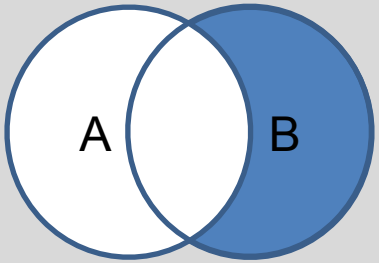


Inner Join





Not exists B



Not Exists A

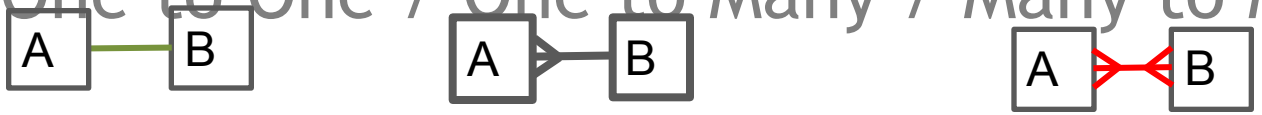
Two Table Join

1. Where to start

1. Start at table with the least rows after predicate filtering

2. Drawing Map:

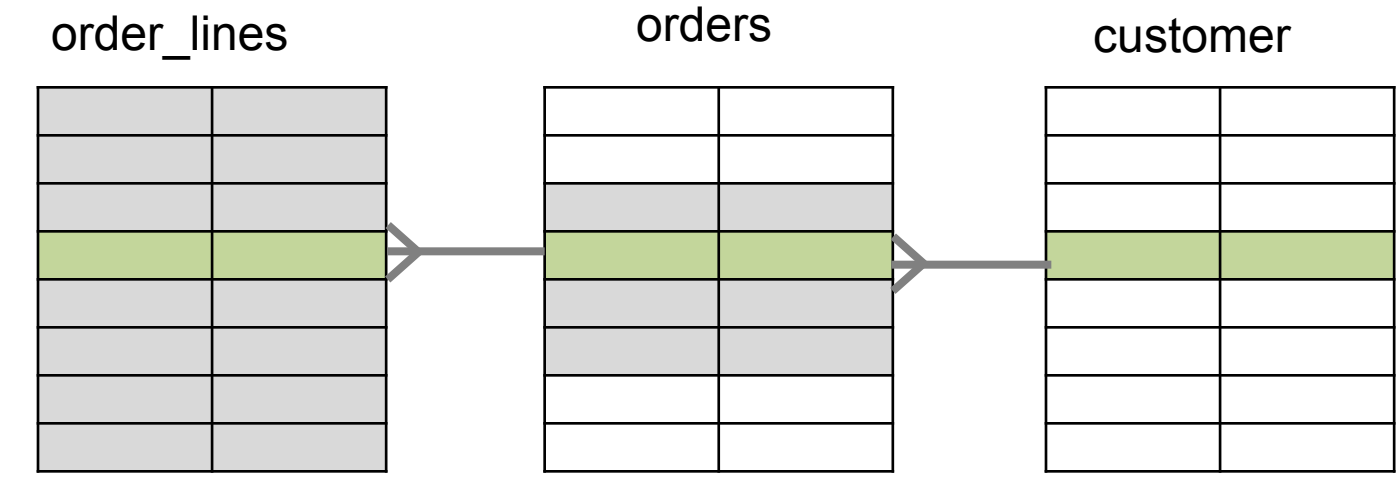
- One to One / One to Many / Many to Many



- Outer Joins / Not Exists

Now , three table joins:

Three table join



Join direction

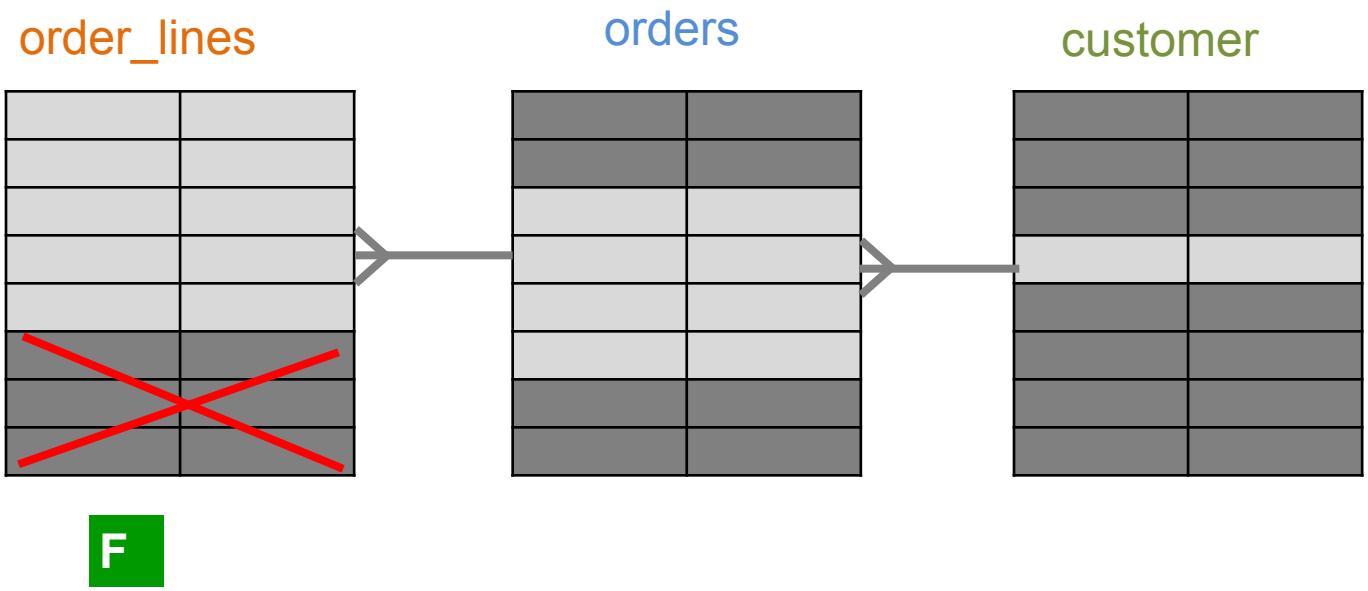
8 rows ← 4 rows ← 1 row

Ex) 1 customer 4 orders, each order has 2 order_lines each

Join direction

1 row → 1 row → 1 row

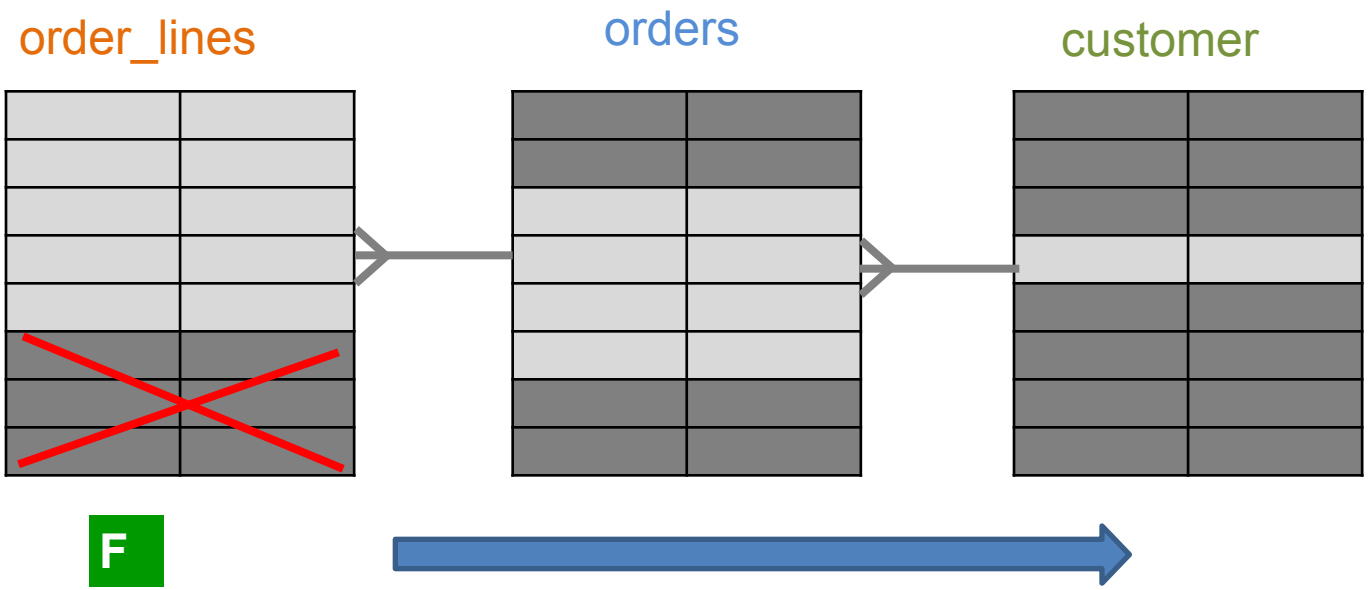
Three table join



Where
order_lines.field = value

A Filter on order_lines is going to eliminate work on orders and customers, if we start at order_lines

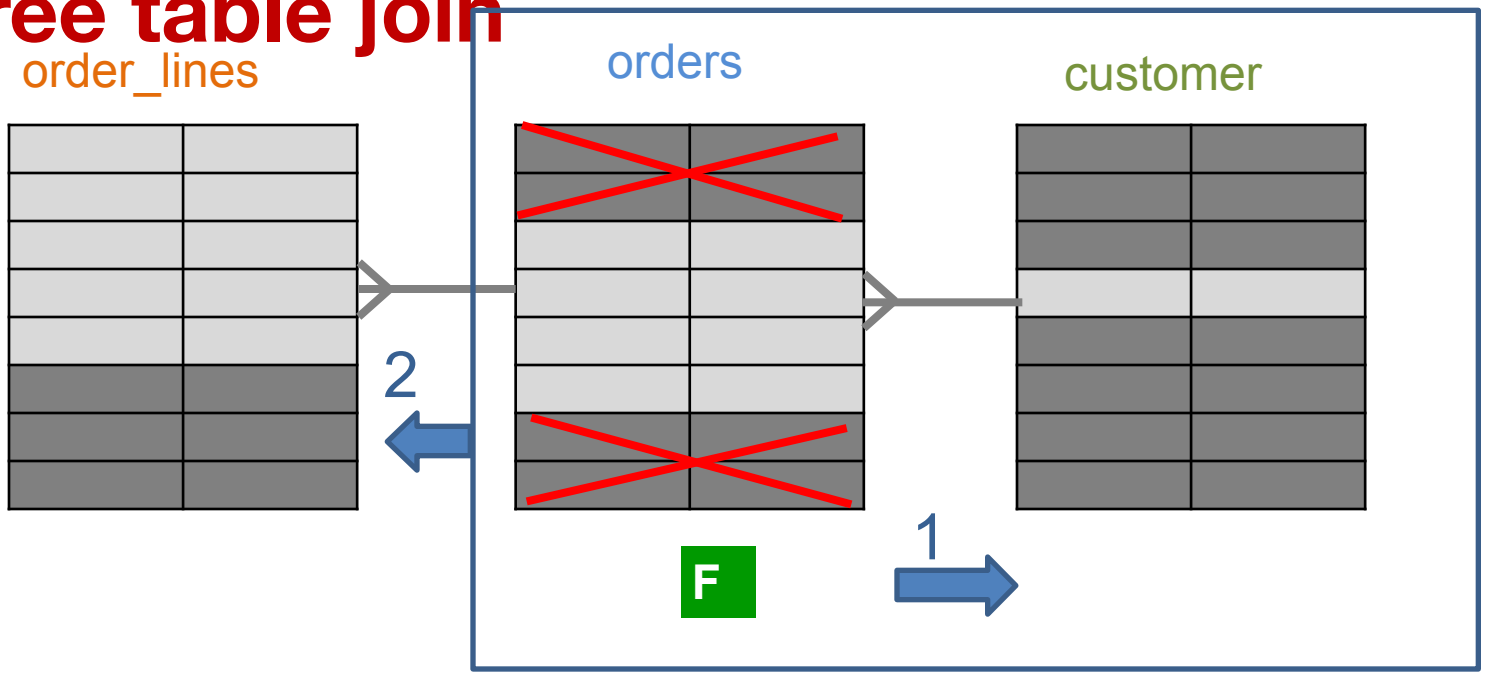
Three table join



Where
order_lines.field = value

Starting with a filter on order_lines is going to eliminate work

Three table join

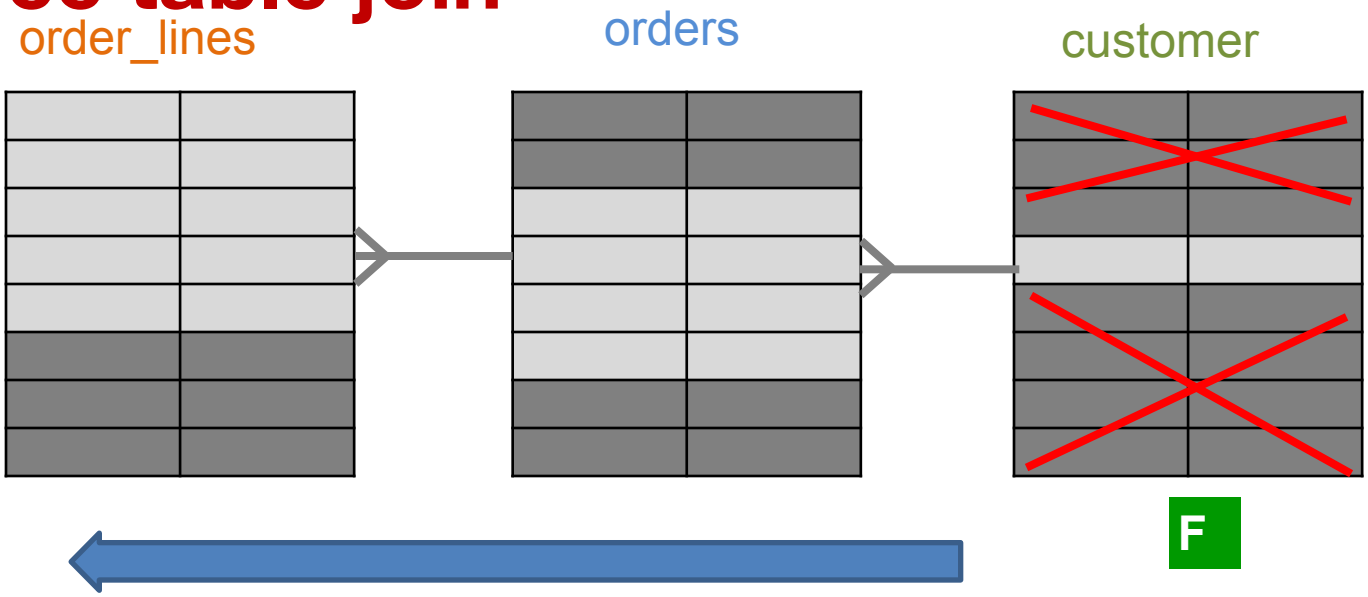


Where
orders.field = value

Starting with a Filter on orders is going to eliminate work

Then Join to customers => keeps the number of rows the same

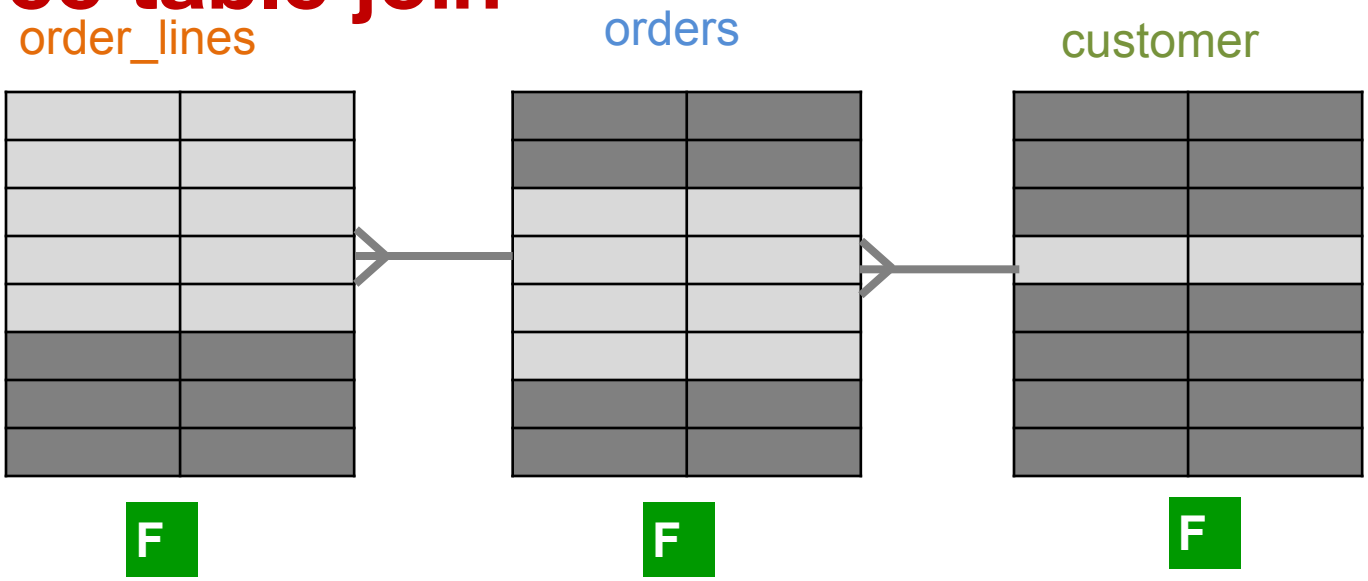
Three table join



Where
customer.field = value

Starting on a filter on customer is going to eliminate work

Three table join

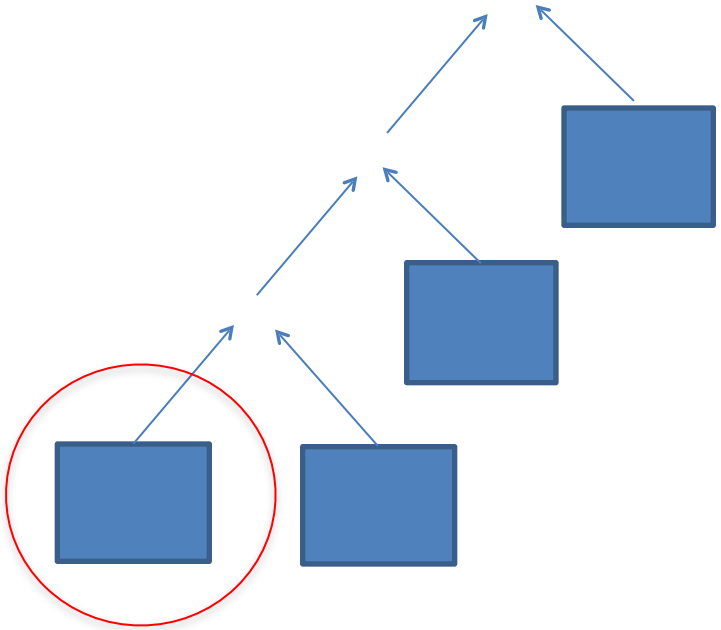


What if a filter on all three?

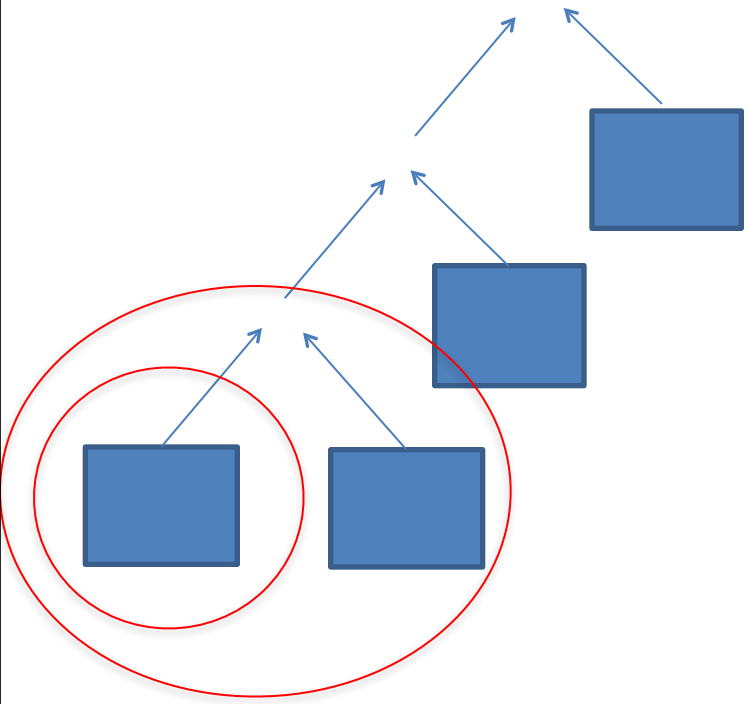
Choose the one that filter's the highest percent of the table.

```
100% * (select count(*) from TAB where condition)
--------------------------
(select count(*) from Tab)
```

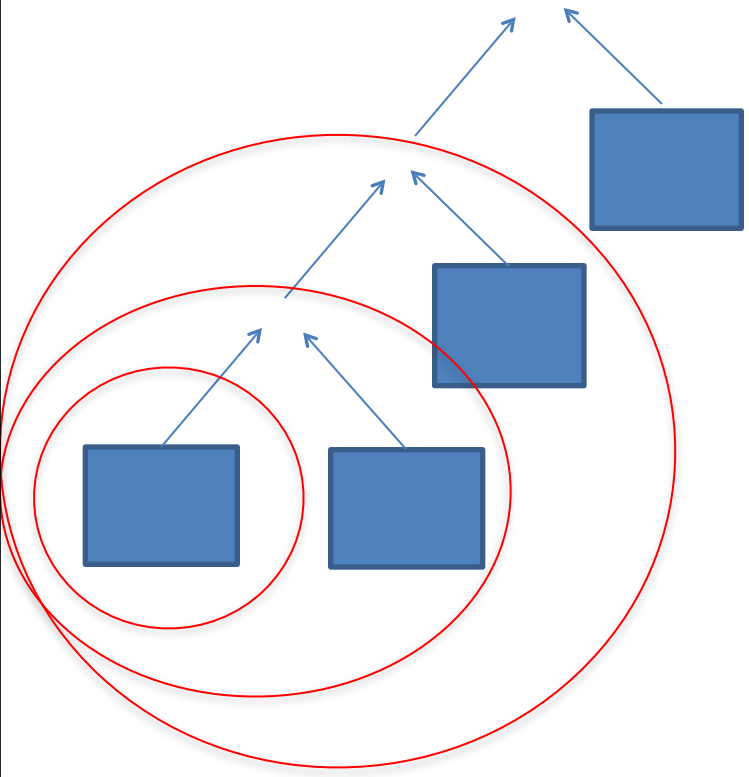
4 or more table join



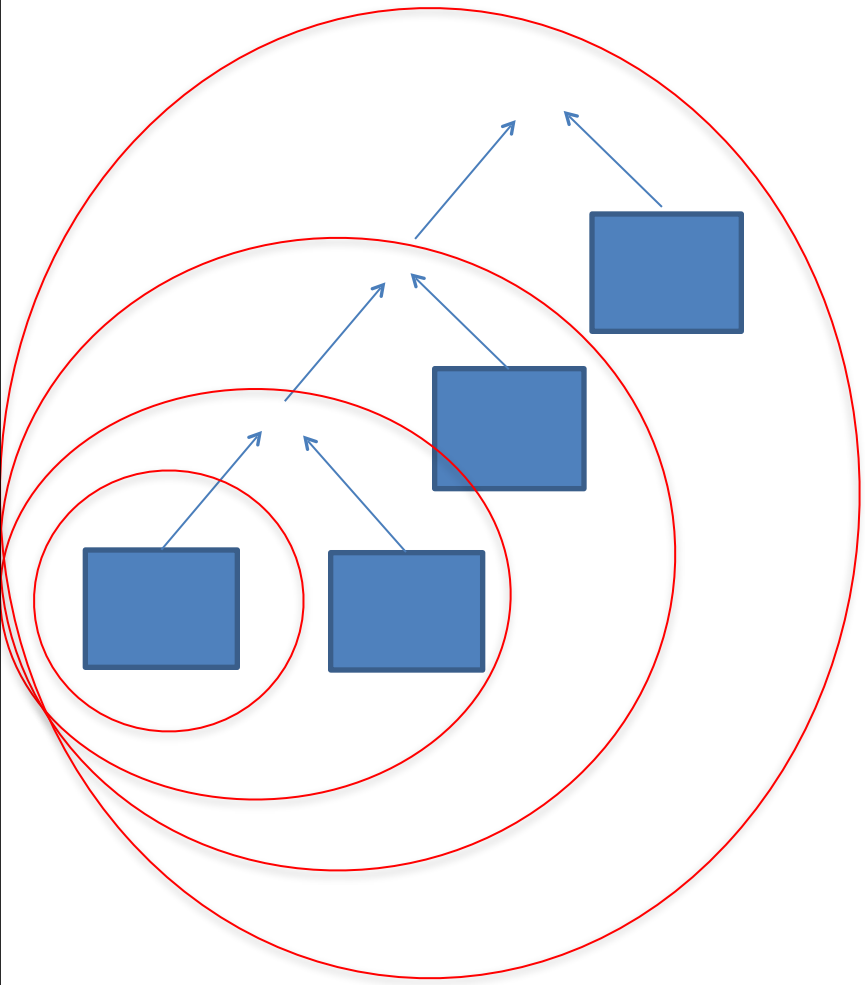
4 or more table join



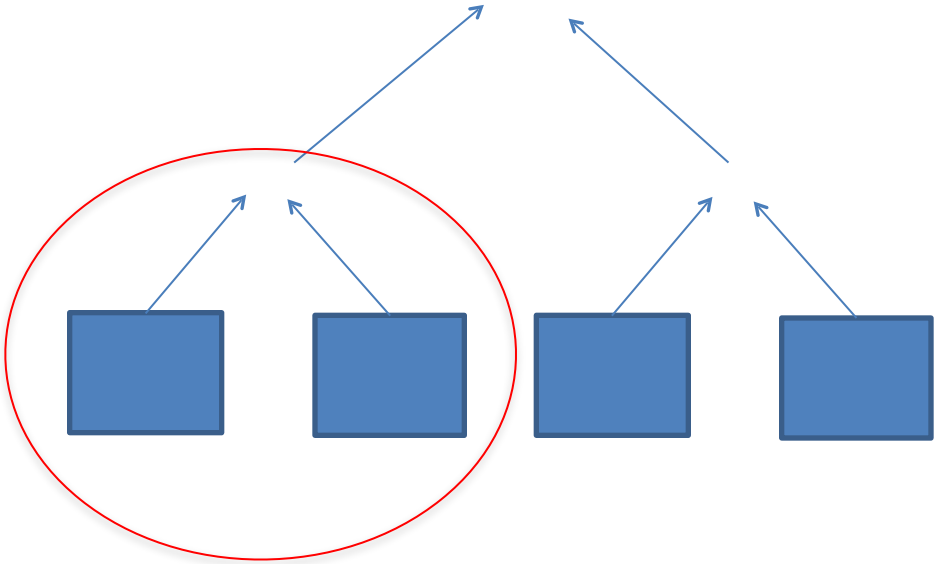
4 or more table join



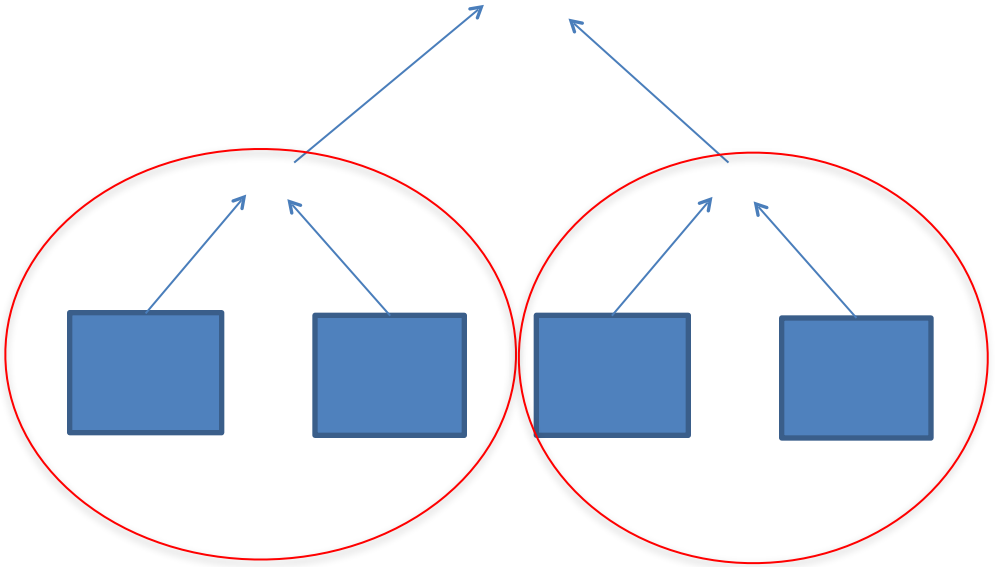
4 or more table join



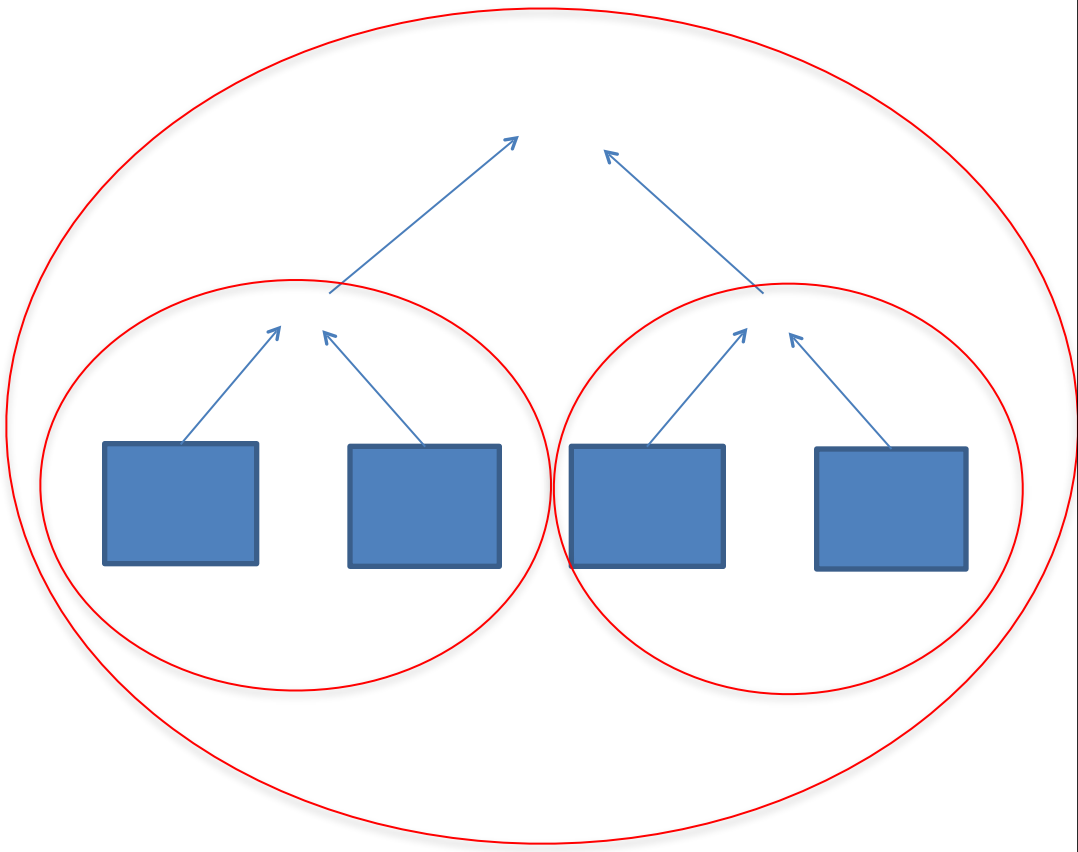
4 or more table join



4 or more table join

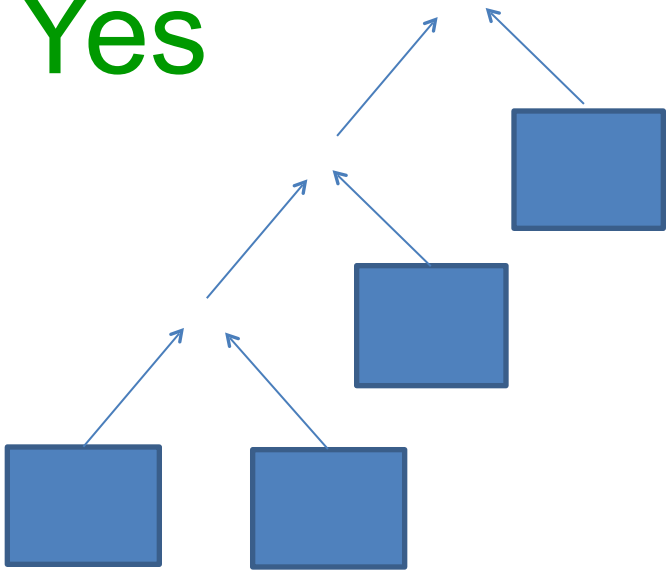


4 or more table join

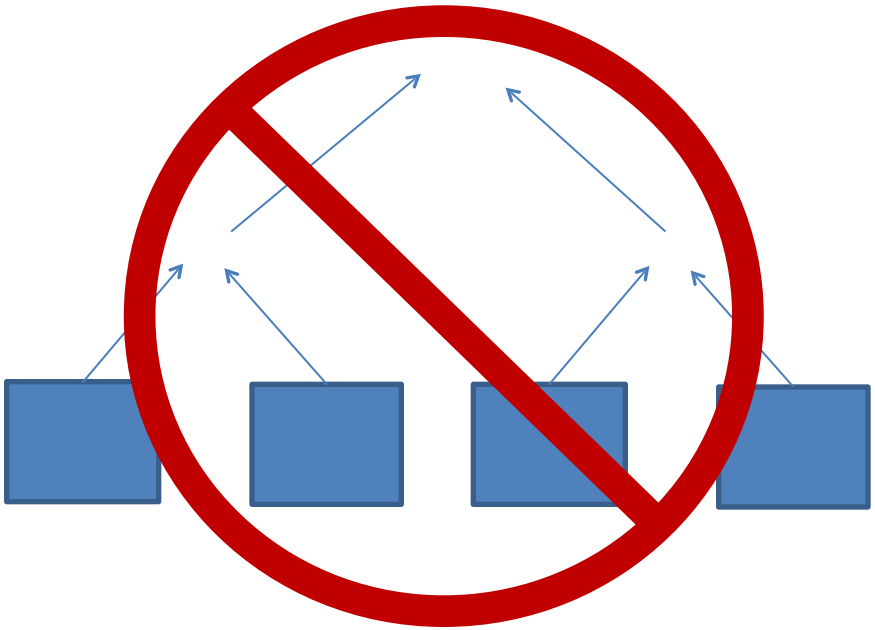


4 or more table join

Yes



No



Oracle can only join one table in at a time.
Oracle can't (normally) join two different sub-branches

Put it all together : VST

1. Tables
   - drawn as nodes
2. Joins :
   - drawn map connector lines
     - One-to-one, one-to-many, many-to-many
     - Exists, not exists, outer joins
3. Filters
   - mark on each table with filter in where clause

How to VST: Tables and Joins

```
SELECT C.Phone_Number, C.Honorific, C.First_Name, C.Last_Name,
       C.Suffix, C.Address_ID, A.Address_ID, A.Street_Address_Li
       A.Street_Address_Line2, A.City_Name, A.State_Abbreviation
       A.ZIP_Code, OD.Deferred_Shipment_Date, OD.Item_Count,
       ODT.Text, OT.Text, P.Product_Description, S.Shipment_Date
FROM Orders O, Order_Details OD, Products P, Customers C, Shipme
     Addresses A, Code_Translations ODT, Code_Translations OT
WHERE UPPER(C.Last_Name) LIKE :Last_Name||'%'
  AND UPPER(C.First_Name) LIKE :First_Name||'%'
  AND OD.Order_ID = O.Order_ID
  AND O.Customer_ID = C.Customer_ID
  AND OD.Product_ID = P.Product_ID(+)
  AND OD.Shipment_ID = S.Shipment_ID(+)
  AND S.Address_ID = A.Address_ID(+)
  AND O.Status_Code = OT.Code
  AND OD.Status_Code = ODT.Code
  AND O.Order_Date > :Now - 366
ORDER BY C.Customer_ID, O.Order_ID DESC, S.Shipment_ID, OD.Order
```

Tables

```
Orders O,
Order_Details OD,
Products P,
Customers C,
Shipments S,
Addresses A,
Code_Translations ODT,
Code_Translations OT
```

Joins

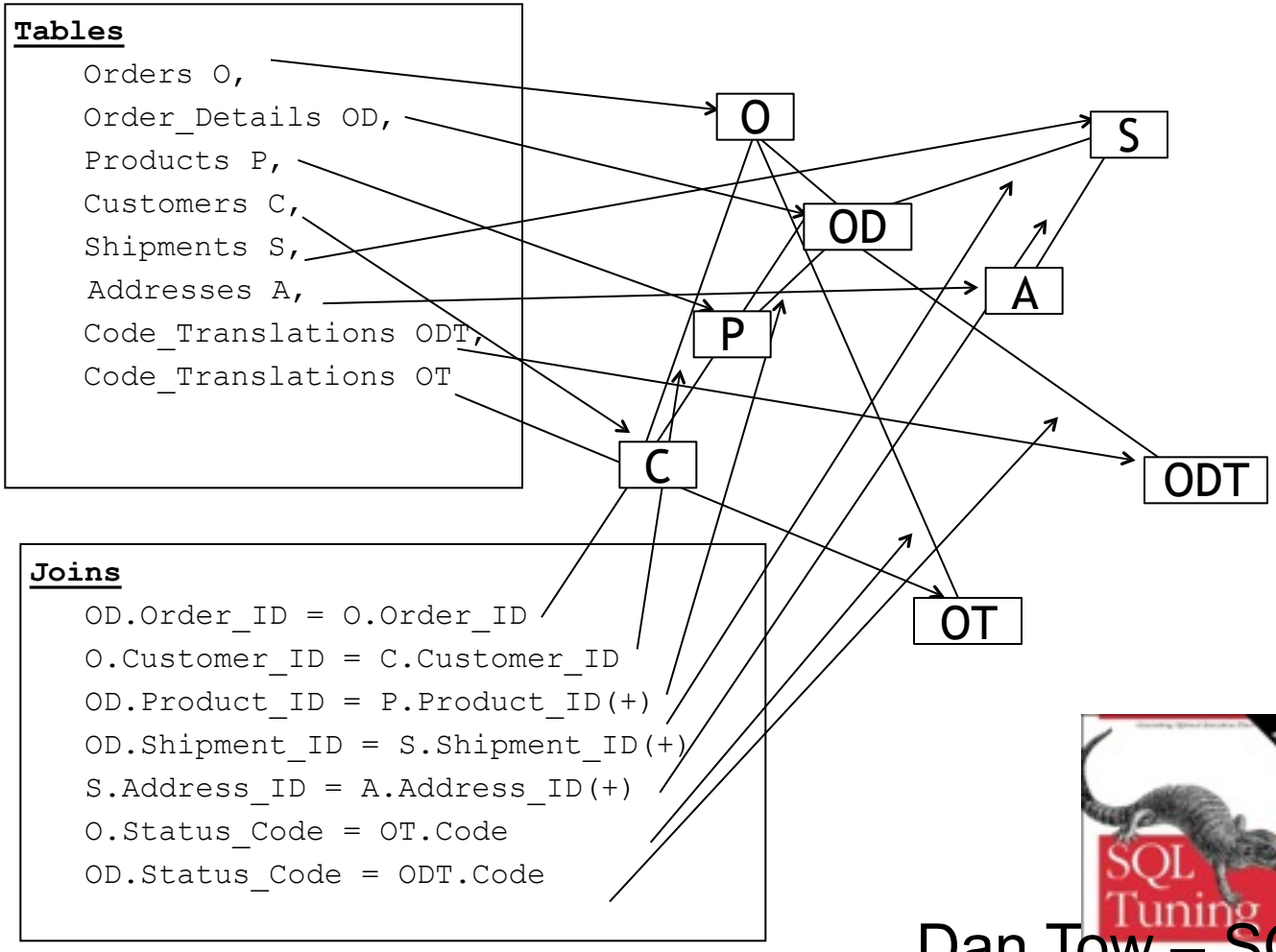
```
OD.Order_ID = O.Order_ID
O.Customer_ID = C.Customer_ID
OD.Product_ID = P.Product_ID(+)
OD.Shipment_ID = S.Shipment_ID(+)
S.Address_ID = A.Address_ID(+)
O.Status_Code = OT.Code
OD.Status_Code = ODT.Code
```

Dan Tow – SQL TUNING

Filters

```
WHERE UPPER(C.Last_Name) LIKE :Last_Name||'%'
  AND UPPER(C.First_Name) LIKE :First_Name||'%'
  AND O.Order_Date > :Now - 366
```

Layout tables and connections

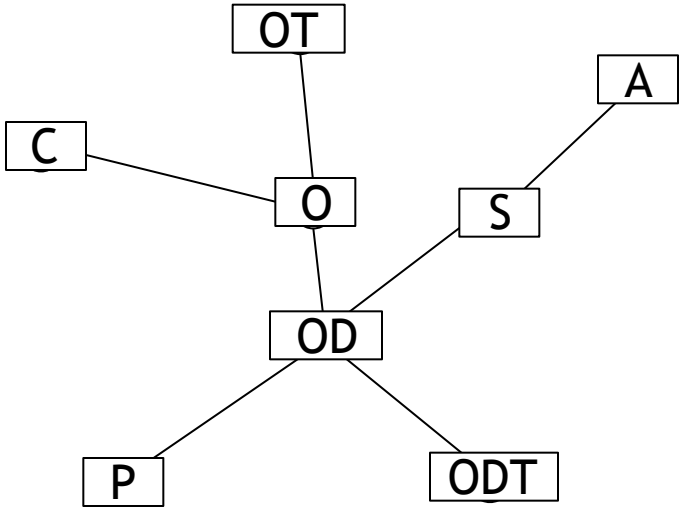


Dan Tow – SQL TUNING

Unstructured

Joins

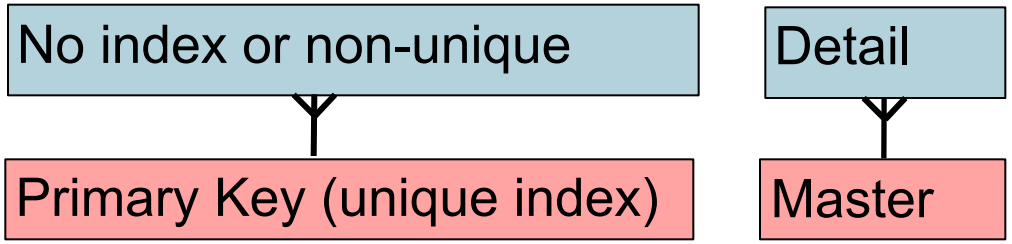
```
OD.Order_ID = O.Order_ID
O.Customer_ID = C.Customer_ID
OD.Product_ID = P.Product_ID(+)
OD.Shipment_ID = S.Shipment_ID(+)
S.Address_ID = A.Address_ID(+)
O.Status_Code = OT.Code
OD.Status_Code = ODT.Code
```

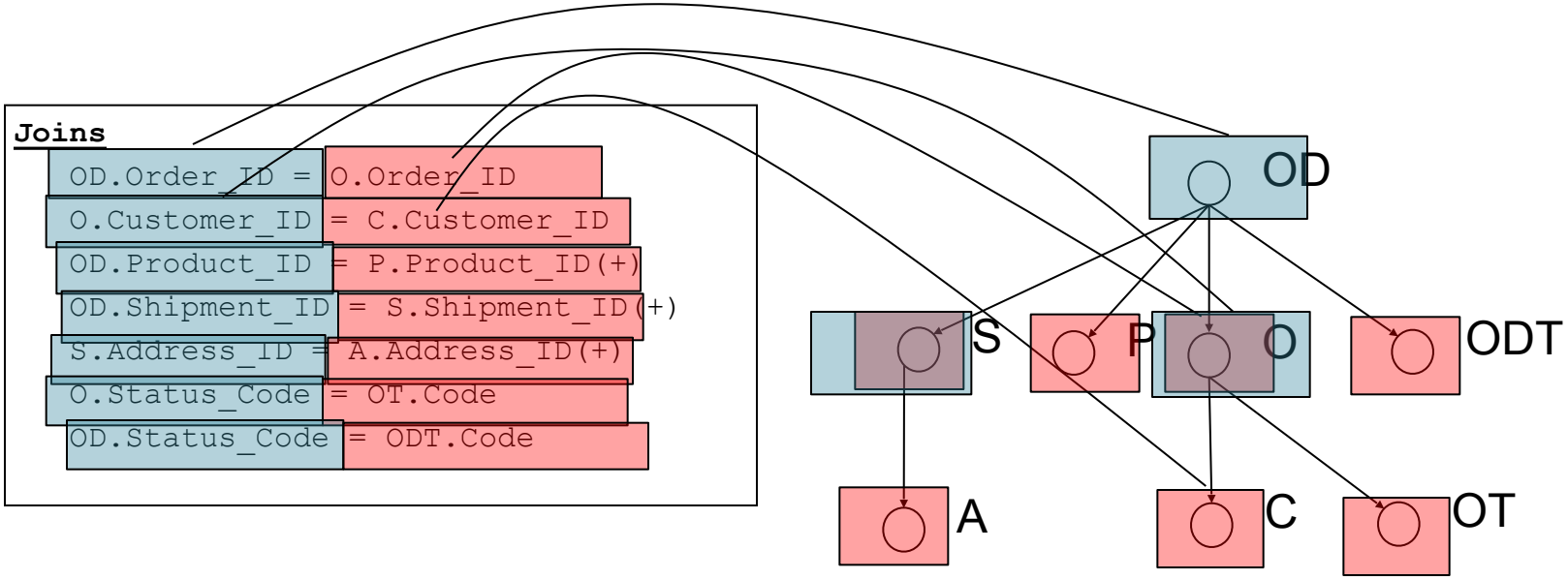


Neater, but can you do anything with it?
What's the optimal execution path?

Parents and Children

Structure the tree

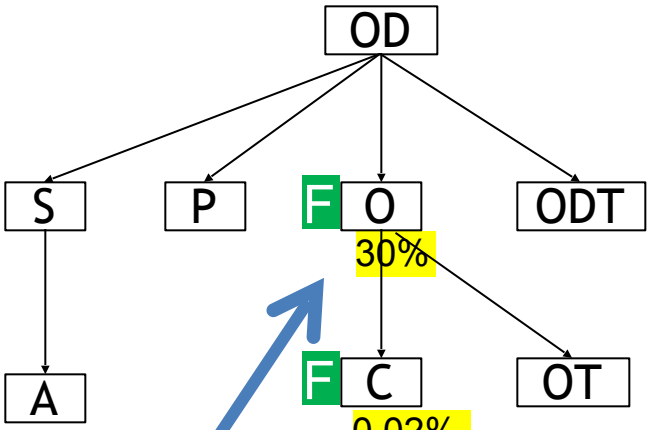




VST – filters and best path

- Filters help determine best path

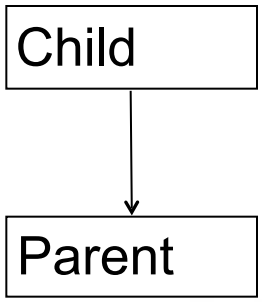
```
Filters
WHERE UPPER(C.Last_Name) LIKE :Last_Name||'%'
  AND UPPER(C.First_Name) LIKE :First_Name||'%'
  AND O.Order_Date > :Now - 366
```

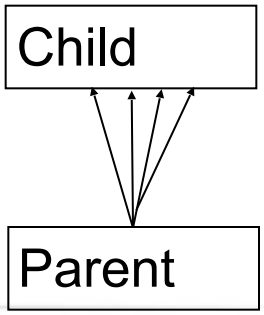



```
100% * (select count(*) from TAB where condition)
---------------------------
(select count(*) from Tab)
```

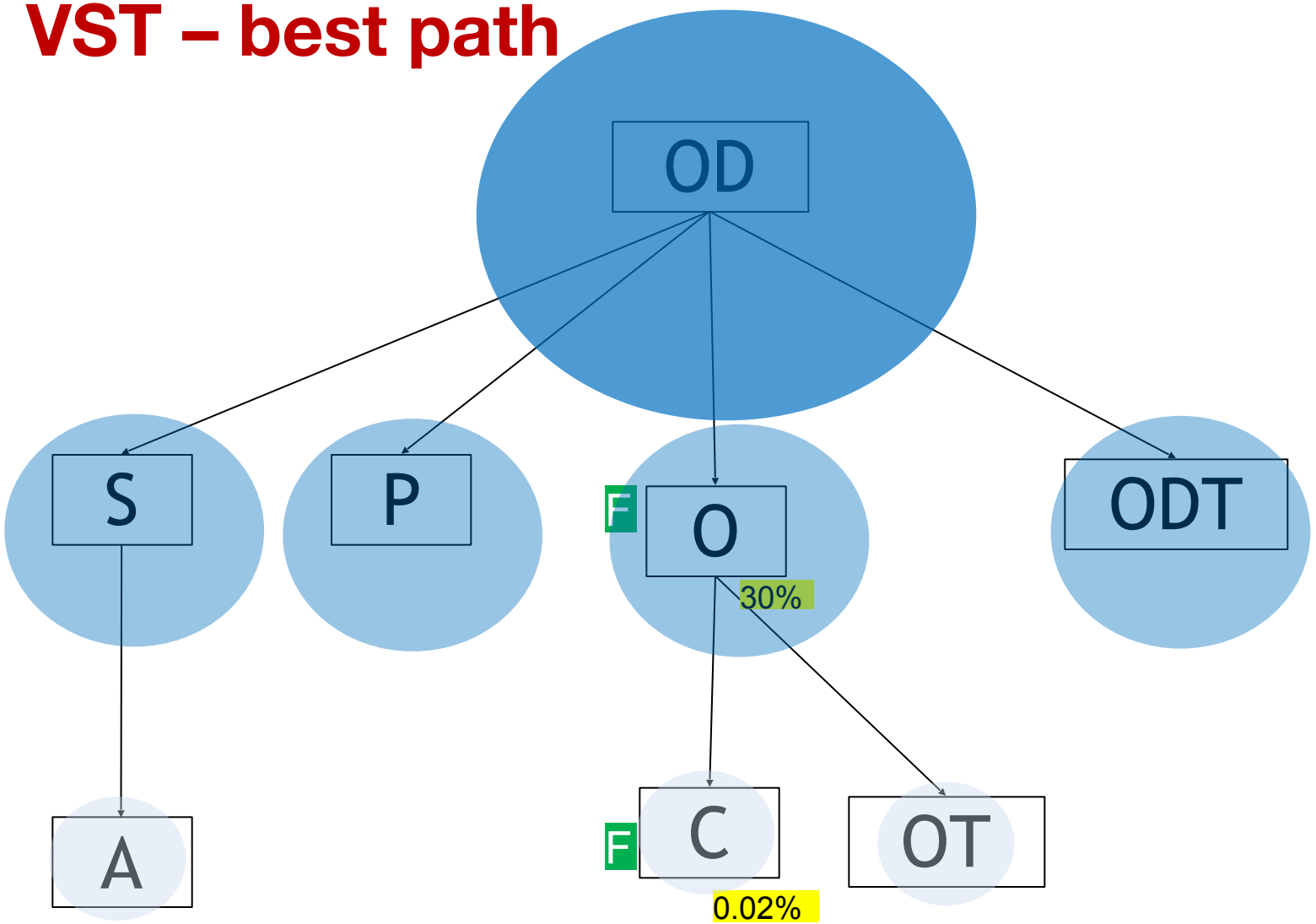
Concept:

1. Start at most selective filter
2. Join down first, before joining upwards

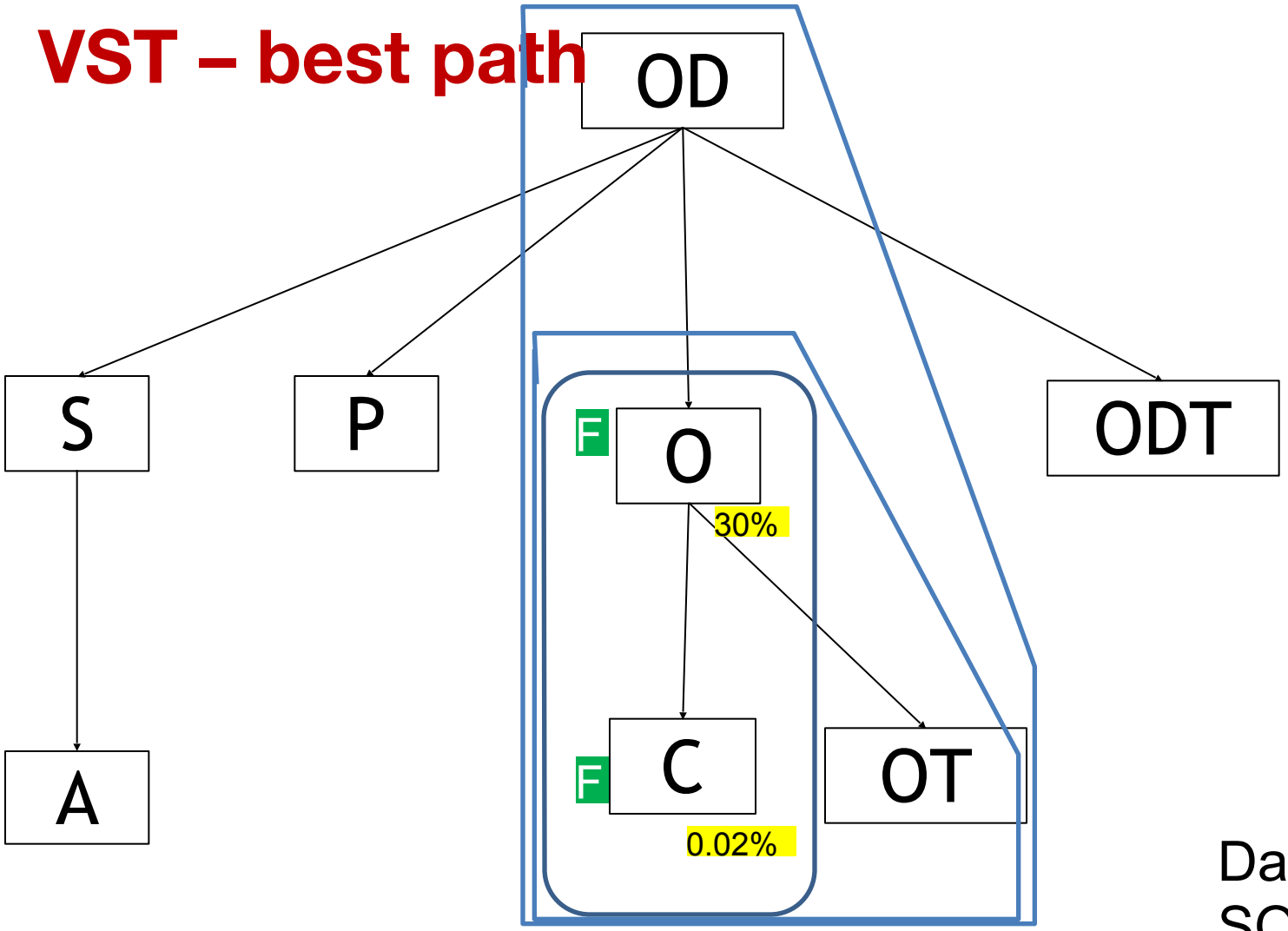




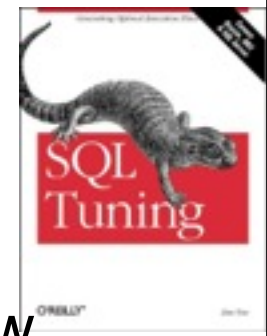
VST – best path



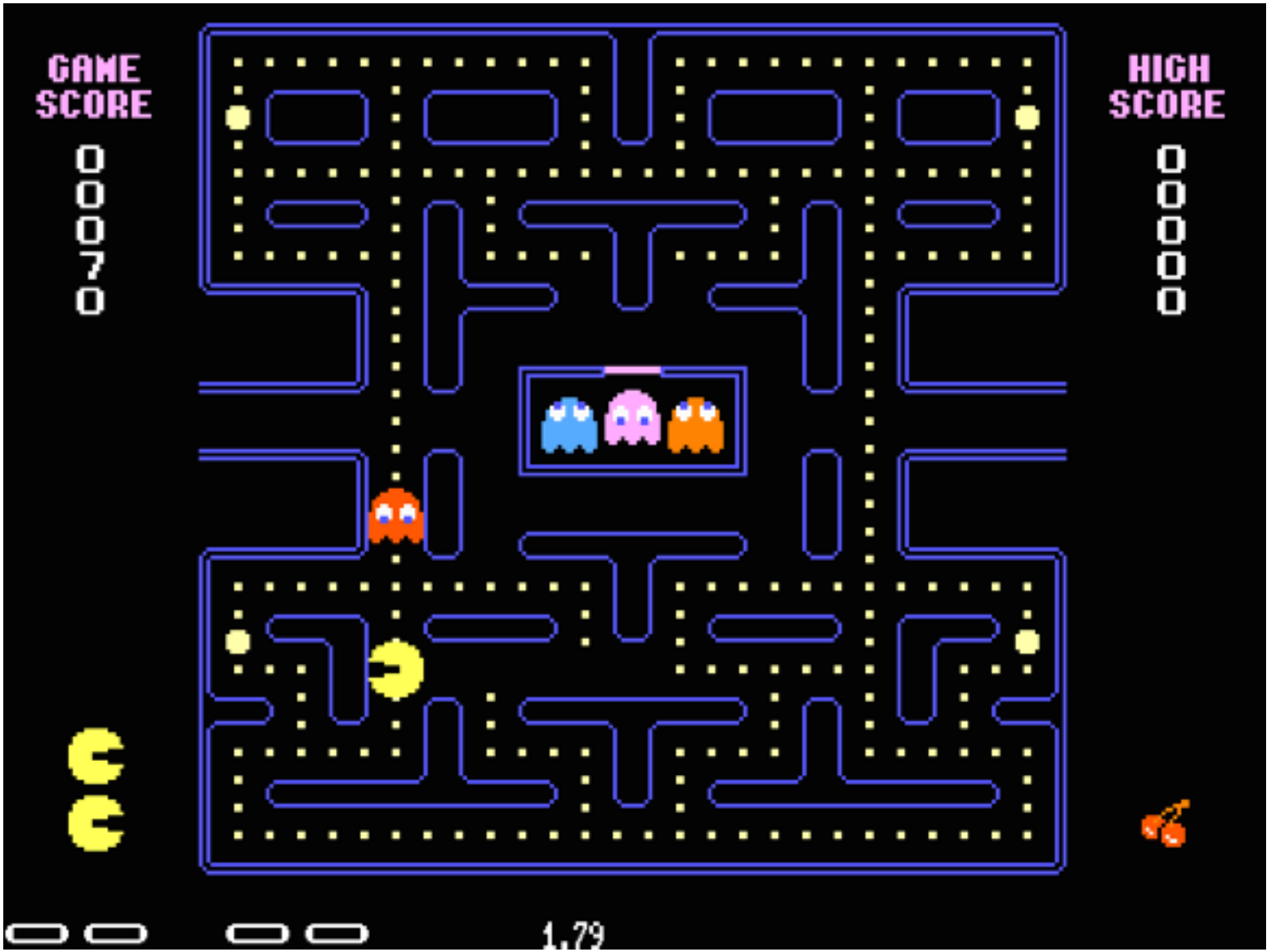
VST – best path

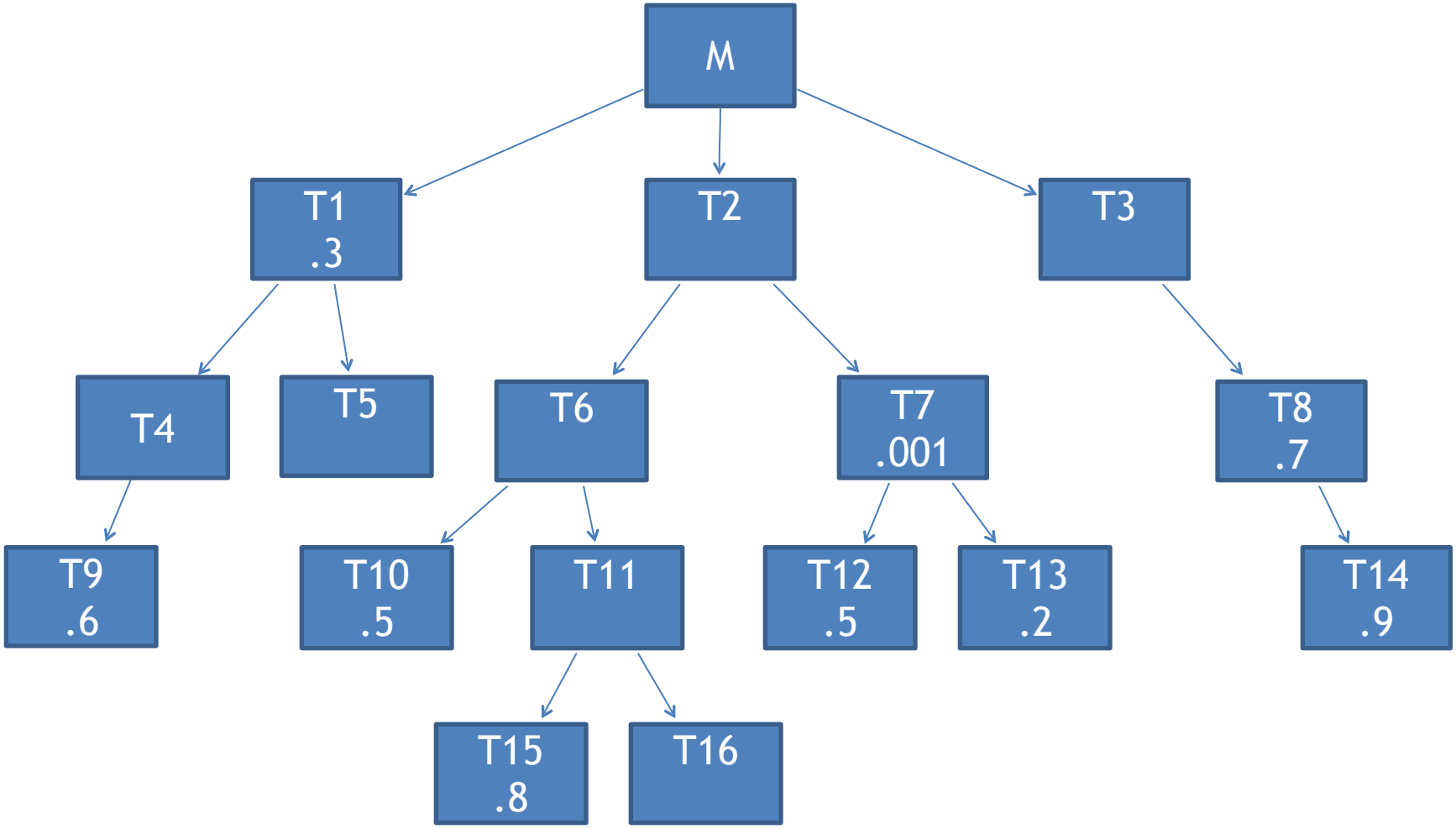


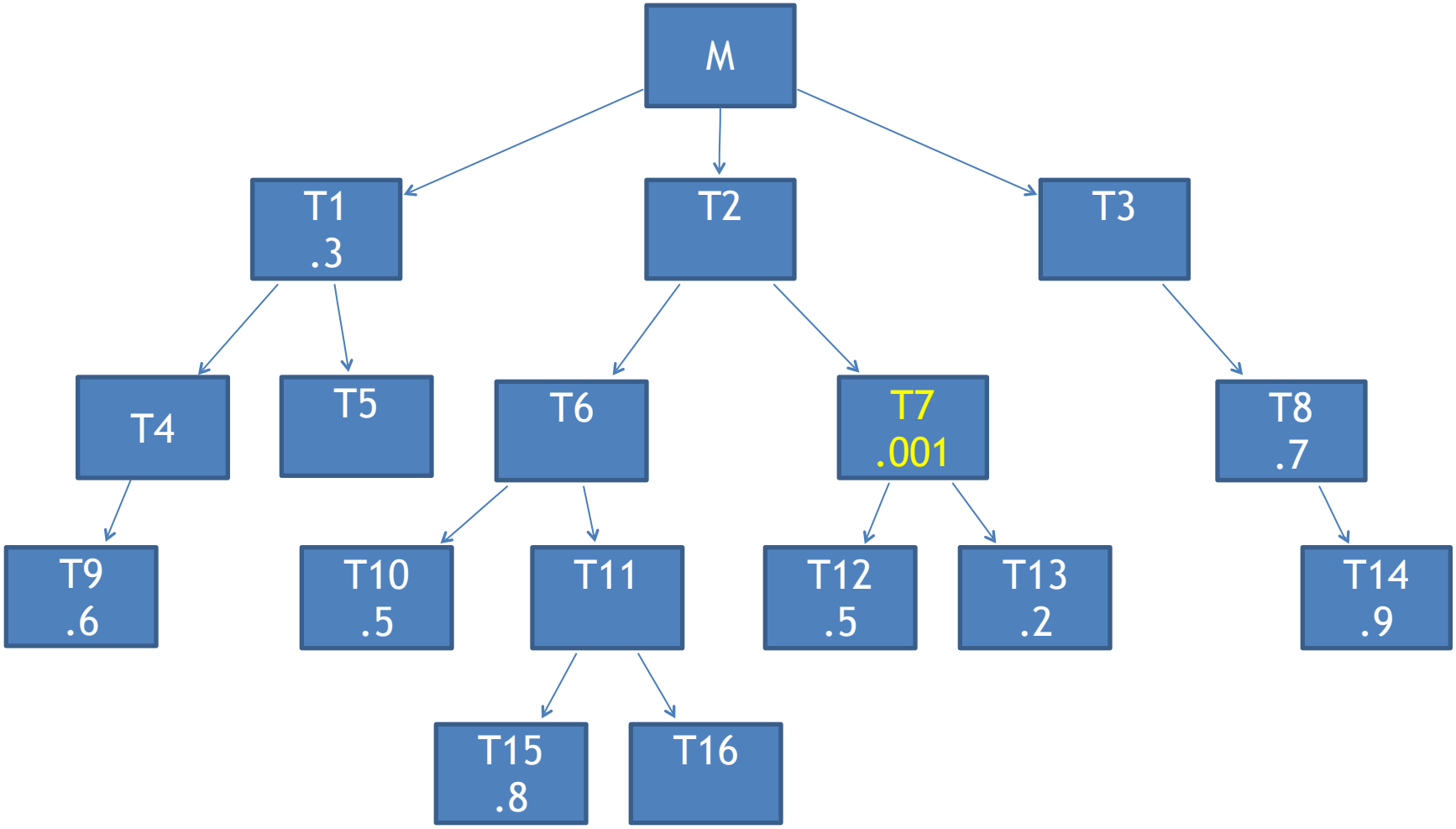
/*+ Leading (C,O,OT, OD) */



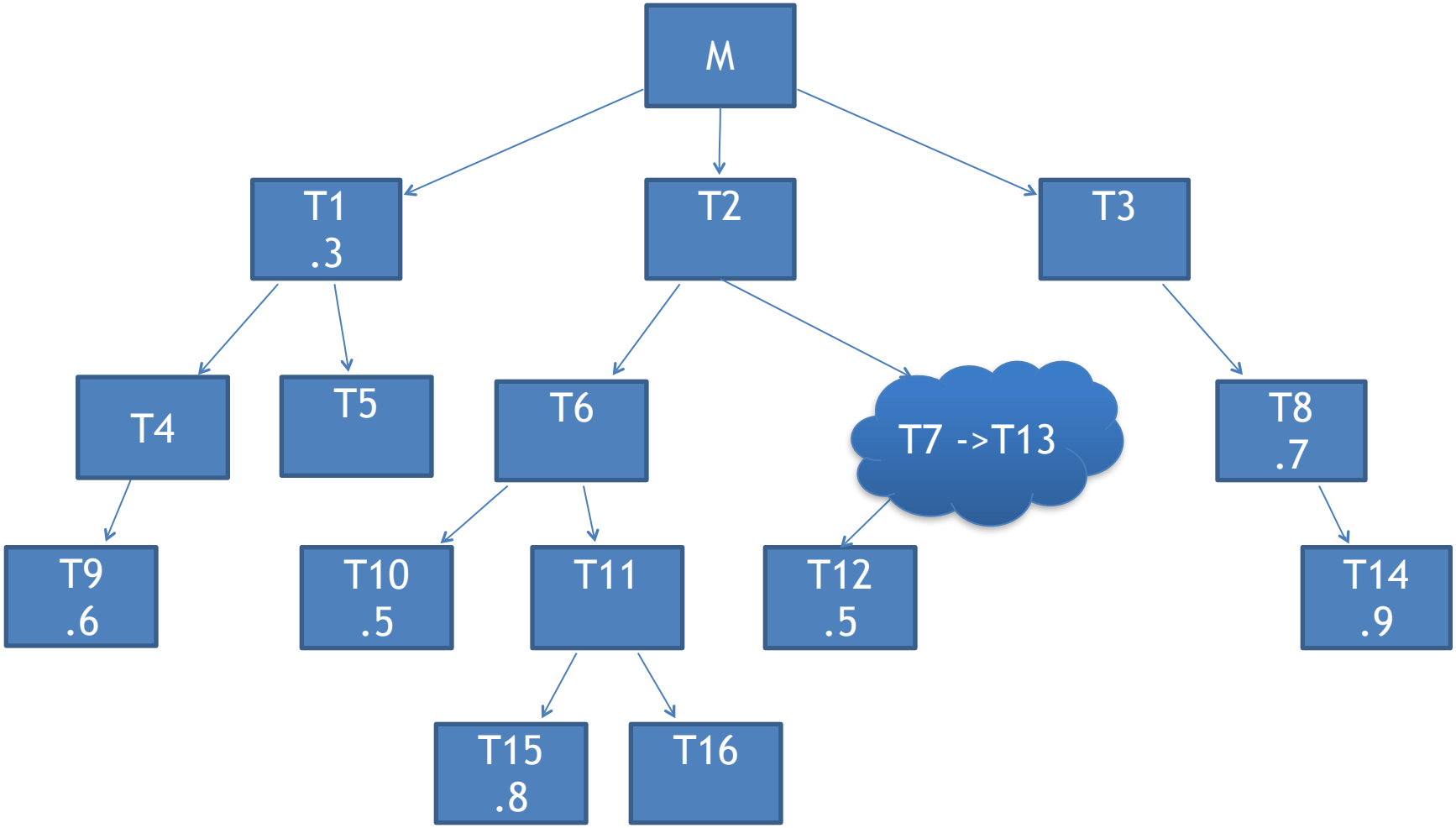
Dan Tow
SQL TUNING


GAME
SCORE
0070
HIGH
SCORE
0000
1.79

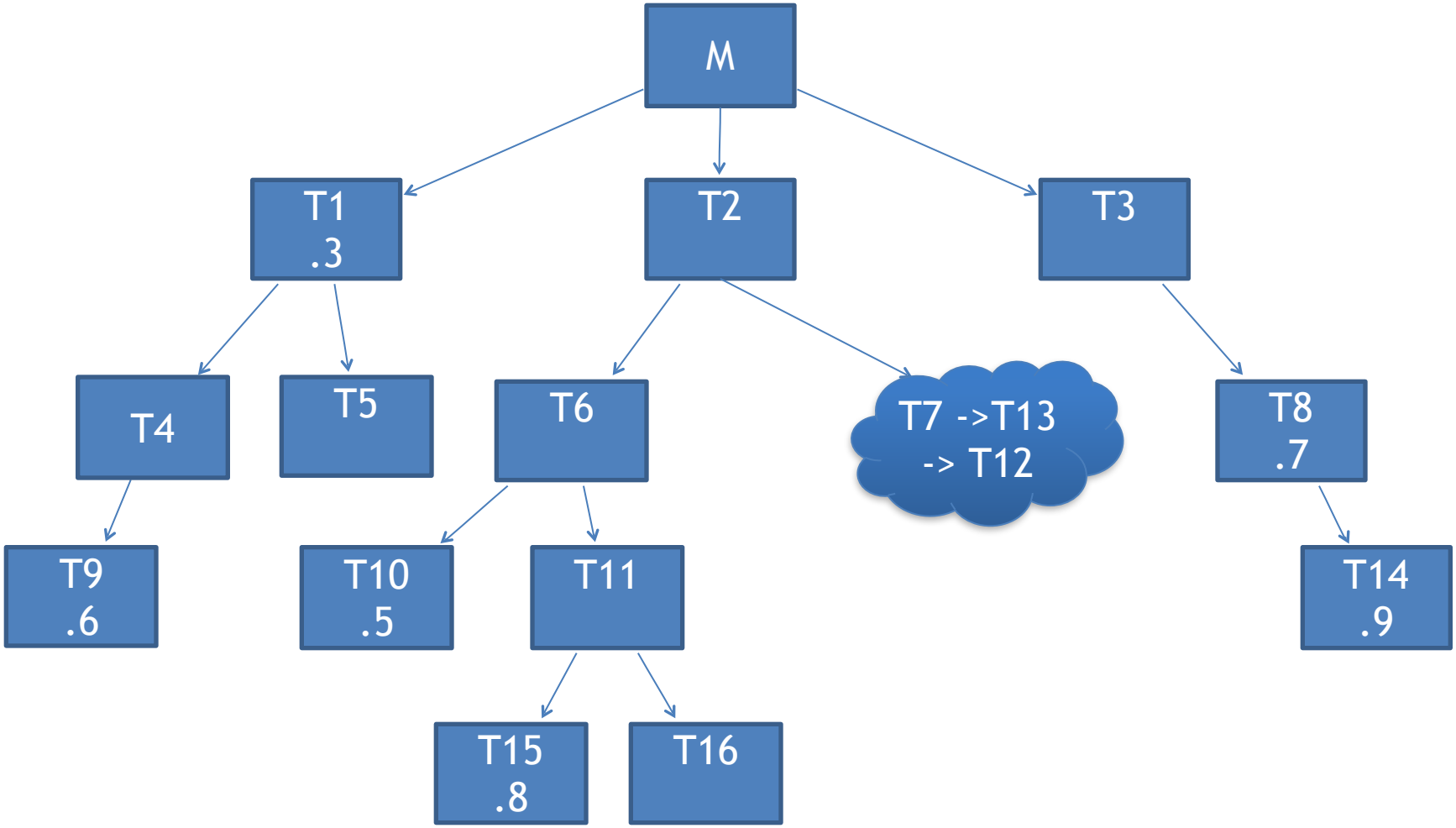

M
T1
.3
T2
T3
T4
T5
T6
T7
.001
T8
.7
T9
.6
T10
.5
T11
T12
.5
T13
.2
T14
.9
T15
.8
T16


M
T1
.3
T2
T3
T4
T5
T6
T7
.001
T8
.7
T9
.6
T10
.5
T11
T12
.5
T13
.2
T14
.9
T15
.8
T16

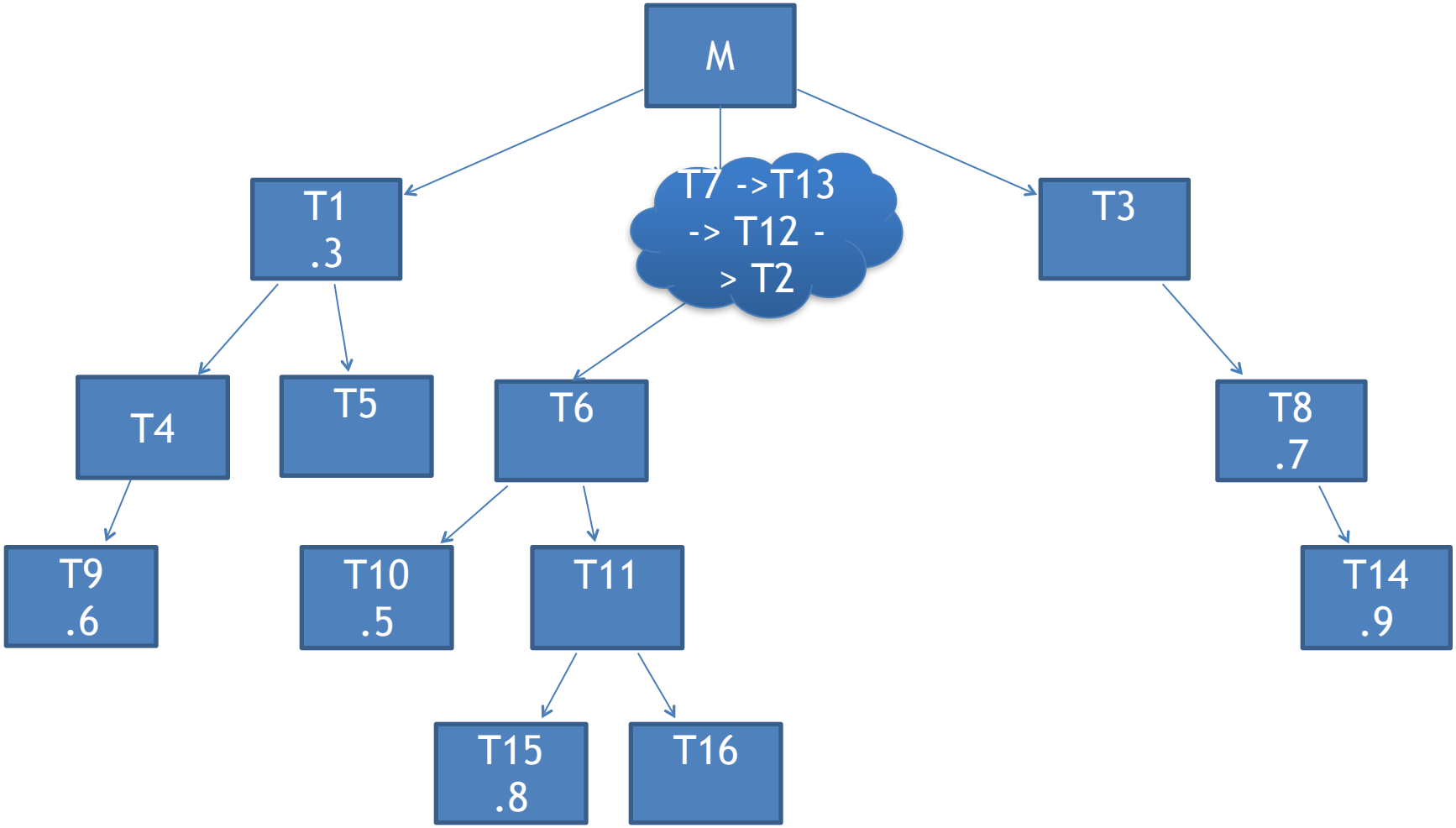
T7 -> T13



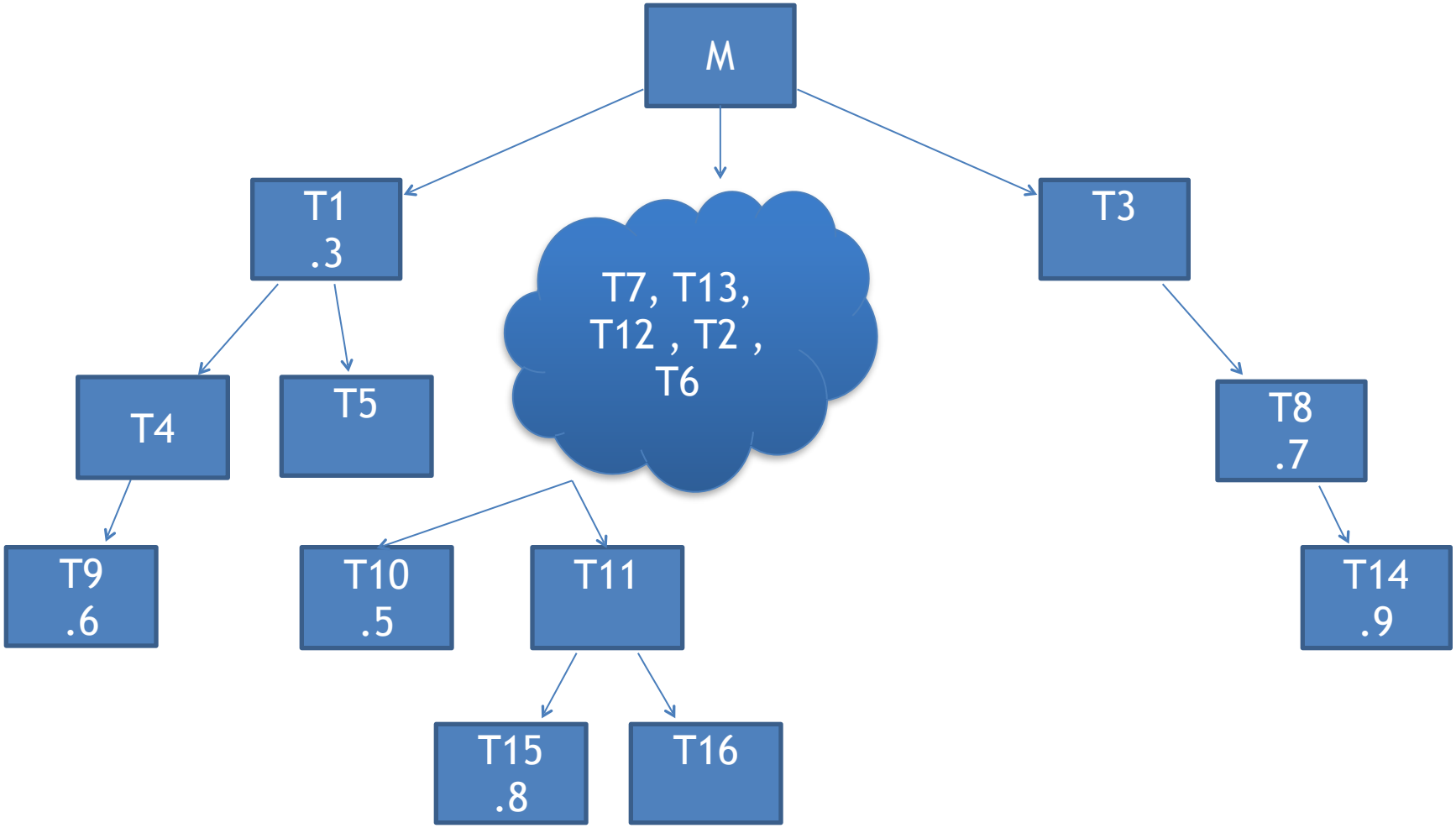
T7 -> T13



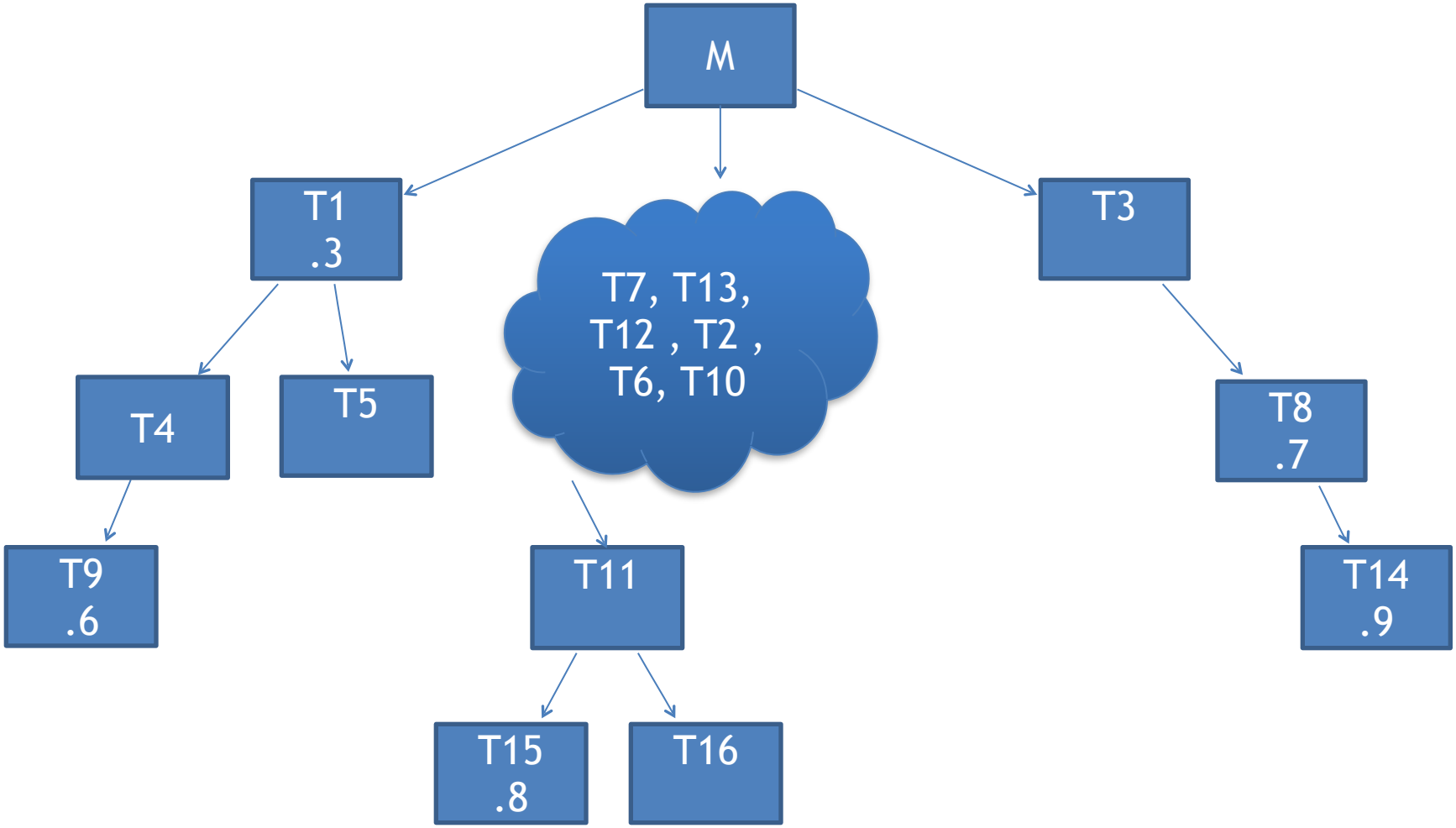
T7 -> T13



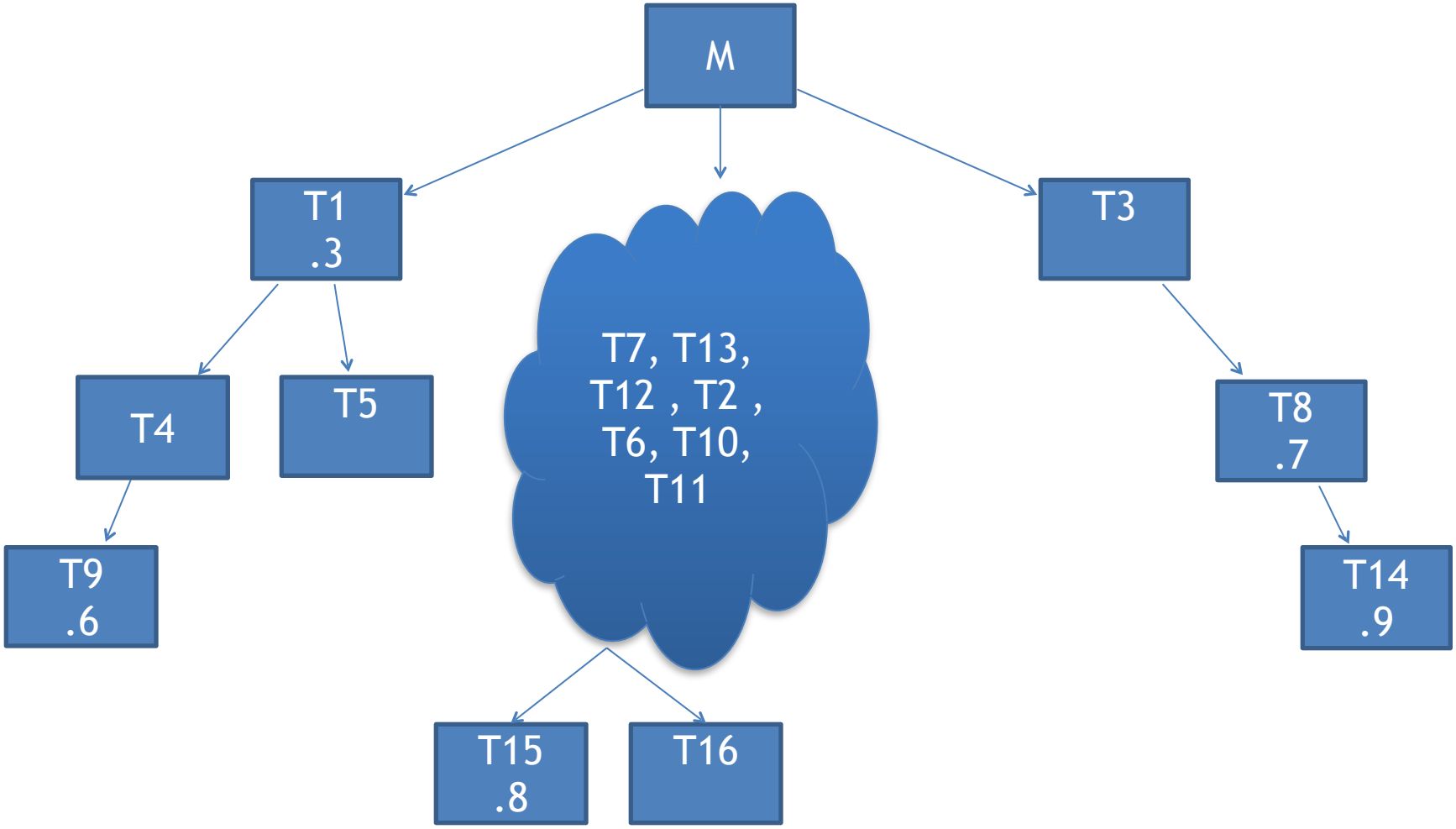
T7 -> T13



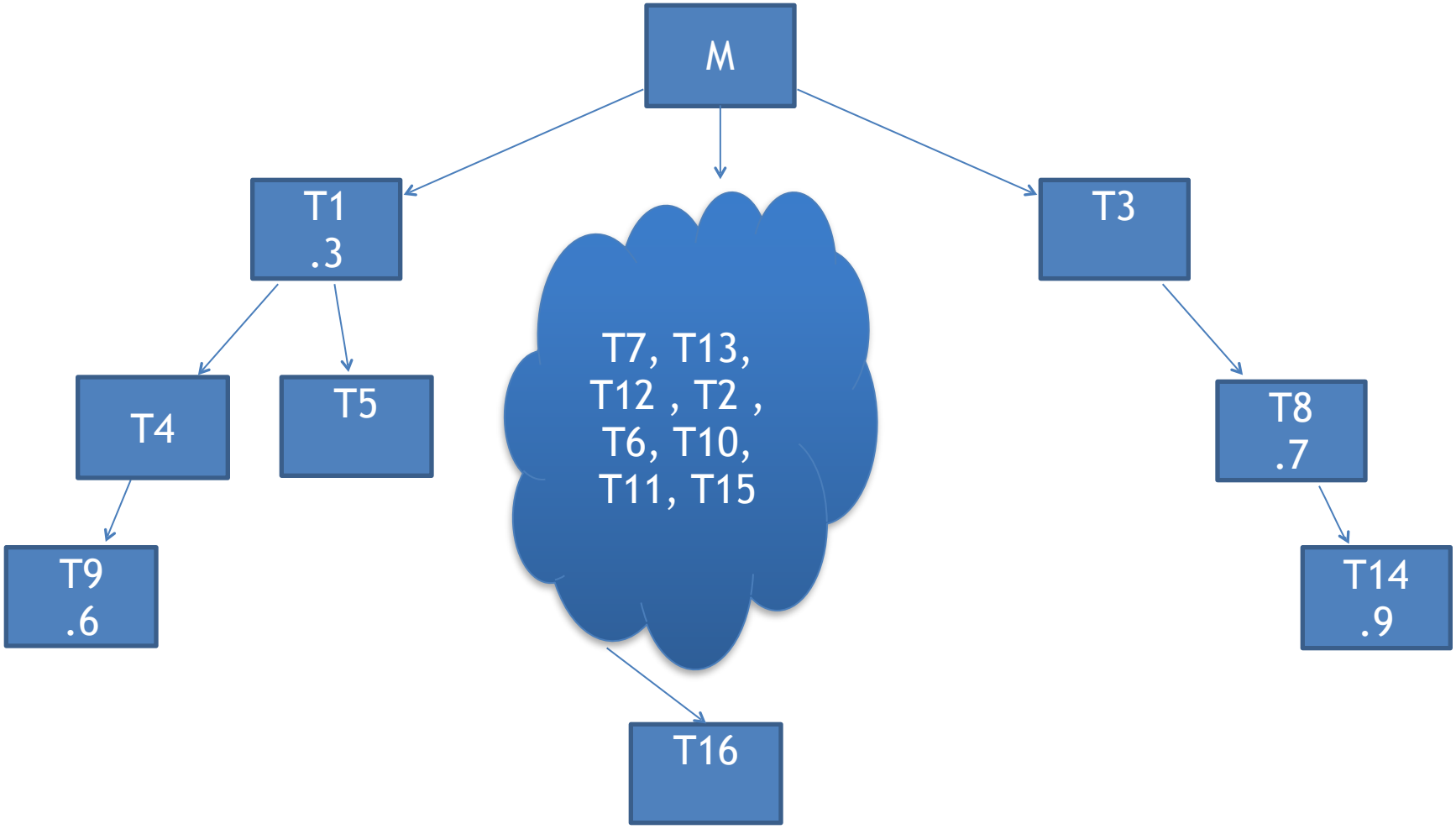
T7 -> T13



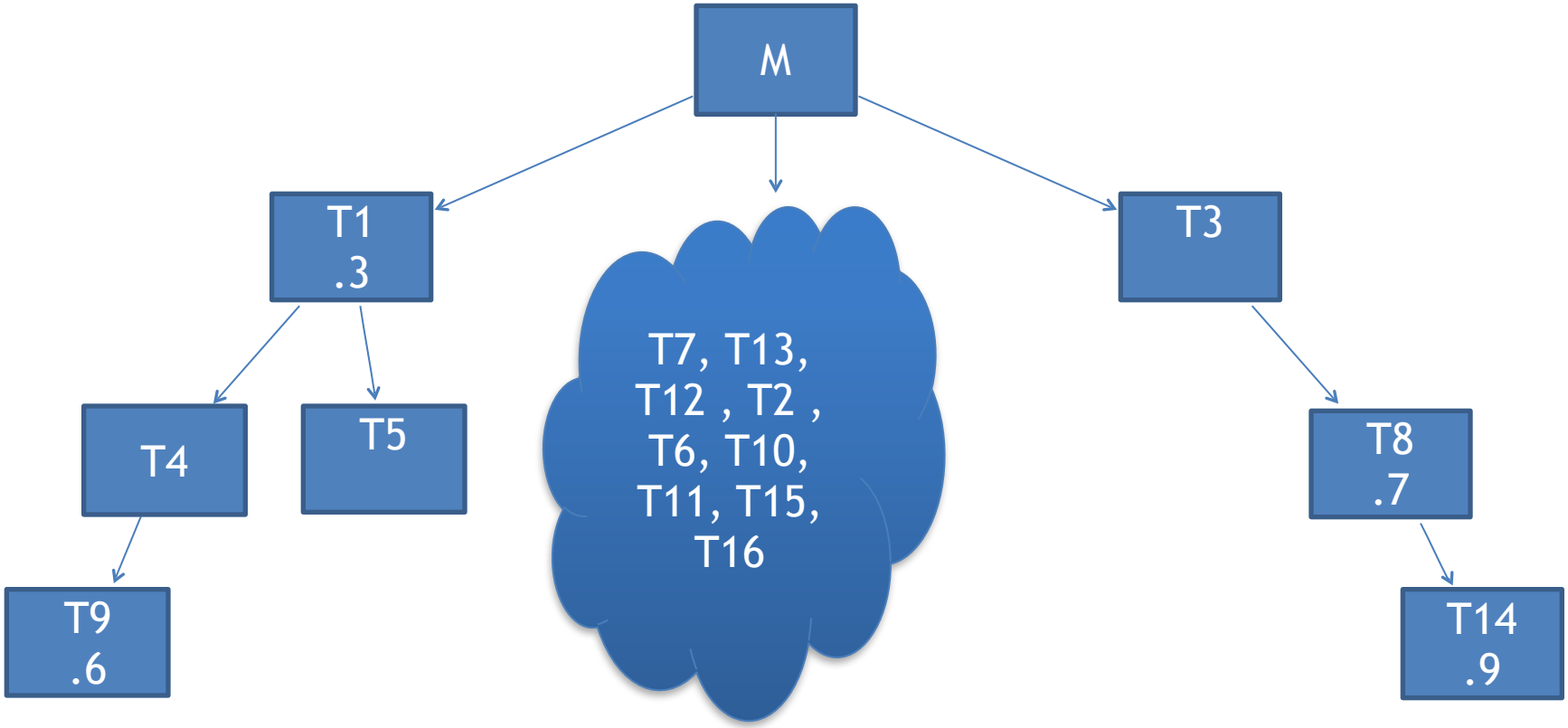
T7 -> T13



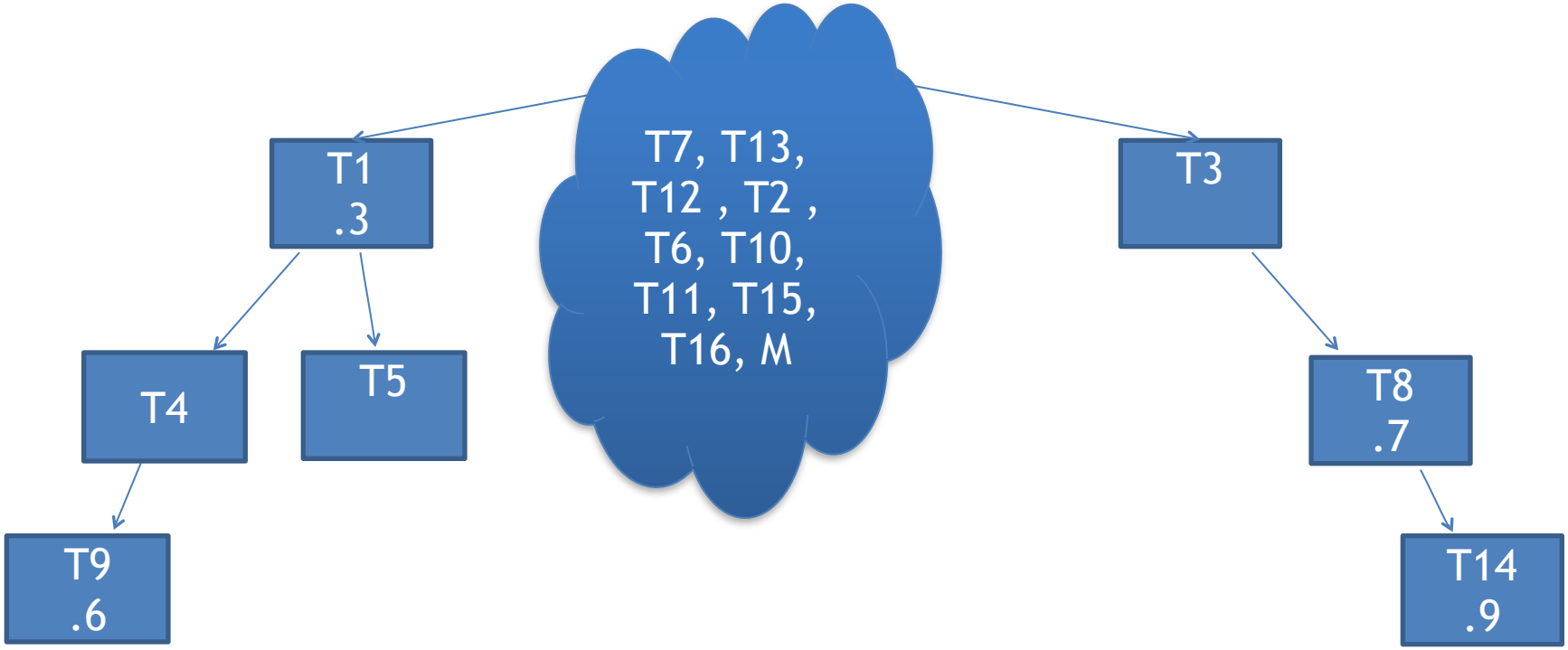
T7 -> T13



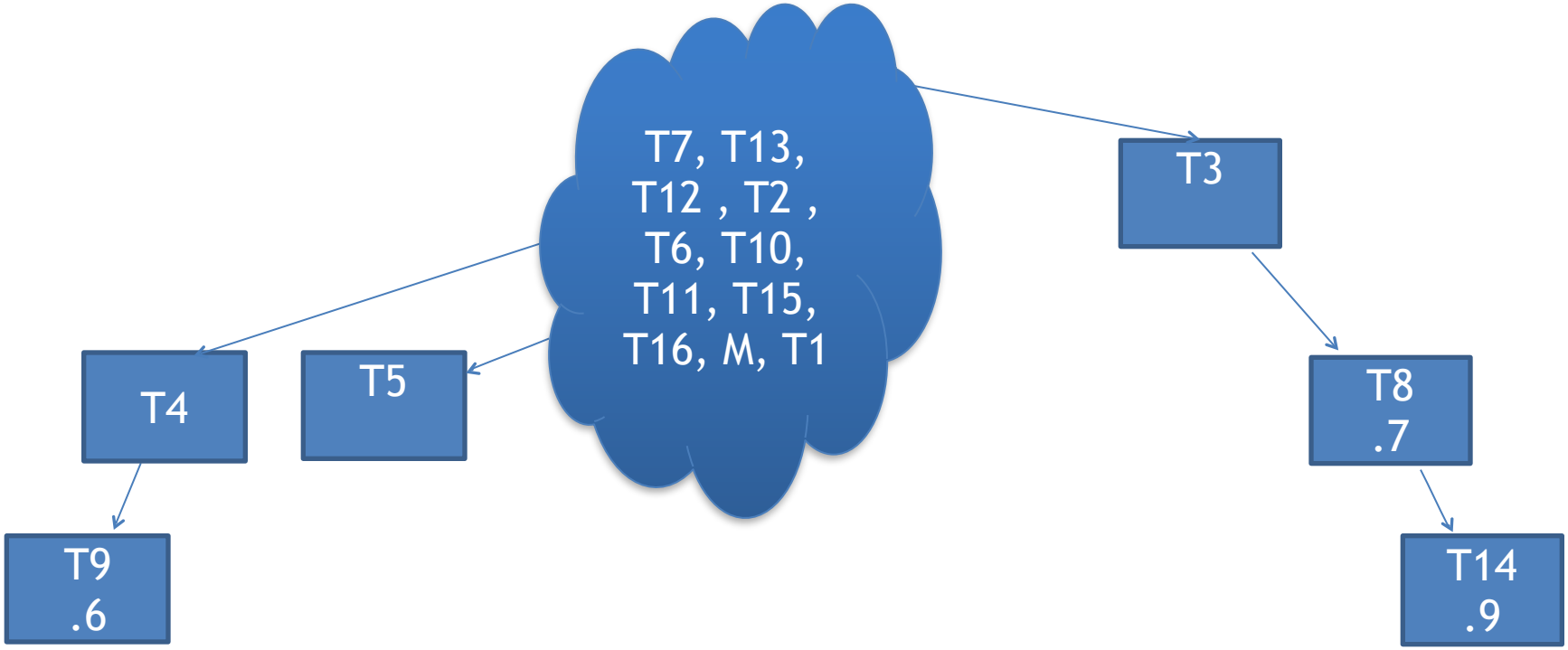
T7 -> T13



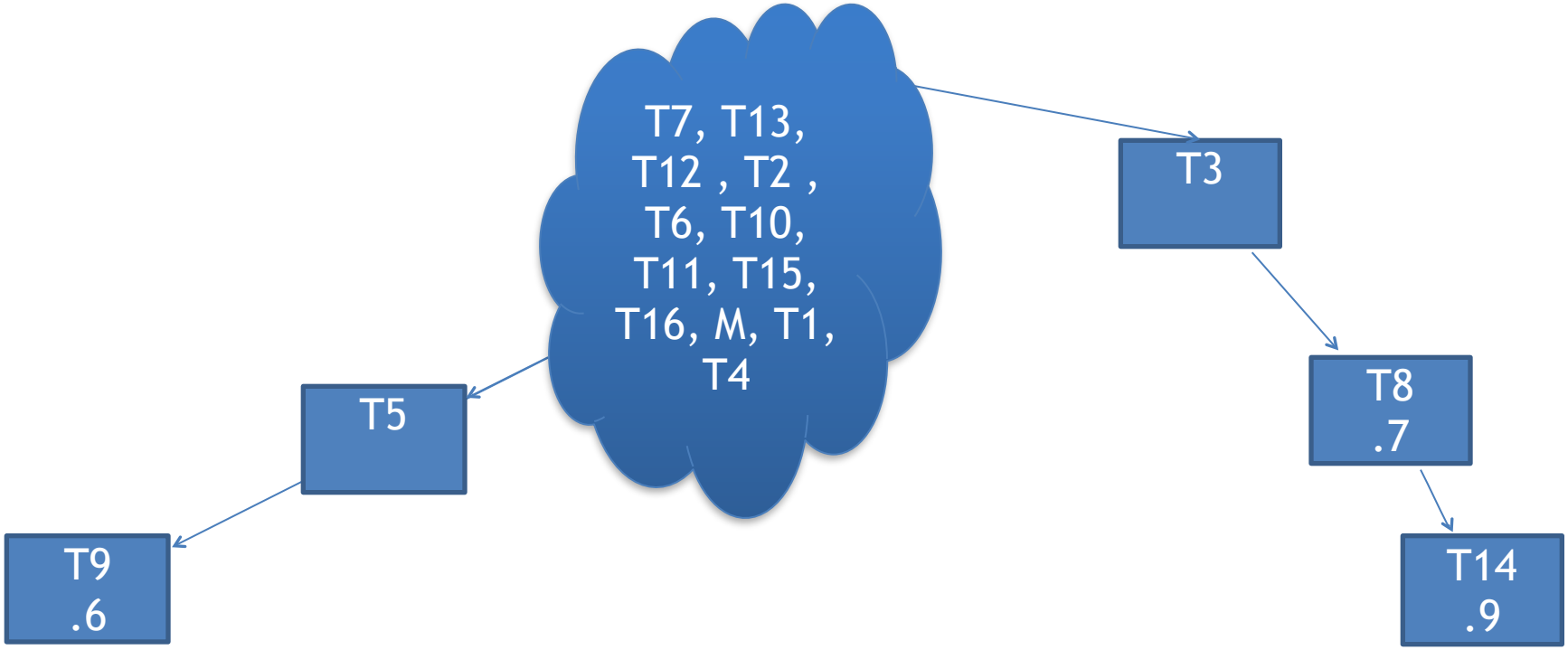
T7 -> T13



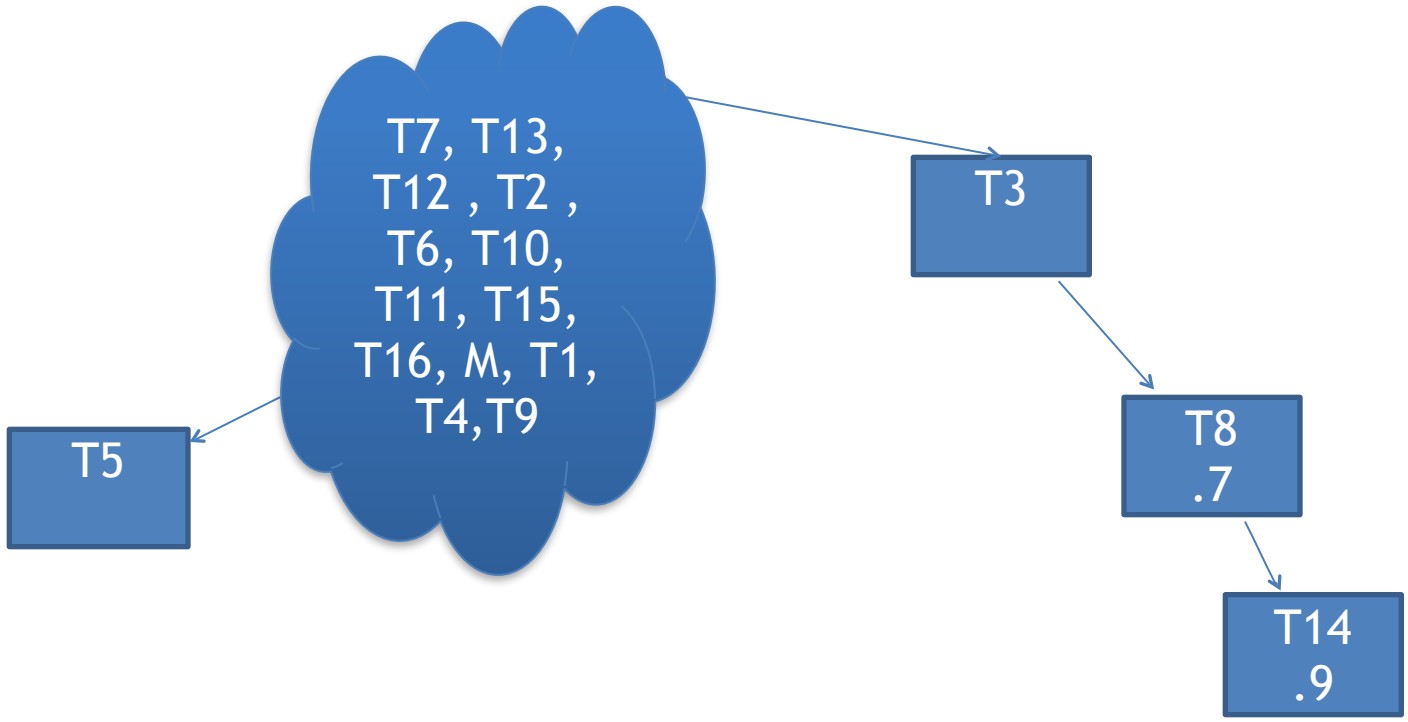
T7 -> T13



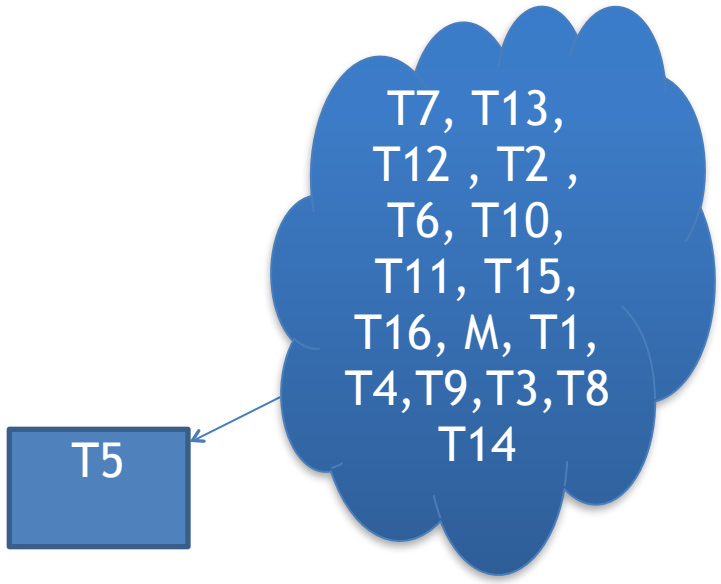
T7 -> T13



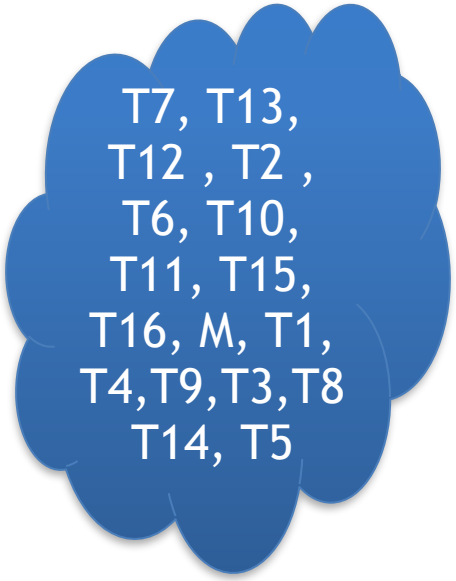
T7 -> T13



T7 -> T13



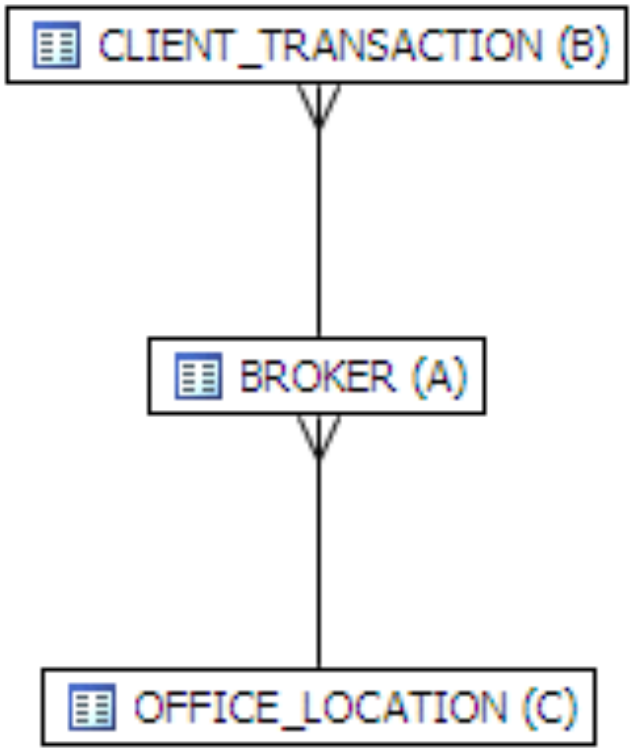
T7 -> T13



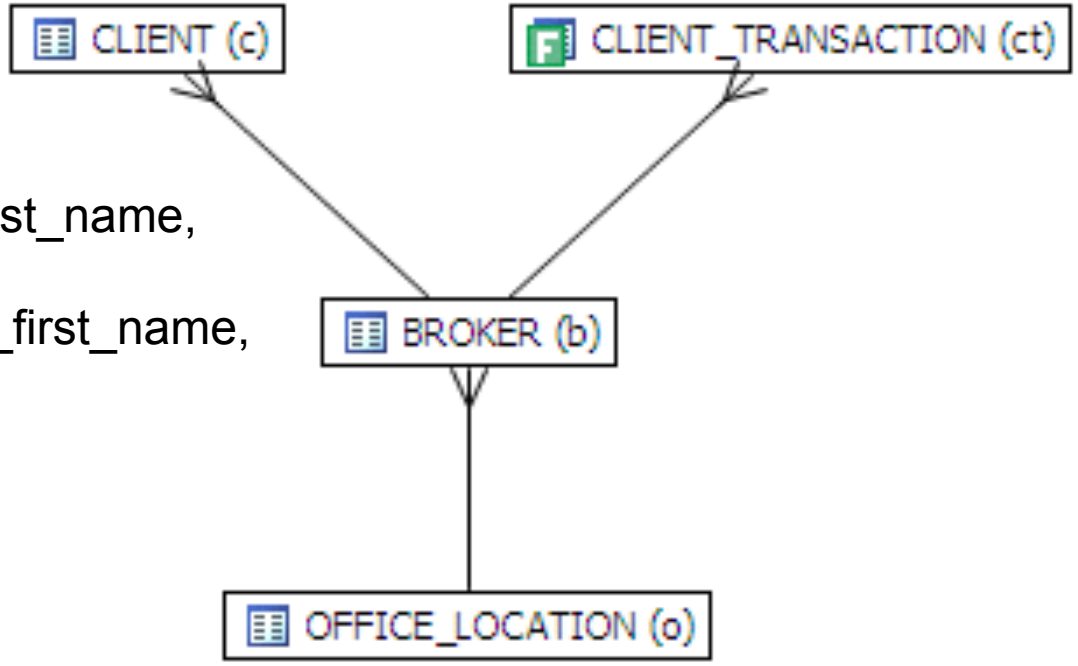
Why not Guess ?
For 17 table join, there are 355 Trillion combinations

Cartesian

```
SELECT
    A.BROKER_ID BROKER_ID,
    A.BROKER_LAST_NAME BROKER_LAST_NAME,
    A.BROKER_FIRST_NAME BROKER_FIRST_NAME,
    A.YEARS_WITH_FIRM YEARS_WITH_FIRM,
    C.OFFICE_NAME OFFICE_NAME,
    SUM (B.BROKER_COMMISSION)
TOTAL_COMMISSIONS
FROM
    BROKER A,
    CLIENT_TRANSACTION B,
    OFFICE_LOCATION C,
    INVESTMENT I
WHERE
    A.BROKER_ID = B.BROKER_ID AND
    A.OFFICE_LOCATION_ID = C.OFFICE_LOCATION_ID
GROUP BY
    A.BROKER_ID,
    A.BROKER_LAST_NAME,
    A.BROKER_FIRST_NAME,
    A.YEARS_WITH_FIRM,
    C.OFFICE_NAME;
```



Implied Cartesian




```
select
     c.client_first_name, c.client_last_name,
     ct.action, ct.price,
     b.broker_last_name, b.broker_first_name,
     o.office_name
from
      client_transaction ct,
      client c,
      broker b,
      office_location o
where
      ct.price > 100
  and b.broker_id=ct.broker_id
  and c.broker_id = b.broker_id
  and o.office_location_id = b.office_location_id
```

Diagram work for Many to One

One to many

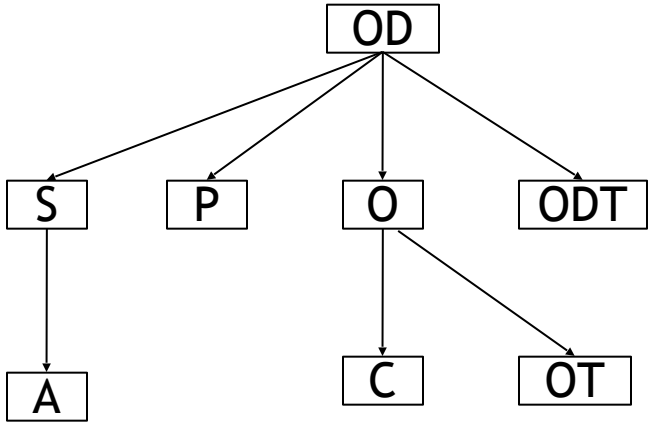
Many to many

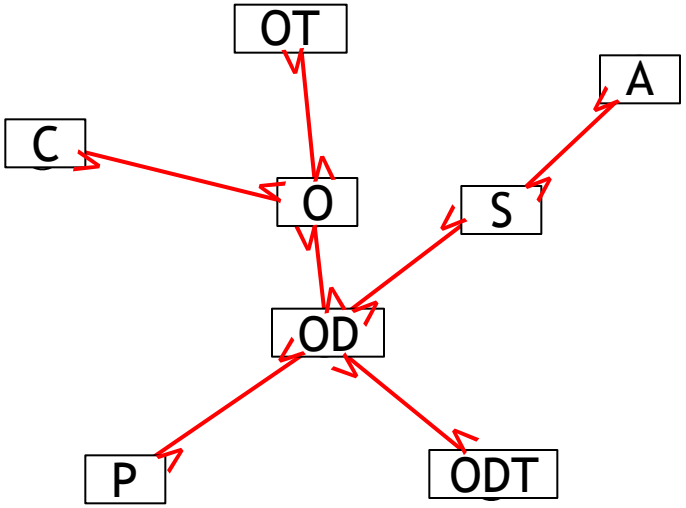
What about many to many?

Unstructured

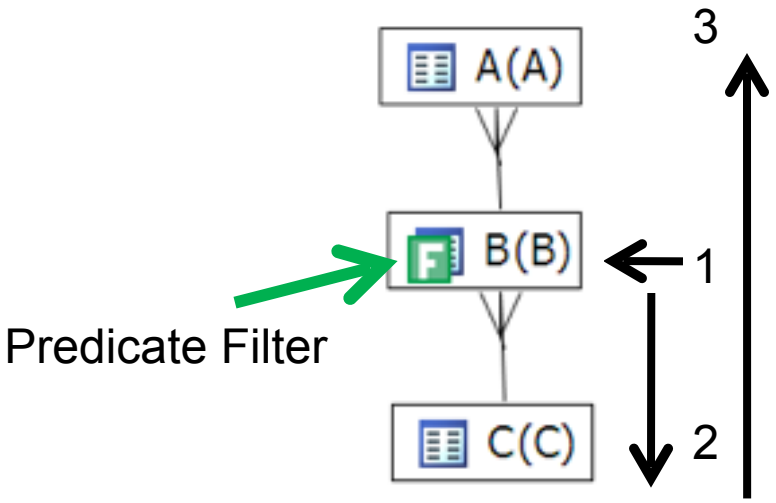
Joins

```
OD.Order_ID = O.Order_ID
O.Customer_ID = C.Customer_ID
OD.Product_ID = P.Product_ID(+)
OD.Shipment_ID = S.Shipment_ID(+)
S.Address_ID = A.Address_ID(+)
O.Status_Code = OT.Code
OD.Status_Code = ODT.Code
```

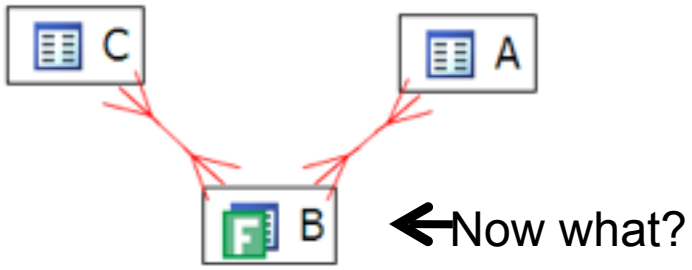




Many-to-One vs Many-to-Many



B -> C -> A

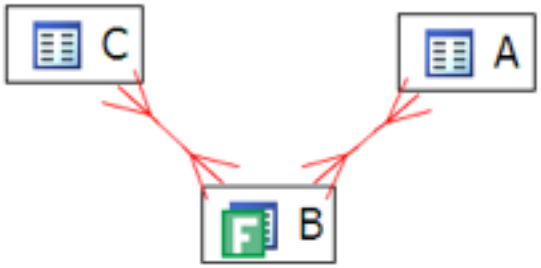


go to A or C?

Adding Constraints

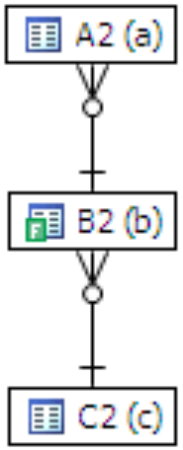
```
SELECT COUNT (*)
FROM
    b,
    c,
    a
WHERE
    b.val2 = 100 AND
    a.val1 = b.id AND
    b.val1 = c.id;
```

58 logical reads



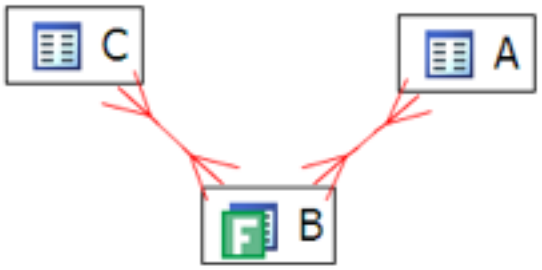

```
alter table c add constraint c_pk unique (id);
alter table b add constraint b_pk unique (id);
```

7 logical reads



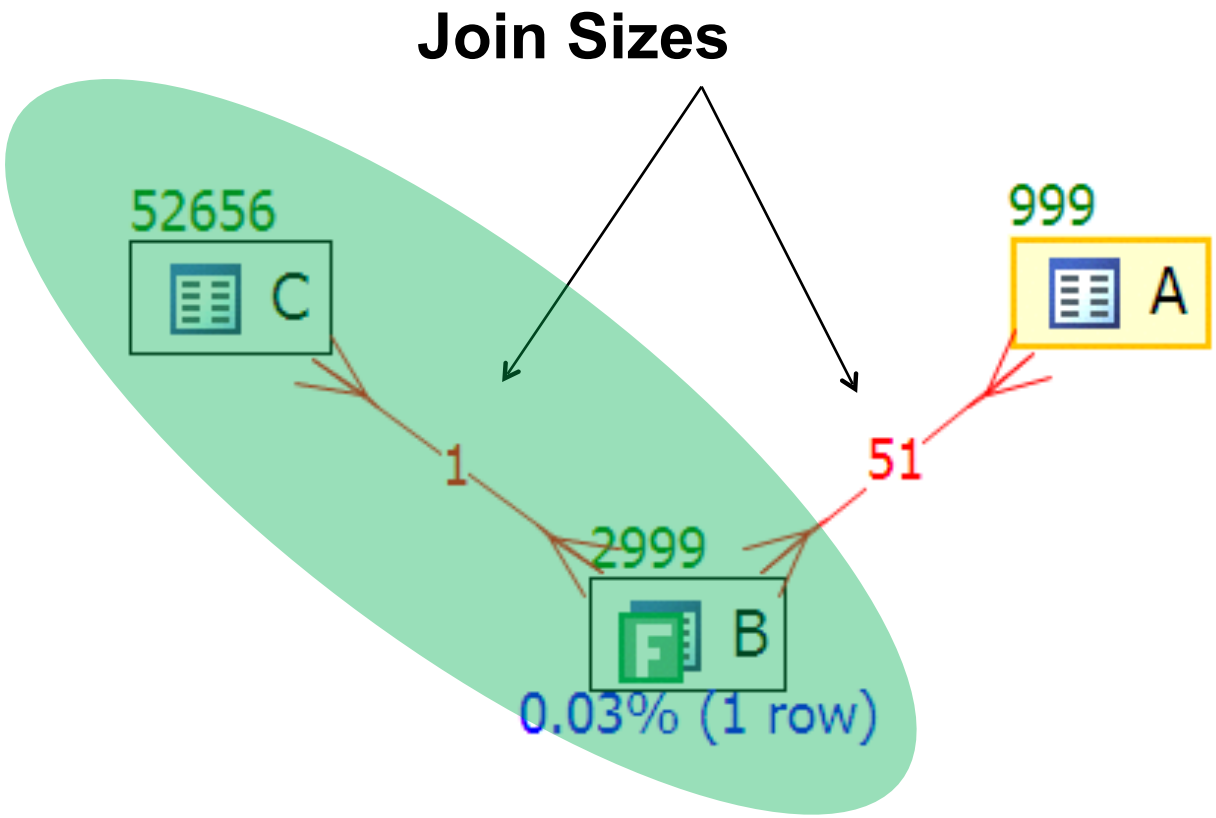
Diagraming: what to do with many to many

- Problem : with many-to-many don't know where do we go



- Solution: Add two table join result set sizes

Join sizes

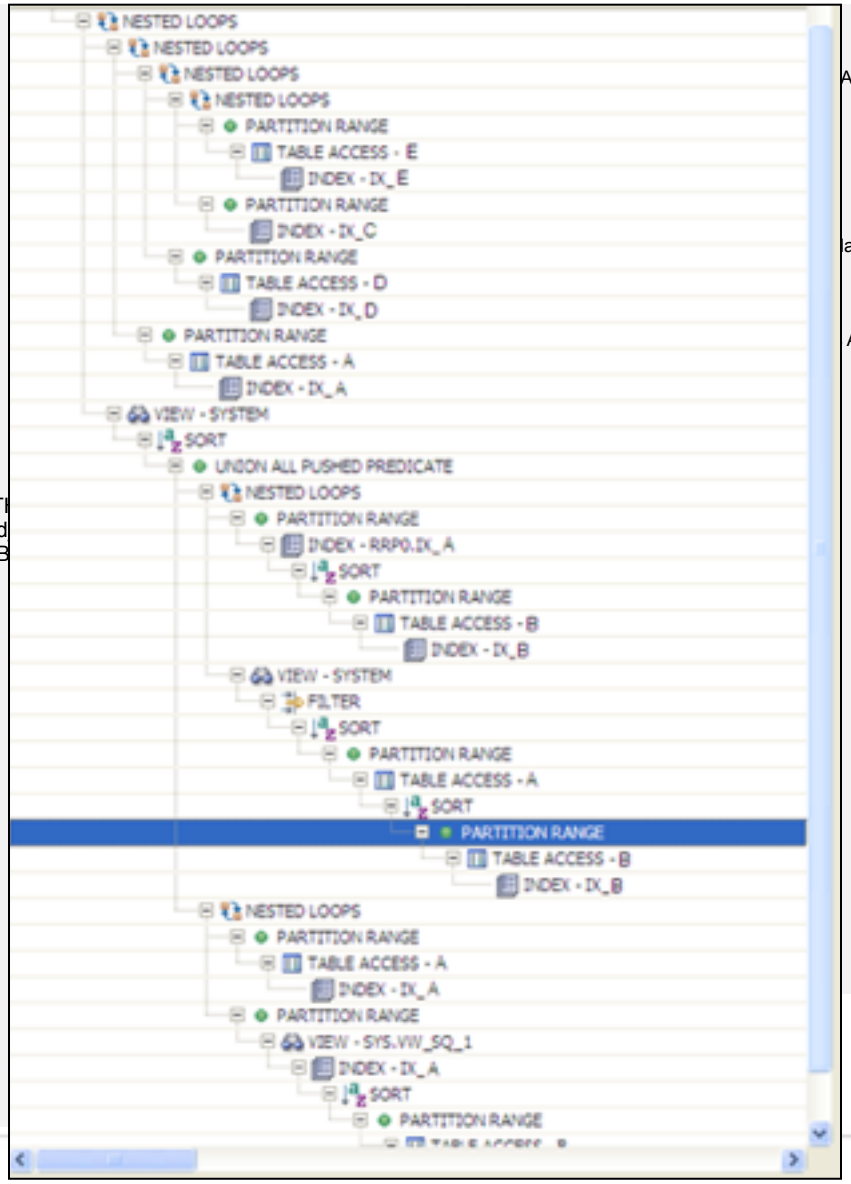


Look at 3 queries

- Query 1 runs more than 24 hours
- Query 2 outer joins and scalar sub-queries
- Query 3 create path not available to Oracle

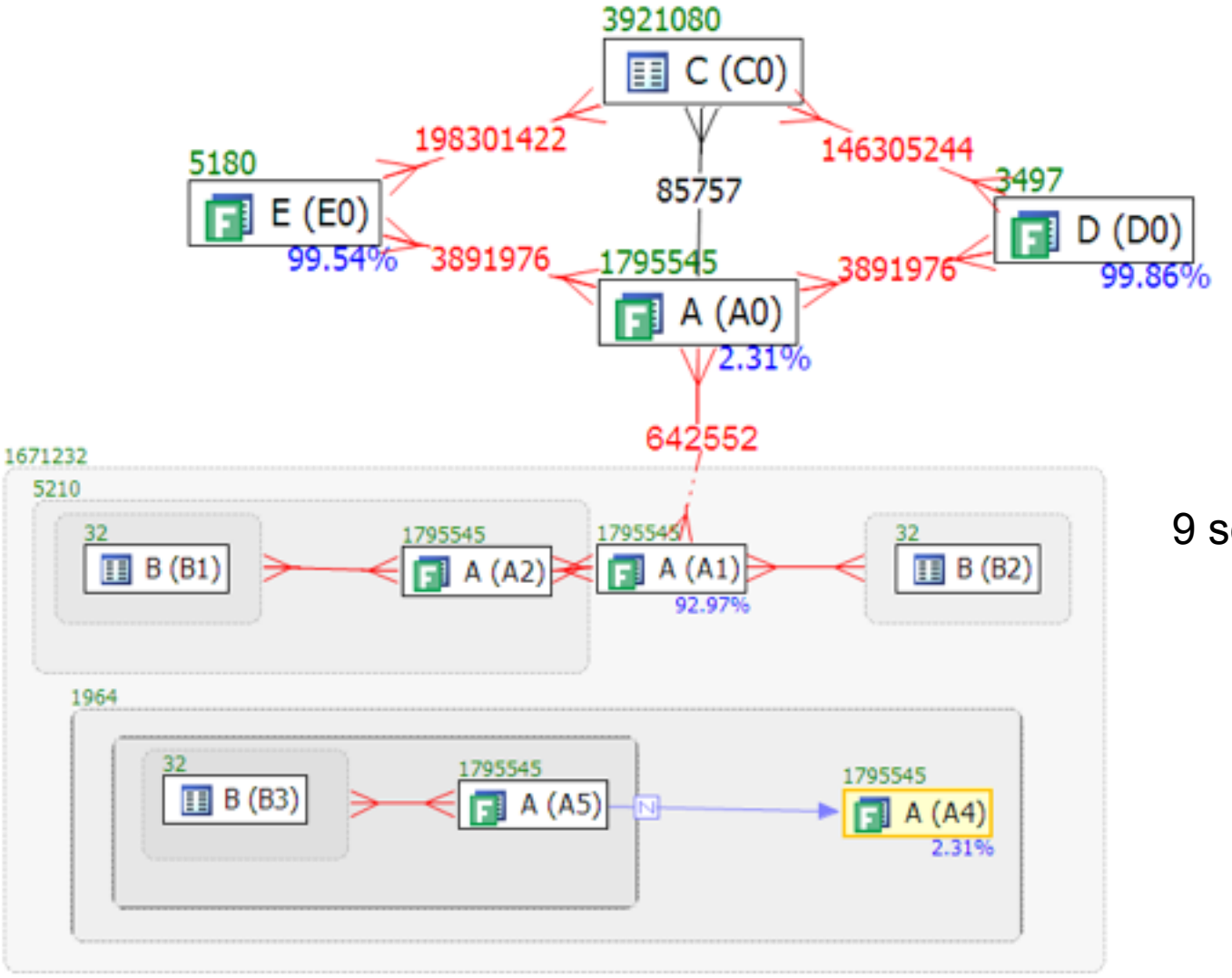
Query 1 : Over 24 hours to run

```
SELECT
   A0.zuchinis,
   A0.brocoli,
   C0.Oranges
FROM
   (
      SELECT
         A1.planted_date,
         A1.pears,
         A1.zuchinis,
         A1.brocoli
      FROM
         FOO.A A1,
         (
            SELECT
               zuchinis,
               brocoli
            FROM FOO.A A2
            WHERE
               pears = 'M' AND
               planted_date + 0 >= ADD_MONTH
                                    MAX (planted
                                 FROM FOO.B B
                                 WHERE
                                    pears = 'M'
         ),
                                  - 11)
            GROUP BY
               zuchinis,
               brocoli
            HAVING COUNT (*) = 12
         )
         i2
      WHERE
         A1.planted_date = (SELECT
                              MAX (planted_date)
                           FROM FOO.B B2
                           WHERE
                              pears = 'M'
      ) AND
         A1.pears = 'M' AND
         A1.zuchinis = i2.zuchinis (+) AND
         A1.brocoli = i2.brocoli (+)
 UNION
         SELECT
            A4.planted_date,
```



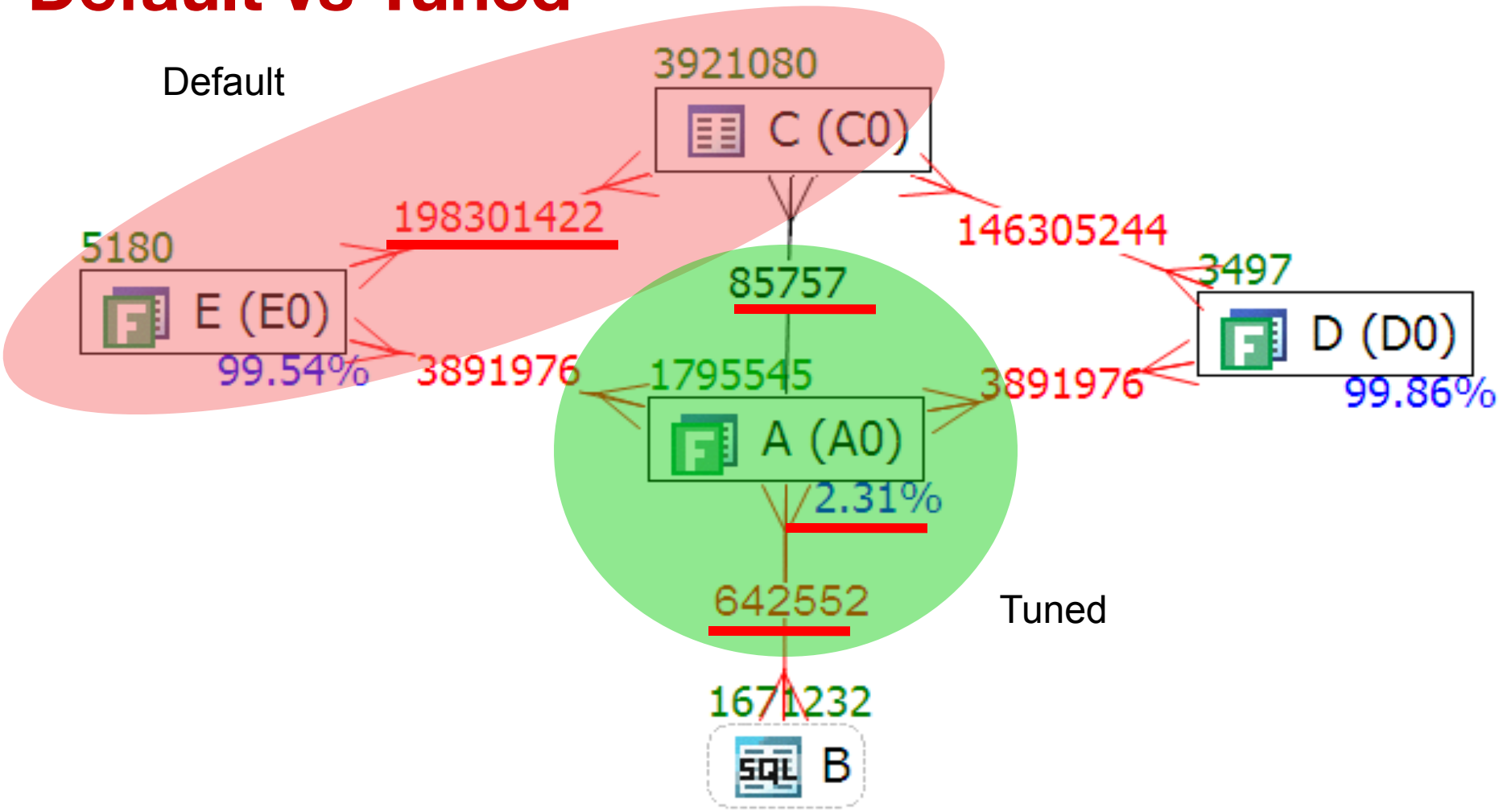

```
A4.planted_date <'03-OCT-08' AND

ate)

 AND
```

Visual SQL Diagram



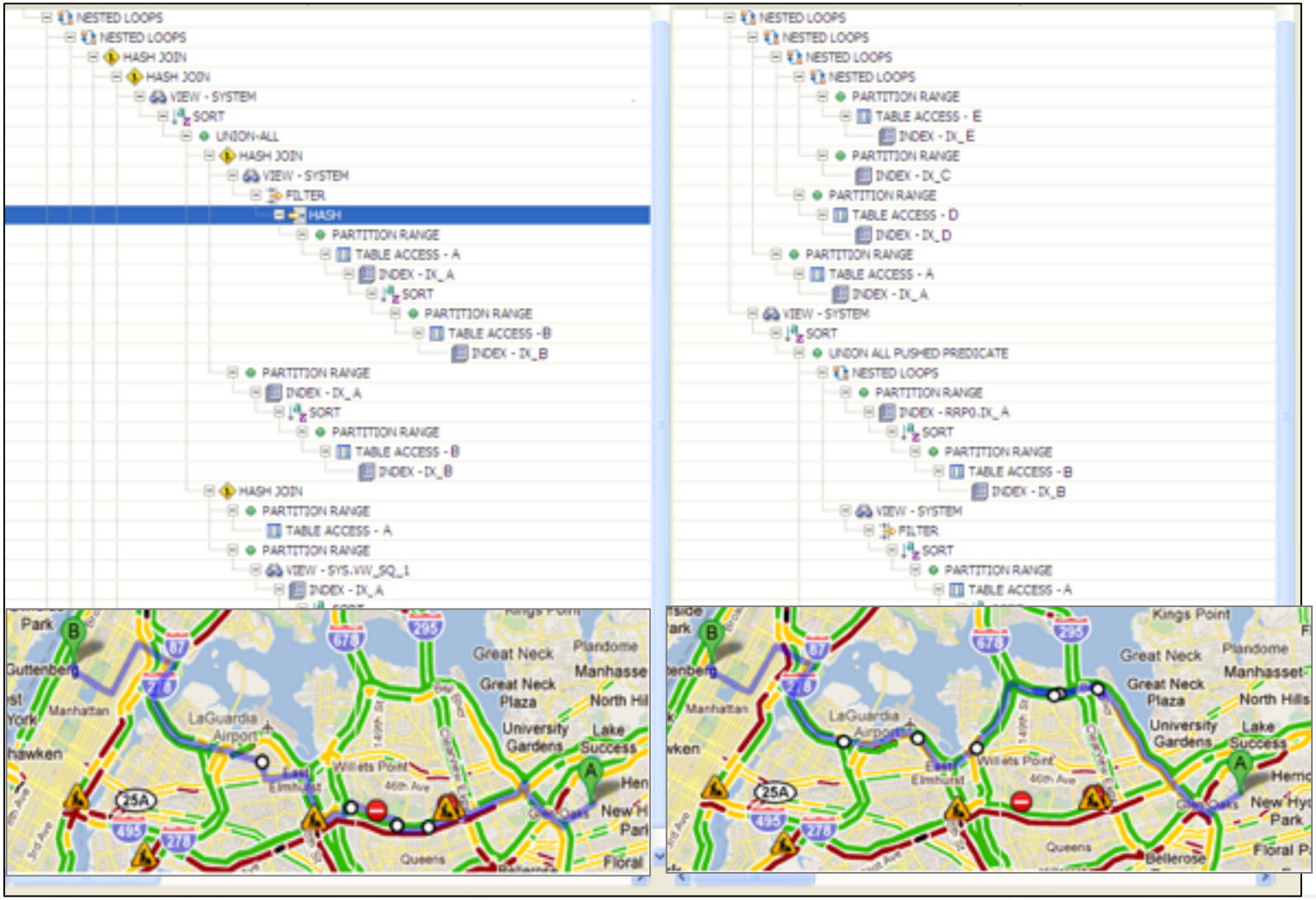
9 secs

Default vs Tuned

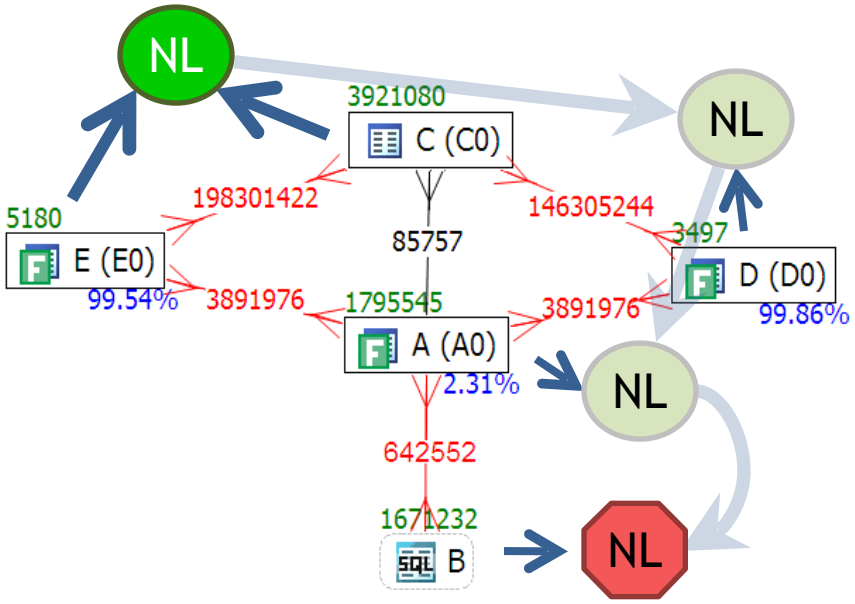


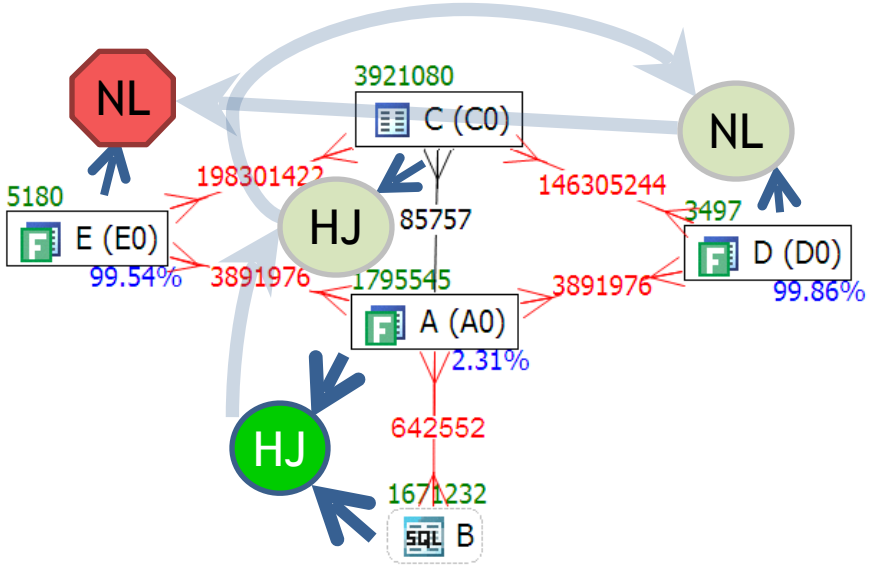
DELPHIX

Where is the map?

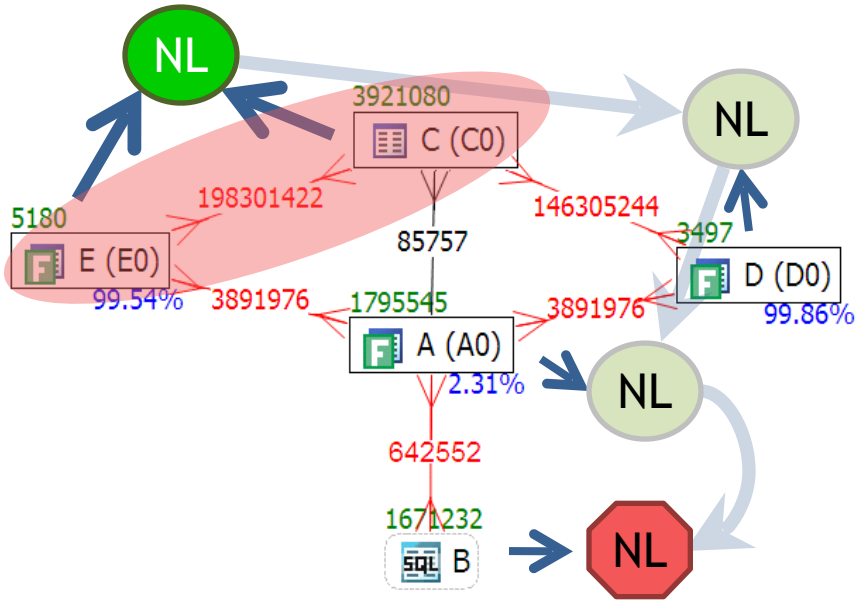


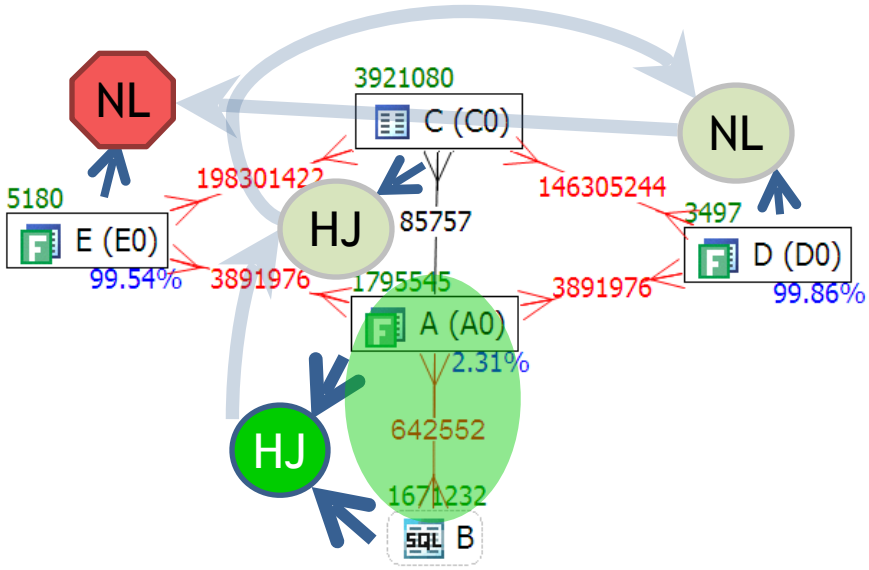
Comparing Plans : 24 hours to 5 mins





Comparing Plans : 24 hours to 5 mins

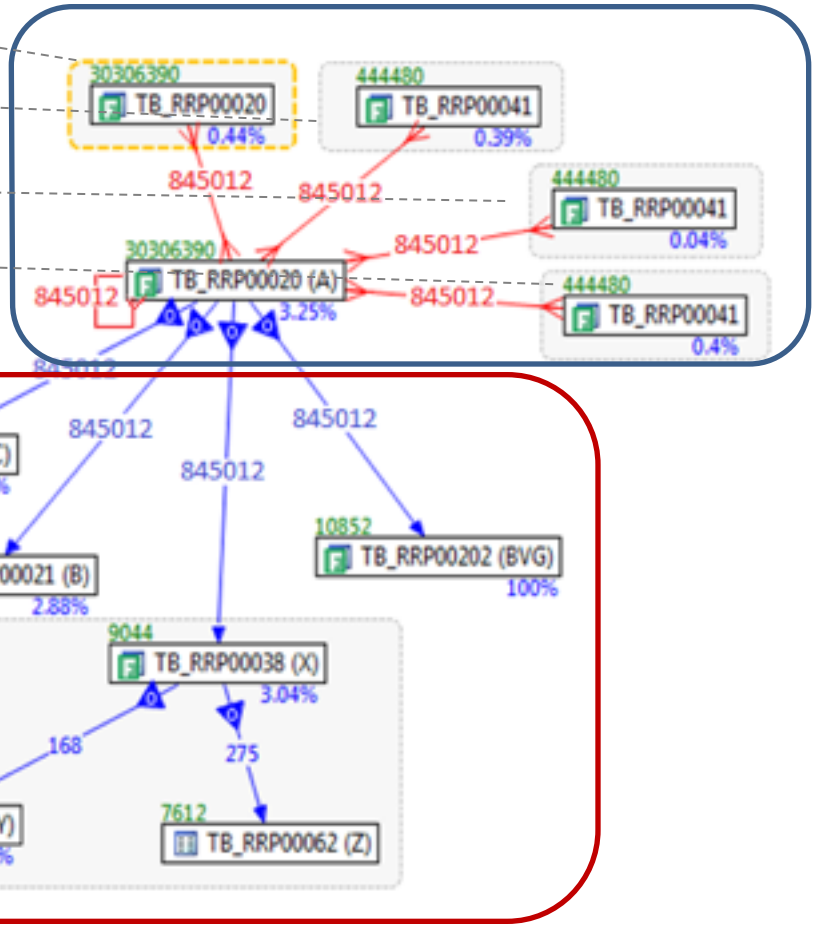




Q2

4 Scalar sub-queries
In the select clause

```
SELECT CASE WHEN M.NYC IS NULL THEN (SELECT /*+ qb_name(qb1) */ MAX (Kona)
                FROM foo.F
                WHERE harvest_date = to_date('08/10/2008','dd/mm/yyyy')
                 AND Argentina = TRIM ('D') AND Norway = F_OUTER.Norway
            ELSE M.NYC END AS NYC,
        CASE WHEN F_OUTER.Perth IS NULL THEN NULL
            ELSE (SELECT /*+ qb_name(qb2) */ Georgia FROM foo.P
                WHERE harvest_date = to_date('08/10/2008','dd/mm/yyyy')
                 AND Argentina = TRIM ('D') AND Paris = F_OUTER.Perth)
            END AS richard,
        CASE WHEN F_OUTER.Aruba IS NULL THEN NULL
            ELSE (SELECT /*+ qb_name(qb3) */ Georgia FROM foo.P
                WHERE harvest_date = to_date('08/10/2008','dd/mm/yyyy')
                 AND Argentina = TRIM ('D') AND Paris = F_OUTER.Aruba)
            END AS Jody,
         CASE WHEN F_OUTER.Portland IS NULL THEN NULL
            ELSE (SELECT /*+ qb_name(qb4) */ Georgia FROM foo.P
                WHERE harvest_date = to_date('08/10/2008','dd/mm/yyyy')
                 AND Argentina = TRIM ('D') AND Paris = F_OUTER.Portland
            END AS Tom
    FROM foo.F F_OUTER, foo.M , foo.J , foo.N ,
(SELECT /*+ qb_name(qb5) */ H.SF, Oregon, H.Haiti, K.Bermuda, L.Denmar
          FROM (foo.H  LEFT OUTER JOIN foo.K
                    ON H.harvest_date = K.harvest_date
                   AND H.Argentina = K.Argentina AND H.SF = K.SF
                   AND K.Dallas = '001')
                    LEFT OUTER JOIN FOo.L
                        ON H.harvest_date = L.harvest_date
                       AND H.Argentina = L.Argentina AND H.SF = L.SF
                    WHERE H.harvest_date = to_date('08/10/2008','dd/mm/yyy
                     AND H.Argentina = TRIM ('D')) extra
        WHERE F_OUTER.harvest_date = M.harvest_date(+)
          AND F_OUTER.Argentina = M.Argentina(+)
          AND F_OUTER.Norway = M.Norway(+)
          AND M.Norway(+) = M.Texas(+)
          AND F_OUTER.harvest_date = to_date('08/10/2008','dd/mm/yyyy')
          AND F_OUTER.Argentina = TRIM ('D')
          AND M.harvest_date(+) = to_date('08/10/2008','dd/mm/yyyy')
          AND M.Argentina(+) = TRIM ('D')
          AND F_OUTER.Norway = F_OUTER.Hawaii
          AND F_OUTER.harvest_date = J.harvest_date(+)
          AND F_OUTER.Argentina = J.Argentina(+)
          AND F_OUTER.Norway = J.Texas(+)
          AND J.harvest_date(+) = to_date('08/10/2008','dd/mm/yyyy')
          AND J.Argentina(+) = TRIM ('D')
          AND F_OUTER.Iraq = extra.SF(+)
          AND F_OUTER.harvest_date = N.harvest_date(+)
          AND F_OUTER.Argentina = N.Argentina(+)
          AND F_OUTER.Norway = N.Hawaii(+)
          AND N.Jordon(+) = '0'
/
```



Scalar Sub-queries

Q2

The subqueries in the select clause look like

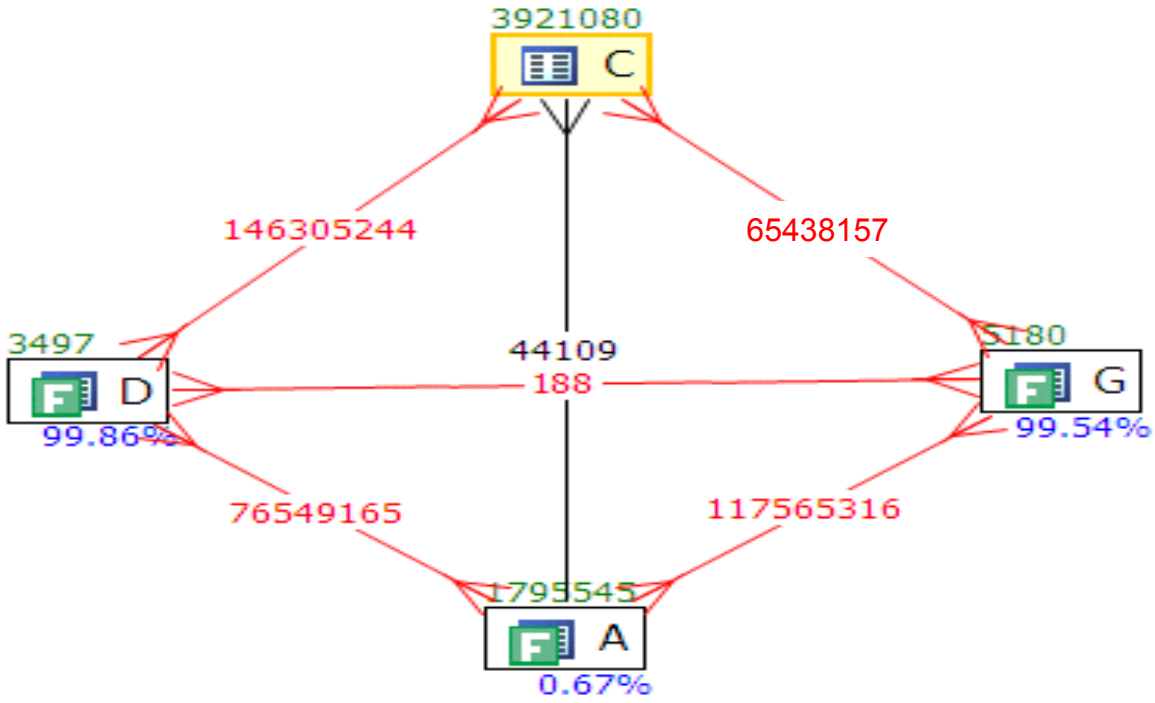
```
select CASE WHEN F.f1 IS NULL
            THEN NULL
            ELSE (SELECT X.f2
                  FROM X
                  WHERE code_vl = F.f2)
            END AS f0
from F;
```

and should be merged into the query like:

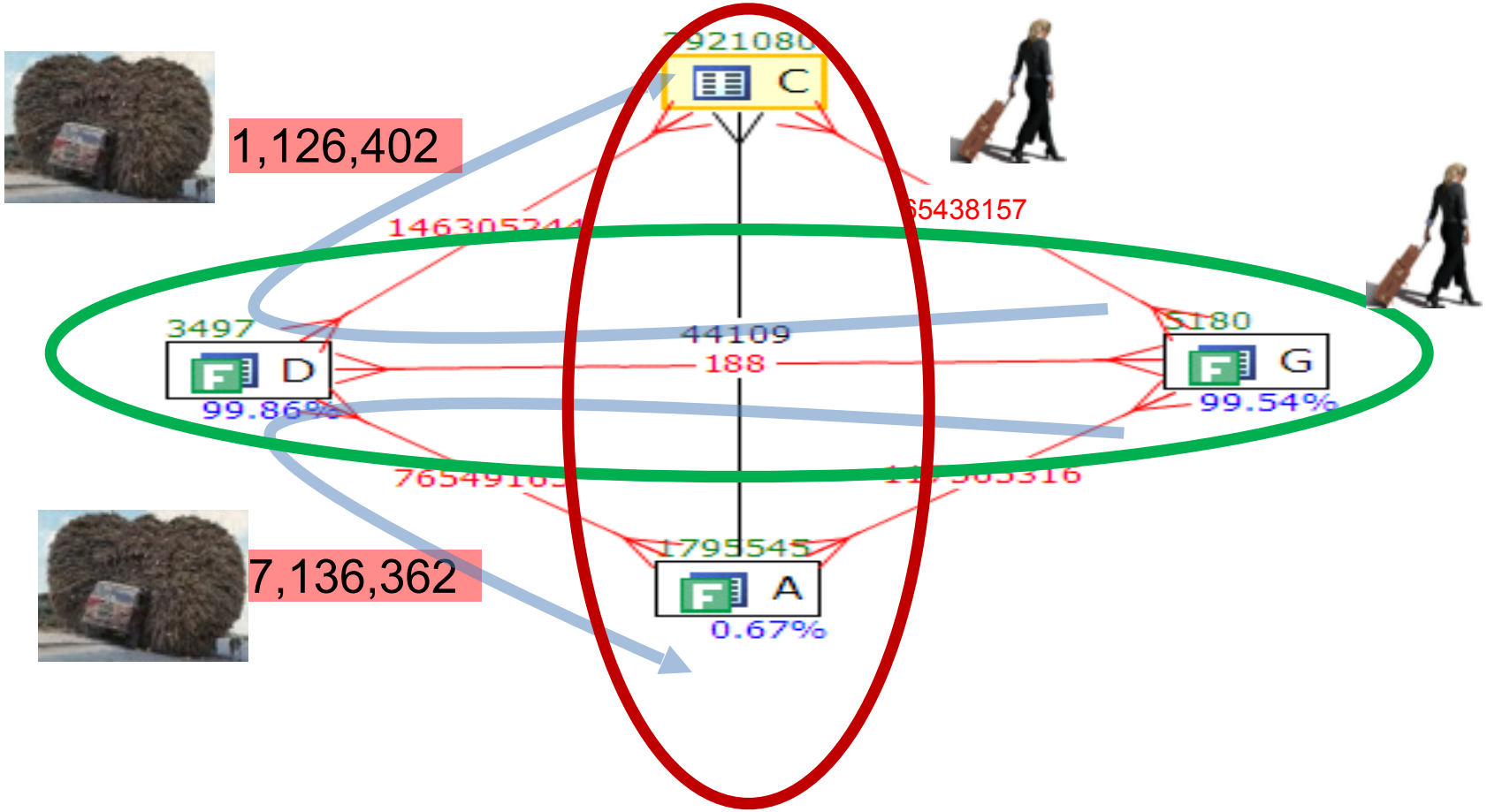
```
 select CASE WHEN F.f1 IS NULL
             THEN NULL
             ELSE ( X.f2)
             END AS f0
 from F , X
where code_vl(+) = decode(f.f1, null, null, F.F2)
```

Q3

```
SELECT DISTINCT *
FROM
    FOO.a a, FOO.c c, FOO.d d, FOO.g g
WHERE
    a.planted_date > '01-OCT-08' AND
    a.planted_date < '03-OCT-08' AND
    a.pears = 'D' AND a.green_beans = '1' AND
    a.planted_date = c.planted_date AND
    a.pears = c.pears AND
    a.zuchinis = c.zuchinis AND
    a.brocoli = c.brocoli AND
    a.planted_date = d.planted_date AND
    a.pears = d.pears AND
    a.harvest_size = d.harvest_size AND
    c.oranges = d.oranges AND
    c.apples = d.apples AND
    (d.lemons = 0 OR d.lemons IS NULL) AND
    a.planted_date = g.planted_date AND
    a.pears = g.pears AND
    a.harvest_size = g.harvest_size AND
    c.oranges = g.oranges AND
    c.apples = g.apples AND
    (g.lemons = 0 OR g.lemons IS NULL) AND
    a.zuchinis = '0236' AND
    d.apples = g.apples AND
    d.oranges = g.oranges
ORDER BY a.zuchinis, a.brocoli;
```

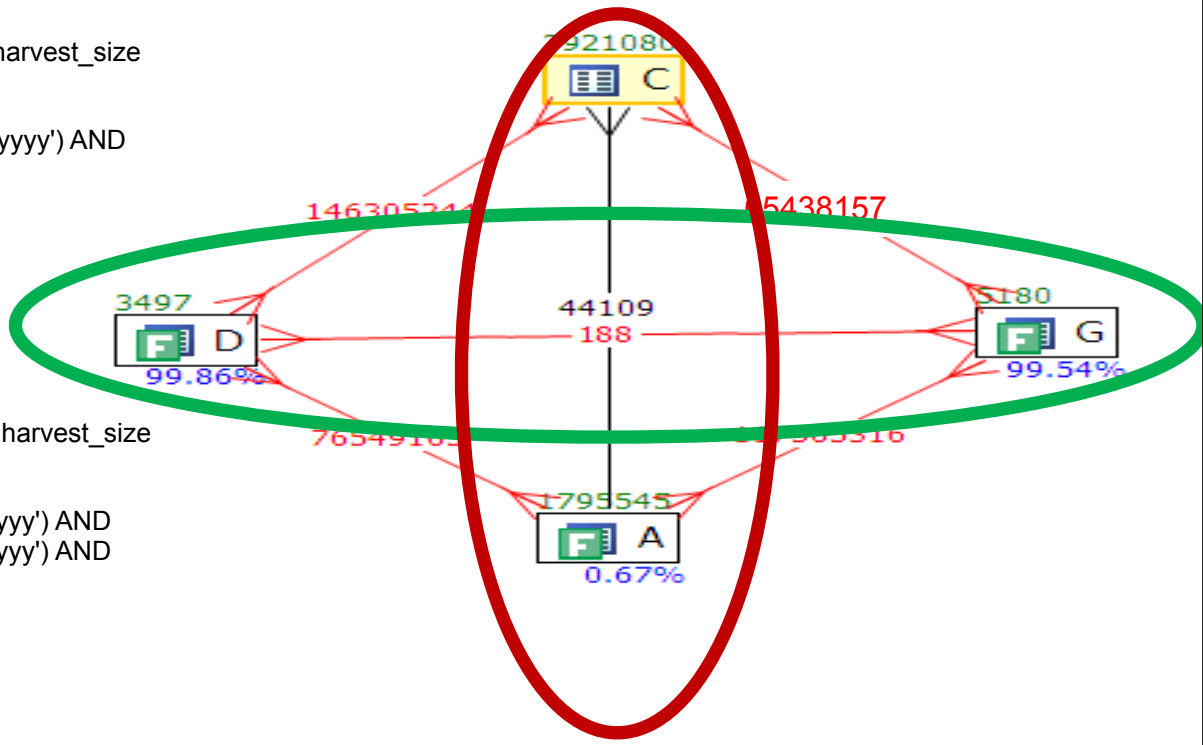


Q3: Transitivity



Q3

```
SELECT * FROM
(
   SELECT /*+  NO_MERGE */ c.apples, c.oranges, a.harvest_size
   FROM a,   c
   WHERE
      a.planted_date = TO_DATE ('02/10/2008', 'dd/mm/yyyy') AND
      a.pears = 'D' AND
      a.green_beans = '1' AND
      a.planted_date = c.planted_date AND
      a.pears = c.pears AND
      a.zuchinis = c.zuchinis AND
      a.brocoli = c.brocoli AND
      a.zuchinis = '0236'
) X,
(
    SELECT /*+  NO_MERGE */ d.apples, d.oranges, d.harvest_size
    FROM  d,  g
    WHERE
    d.planted_date = TO_DATE ('02/10/2008', 'dd/mm/yyyy') AND
    g.planted_date = TO_DATE ('02/10/2008', 'dd/mm/yyyy') AND
    g.apples = d.apples AND
    d.oranges = g.oranges AND
    d.pears = 'D' AND
    g.pears = 'D' AND
    g.pears = d.pears AND
    g.harvest_size = d.harvest_size AND
    (d.lemons = 0 OR d.lemons IS NULL) AND
    (g.lemons = 0 OR  g.lemons IS NULL)
) Y
WHERE
X.oranges = Y.oranges AND
X.apples = Y.apples AND
X.harvest_size = Y.harvest_size;
```

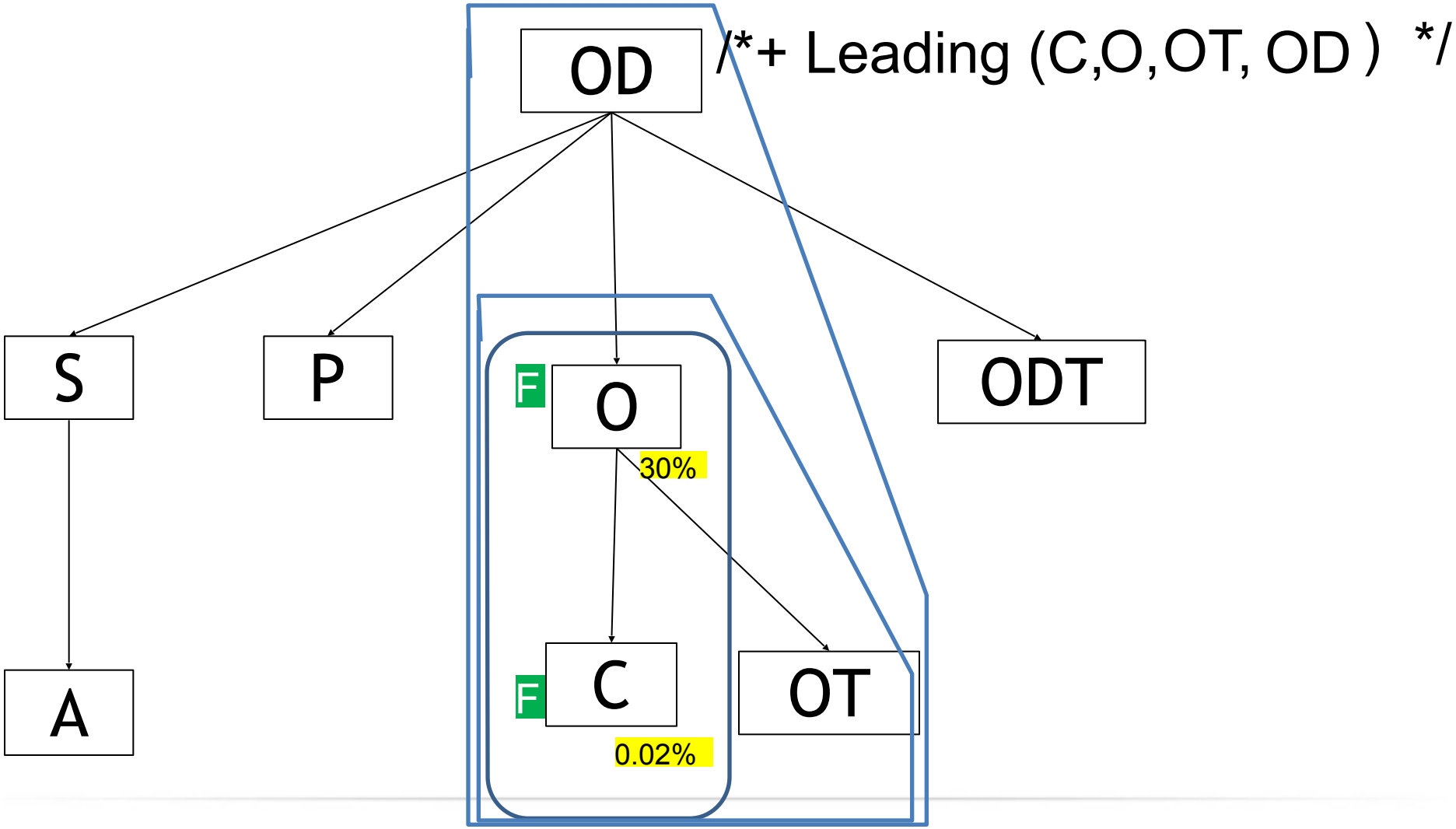


This final version runs in
elapsed **0.33 secs** and 12K logical reads
down from an original
elapsed **4.5 secs** and 1M logical reads

Related info

- Hints
  - Leading hint
  - HJ & NL
- Tools to use
  - graphical execution = visual SQL tuning (VST) diagram
- Example from Jonathan Lewis
- Join Ratios
  - Join detail ratio
  - Join master ratio

Leading Hint, use table alias (not table)



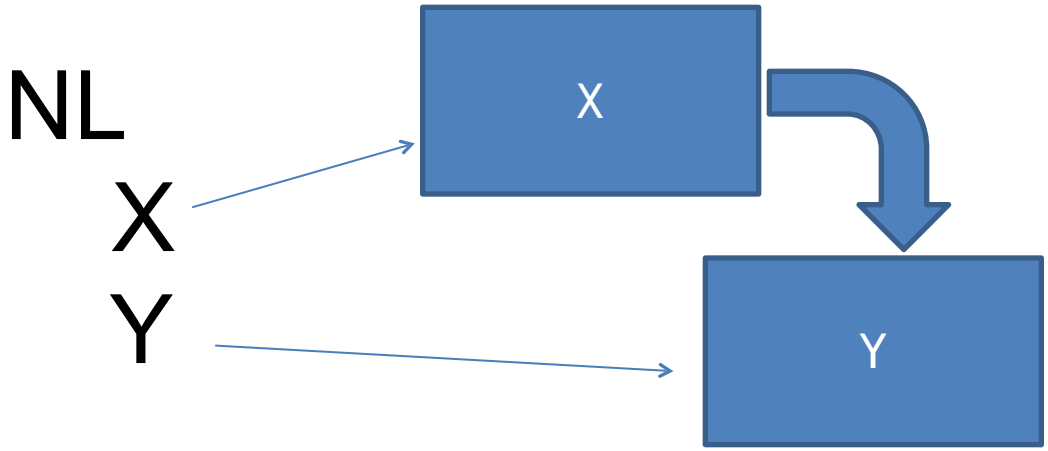
HINTS

- Leading (tab_alias , table_alias …) 10g (9i use ORDERED, not as good)
- USE_NL (table_alias) – 2nd in xplan probed into (Inner Table)
- USE_HASH (table_alias) – 1st in xplan probed into
- INDEX (tab_alias index_name)
- NO_MERGE

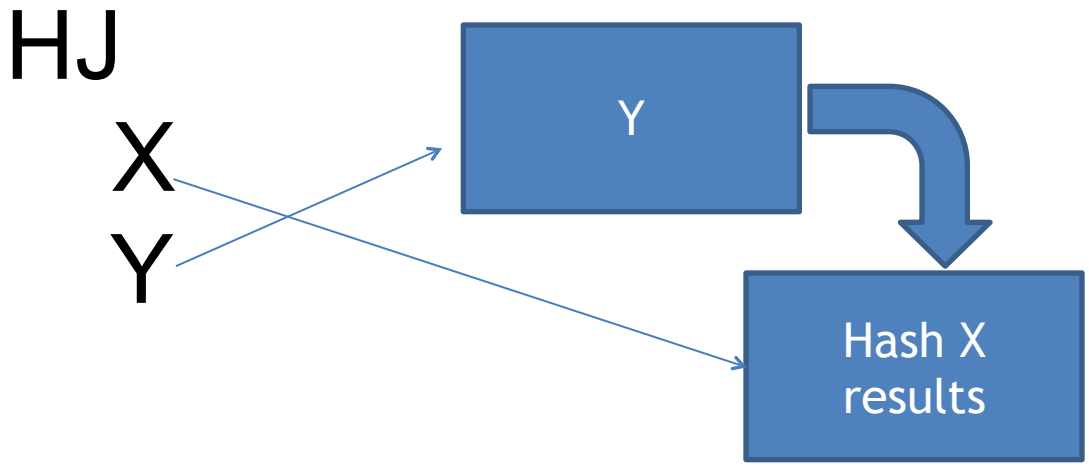
Oracle first decides join order then join type

(example http://www.adp-gmbh.ch/blog/2008/01/17.php)

select /*+ LEADING(X,**Y**) USE_NL(**Y**) */ *



select /*+ LEADING(X,**Y**) USE_HASH(**Y**) */ *

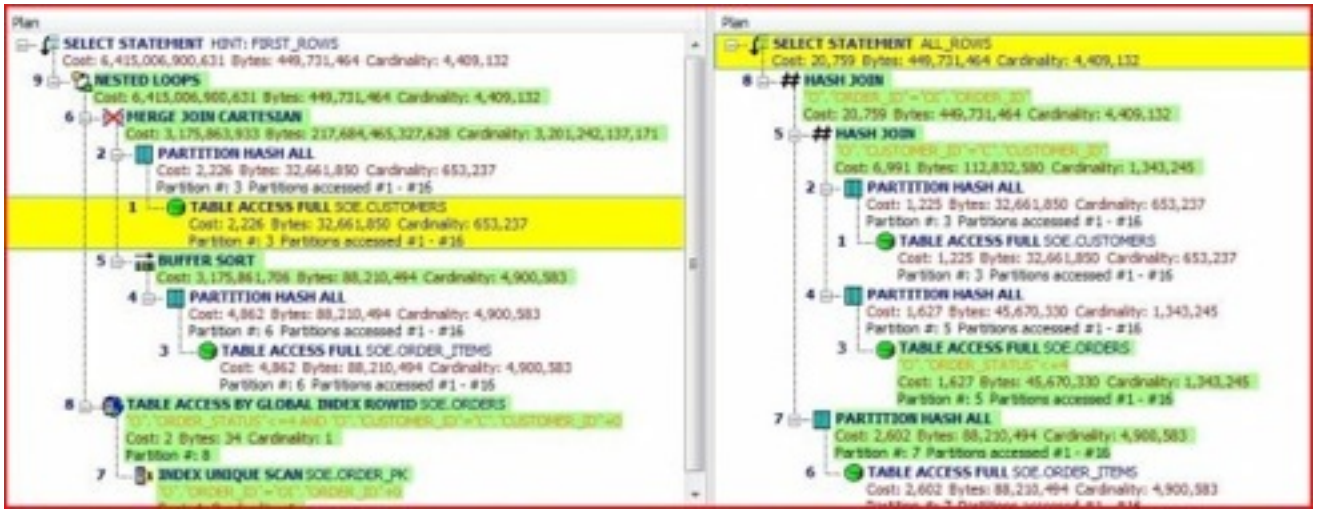


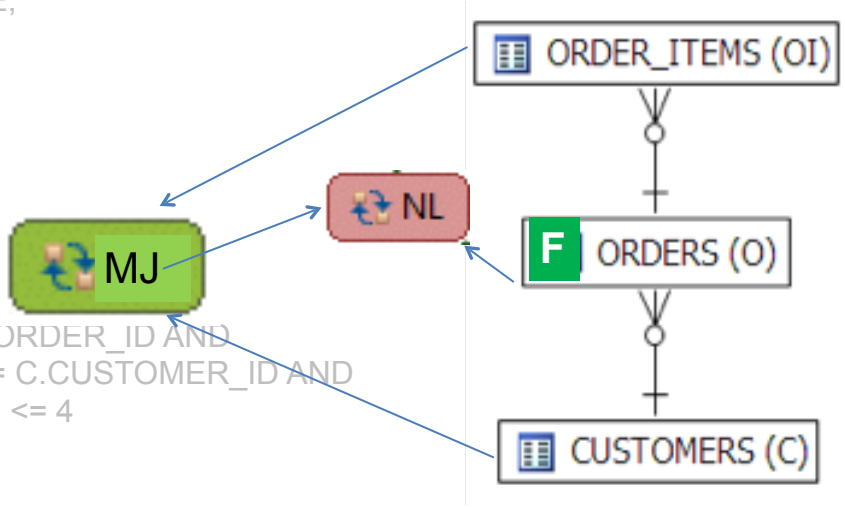
What tools to use to create VST diagrams?

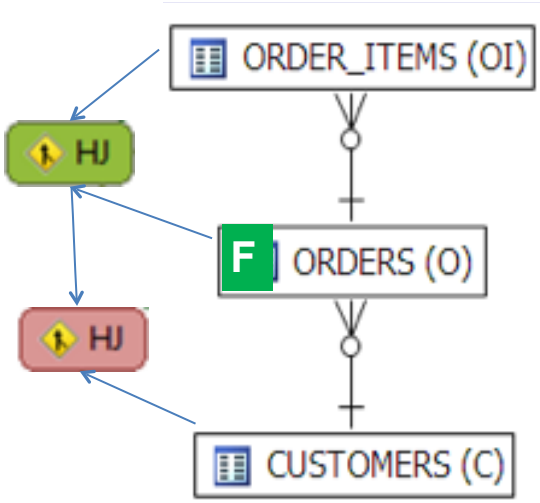
- Paper
  - Modifications messy and difficult
- PowerPoint
  - Can move things around
  - Sticky connectors
  - Easy to modify
- DB Optimizer
  - Automatic and still modifiable

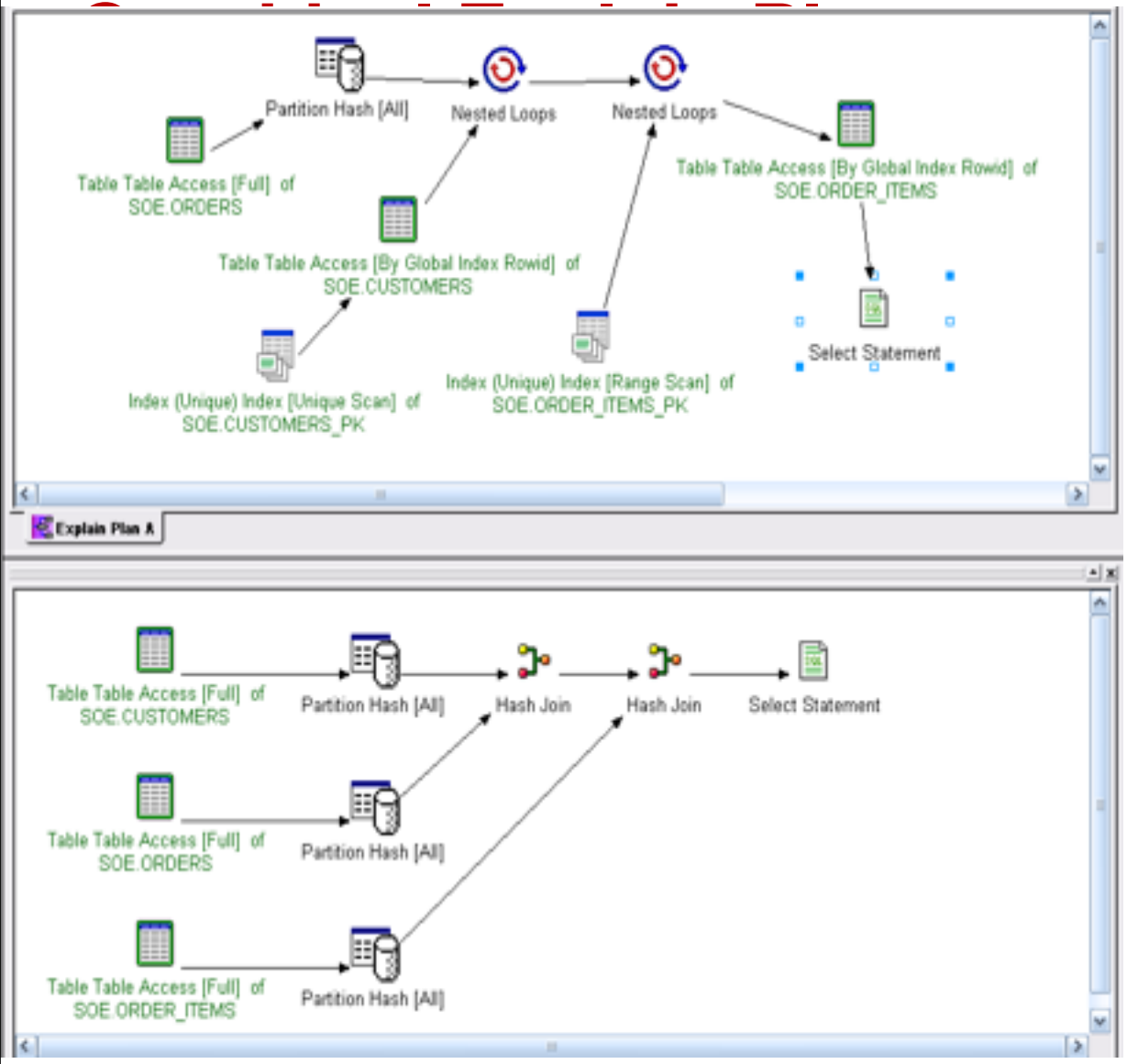
VST vs Explain Plan

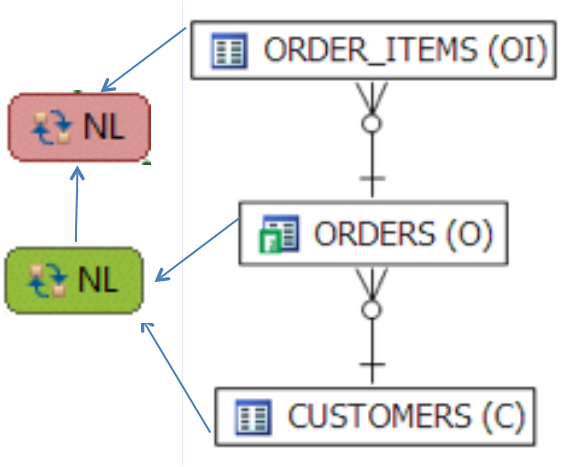
```
SELECT
    O.ORDER_ID,
    LINE_ITEM_ID,
    PRODUCT_ID,
    UNIT_PRICE,
    QUANTITY,
    ORDER_MODE,
    ORDER_STATUS,
    ORDER_TOTAL,
    SALES_REP_ID,
    PROMOTION_ID,
    C.CUSTOMER_ID,
    CUST_FIRST_NAME,
    CUST_LAST_NAME,
    CREDIT_LIMIT,
    CUST_EMAIL,
    ORDER_DATE
FROM
    ORDERS O,
    ORDER_ITEMS OI,
    CUSTOMERS C
WHERE
    O.ORDER_ID = OI.ORDER_ID AND
    O.CUSTOMER_ID = C.CUSTOMER_ID AND
    O.ORDER_STATUS <= 4
```

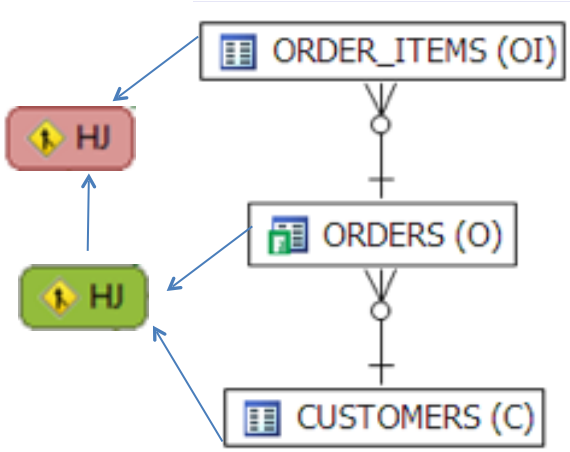







Partition Hash [All]
Nested Loops
Nested Loops
Table Table Access [Full] of SOE.ORDERS
Table Table Access [By Global Index Rowid] of SOE.CUSTOMERS
Table Table Access [By Global Index Rowid] of SOE.ORDER_ITEMS
Index (Unique) Index [Unique Scan] of SOE.CUSTOMERS_PK
Index (Unique) Index [Range Scan] of SOE.ORDER_ITEMS_PK
Select Statement
Explain Plan A
Table Table Access [Full] of SOE.CUSTOMERS
Partition Hash [All]
Hash Join
Hash Join
Select Statement
Table Table Access [Full] of SOE.ORDERS
Partition Hash [All]
Table Table Access [Full] of SOE.ORDER_ITEMS
Partition Hash [All]


ORDER_ITEMS (OI)
NL
NL
ORDERS (O)
CUSTOMERS (C)

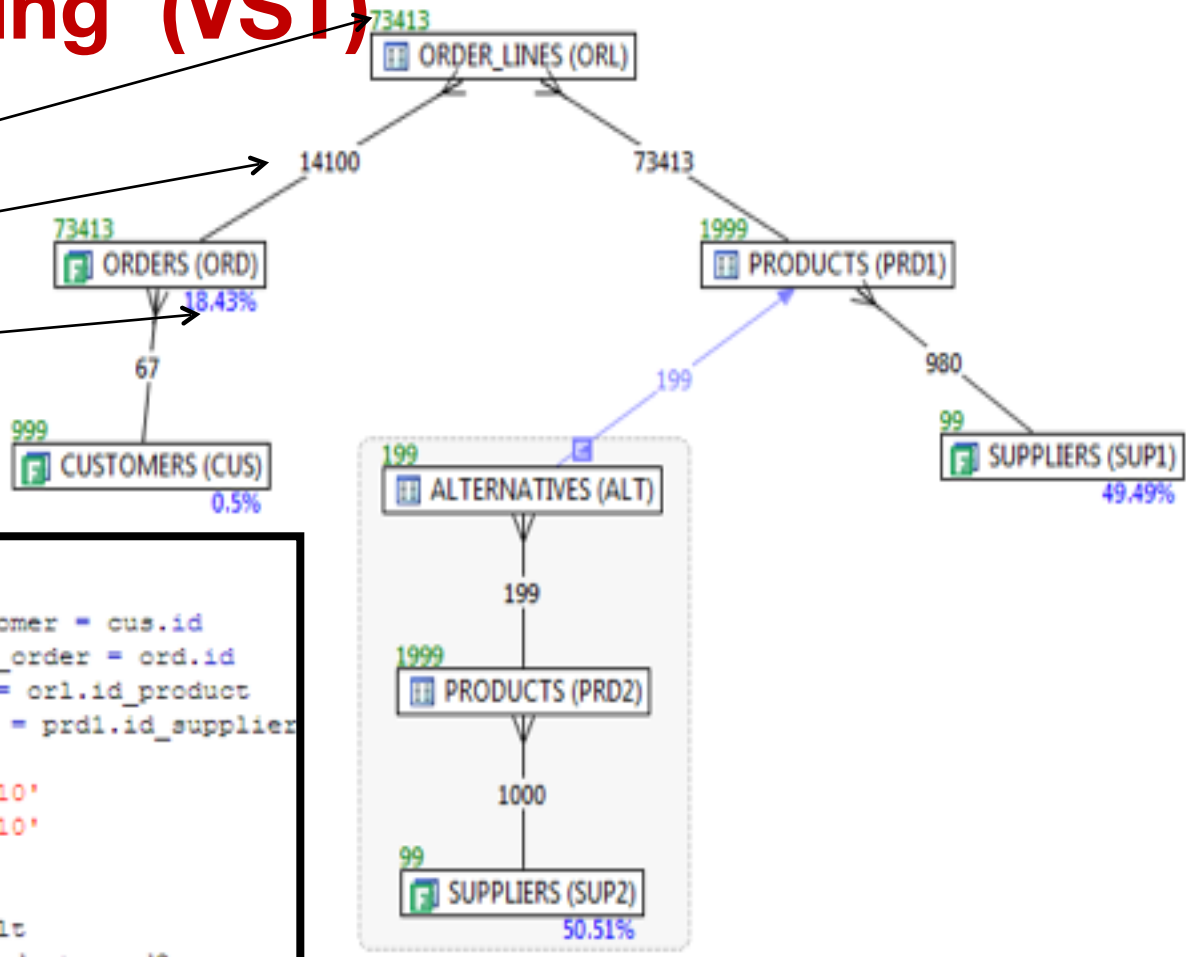

ORDER_ITEMS (OI)
HJ
HJ
ORDERS (O)
CUSTOMERS (C)

Visual SQL Tuning (VST)

Table Sizes

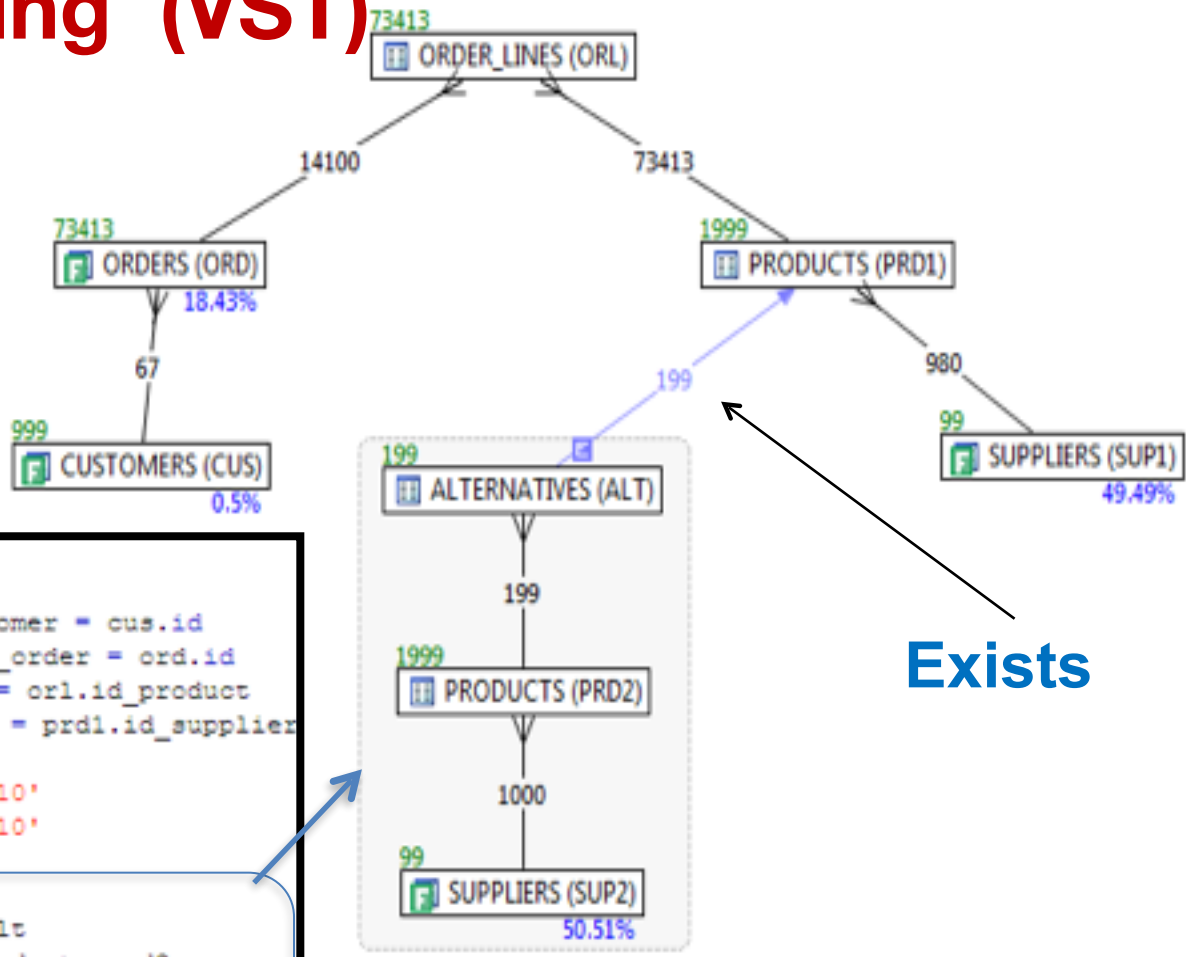
Join Sizes

Filter Ratios



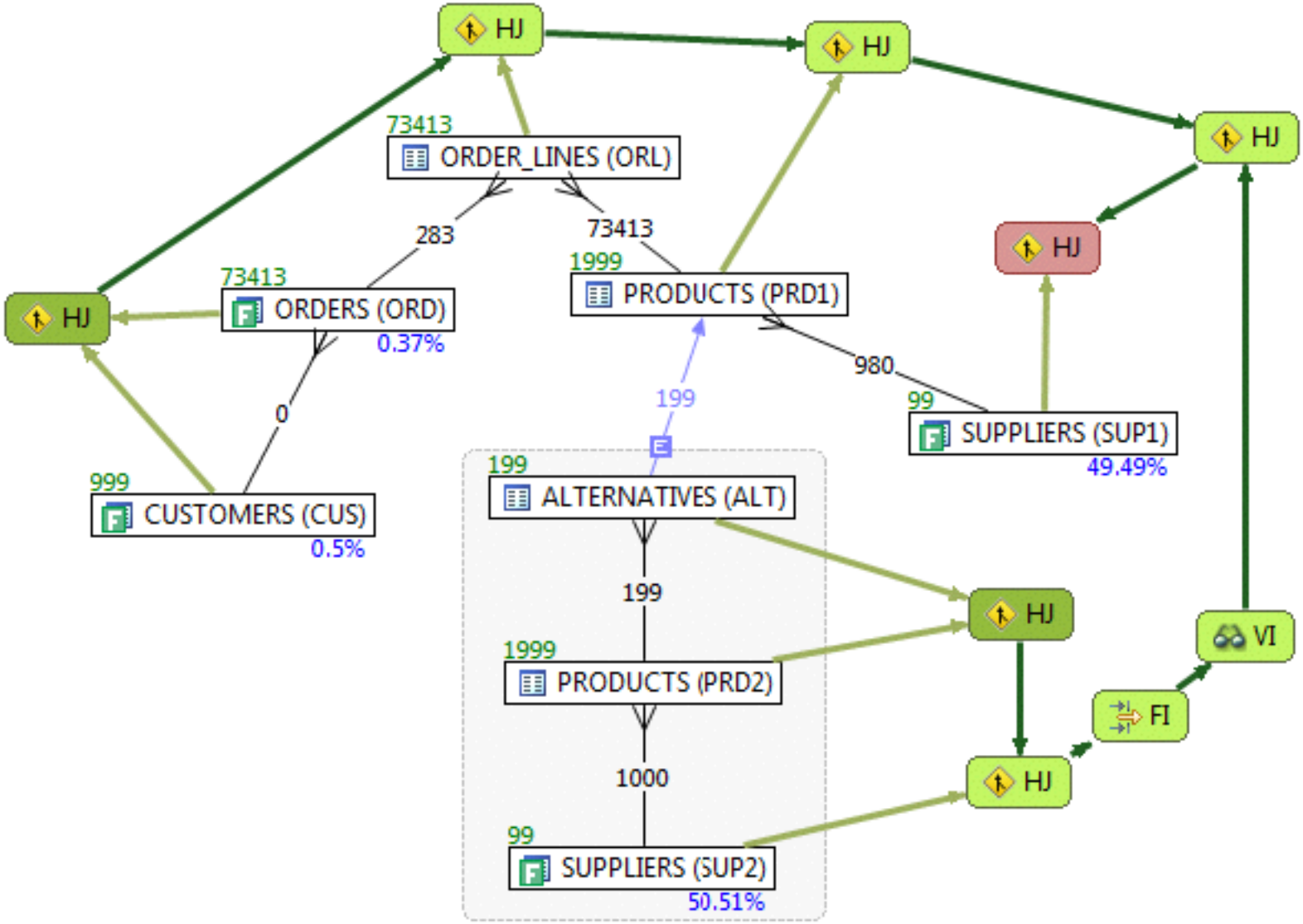

```
SELECT order_line_data
FROM           customers cus
    INNER JOIN orders ord ON ord.id_customer = cus.id
    INNER JOIN order_lines orl ON orl.id_order = ord.id
    INNER JOIN products prd1 ON prd1.id = orl.id_product
    INNER JOIN suppliers sup1 ON sup1.id = prd1.id_supplier
WHERE   cus.location = 'LONDON'
    AND ord.date_placed BETWEEN '04-JUN-10'
                        AND     '11-JUN-10'
    AND sup1.location = 'LEEDS'
    AND EXISTS ( SELECT  NULL
                 FROM  alternatives    alt
                       INNER JOIN     products prd2
                         ON prd2.id = alt.id_product_sub
                       INNER JOIN     suppliers sup2
                         ON sup2.id = prd2.id_supplier
                 WHERE    alt.id_product = prd1.id
                       AND sup2.location != 'LEEDS' )
```

Visual SQL Tuning (VST)

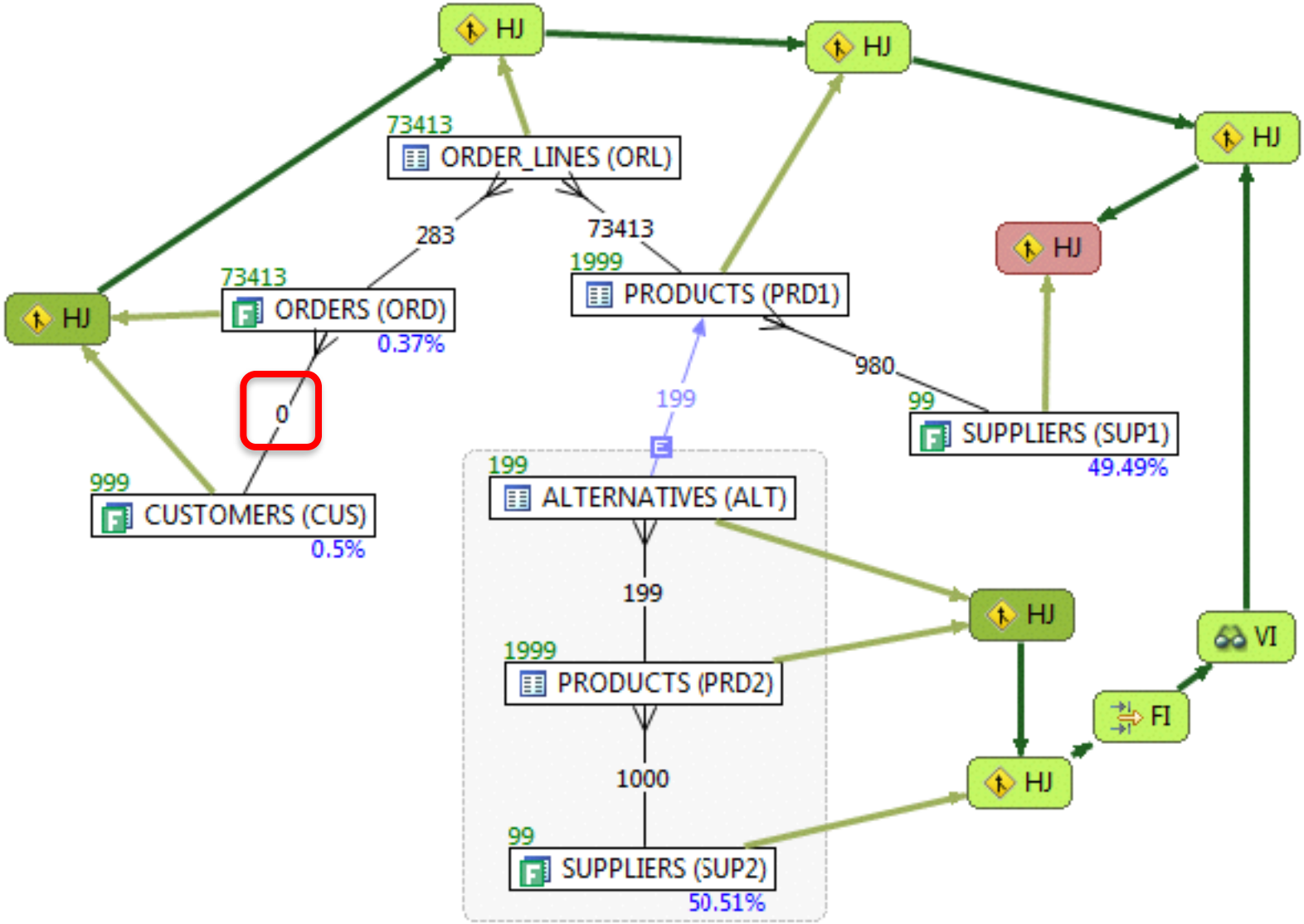



```
SELECT order_line_data
FROM           customers cus
    INNER JOIN orders ord ON ord.id_customer = cus.id
    INNER JOIN order_lines orl ON orl.id_order = ord.id
    INNER JOIN products prd1 ON prd1.id = orl.id_product
    INNER JOIN suppliers sup1 ON sup1.id = prd1.id_supplier
WHERE   cus.location = 'LONDON'
    AND ord.date_placed BETWEEN '04-JUN-10'
                        AND     '11-JUN-10'
    AND sup1.location = 'LEEDS'
    AND EXISTS ( SELECT  NULL
                 FROM  alternatives    alt
                       INNER JOIN     products prd2
                         ON prd2.id = alt.id_product_sub
                       INNER JOIN     suppliers sup2
                         ON sup2.id = prd2.id_supplier
                 WHERE    alt.id_product = prd1.id
                       AND sup2.location != 'LEEDS' )
```

Execution Plan



Execution Plan



Join Filter ratios

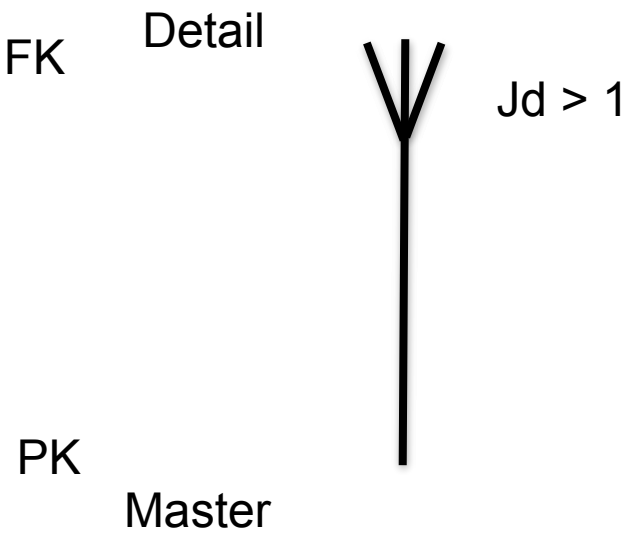
Detail

Master

Jd “Join detail ratio”
= avg #rows returned by joining one row from master

Jm “Join master ratio”
= avg #rows returned by joining one row from Detail

Join Filter ratios

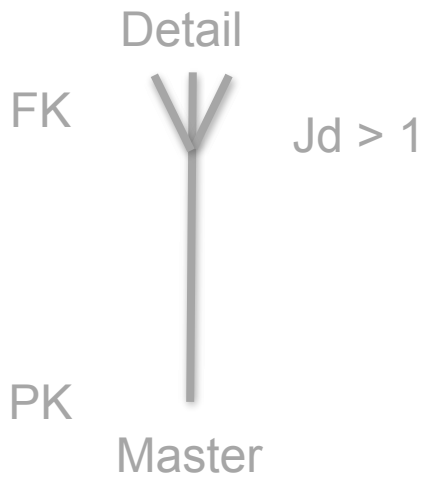


Join Detail Jd > 1 normally

If Jd < 1 joining acts like a filter, unusual

If Jd = 0 then good place to start, rare

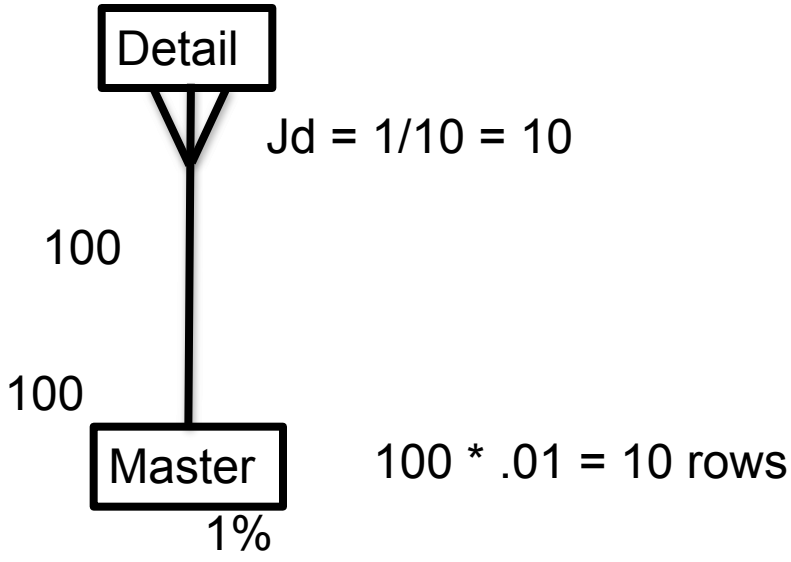
Join Filter ratios

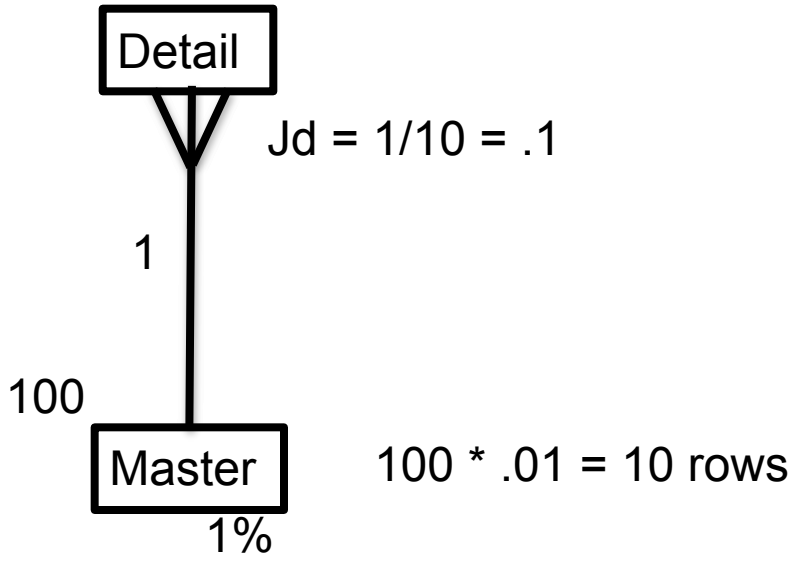


Join Detail Jd > 1 normally

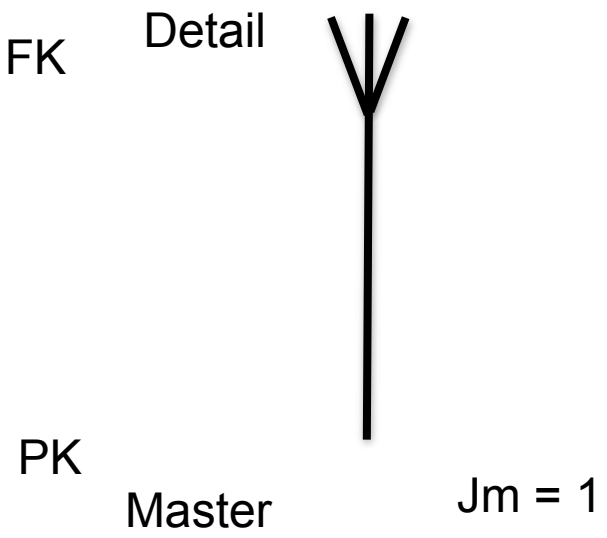
If Jd < 1 joining acts like a filter, unusual

If Jd = 0 then good place to start, rare





Join Filter ratios



Jm = “Join master ratio”

Jm = 1

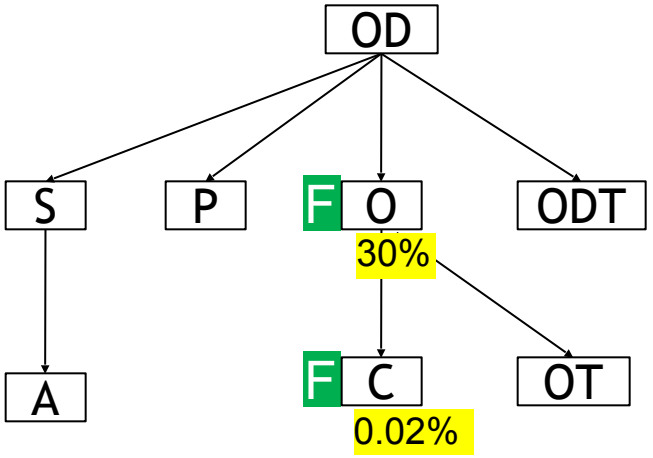
when PK/FK relationship

Jm < 1

possible when not PKFK relationship

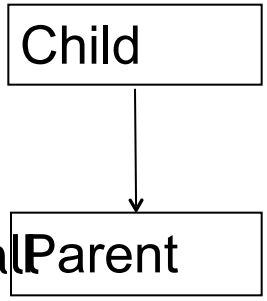
VST Steps Summary

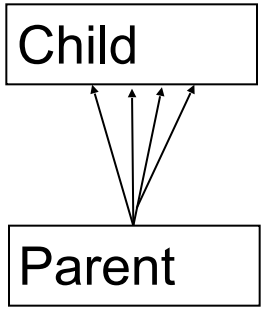
1. Diagram tables
2. Draw connectors for each join
3. Calculate filter ratio



Execution Path

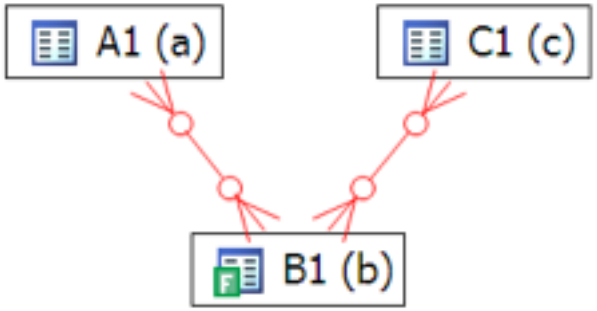
- Start at the most selective join filter
- Join to keep the running result set size small





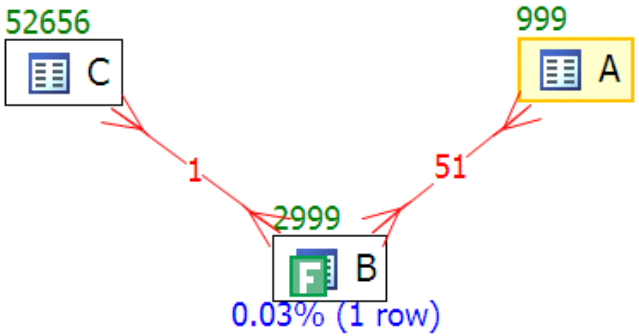
VST Steps Summary

Many to Many relationships = problems



If many to many then

calculate two table join sizes



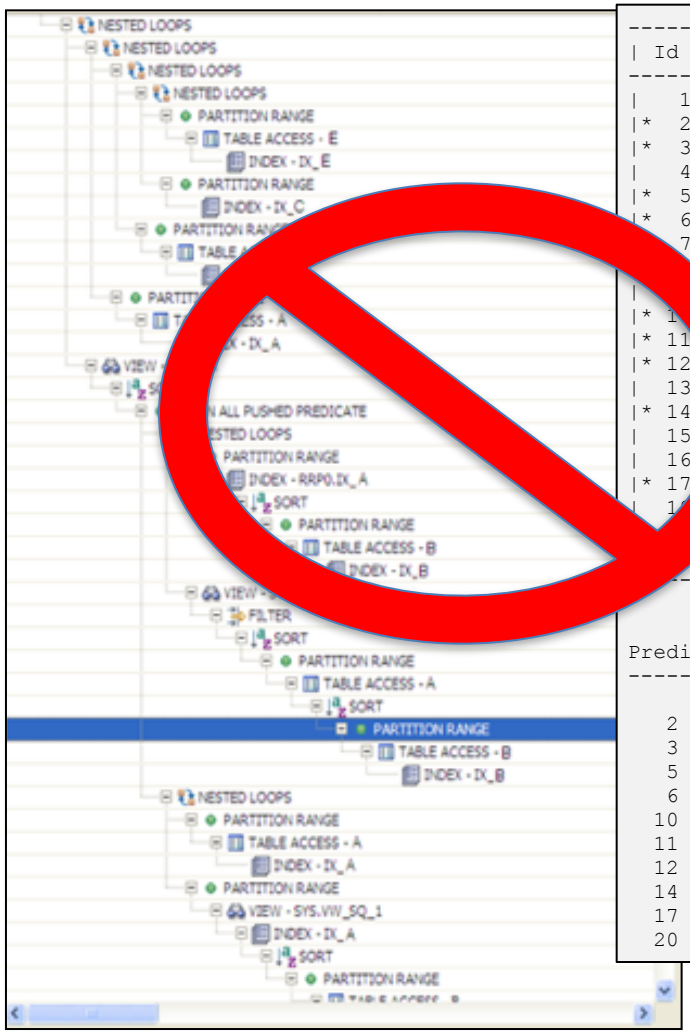
VST steps summary

Note

- Don’t talk about table sizes
  - Talk about filter ratios
  - Two table join sizes
  - Possibly look at join ratios
- Don’t talk about index existence
  - Finding optimal path indicates indexes needed

Step 2: Get Explain Plan

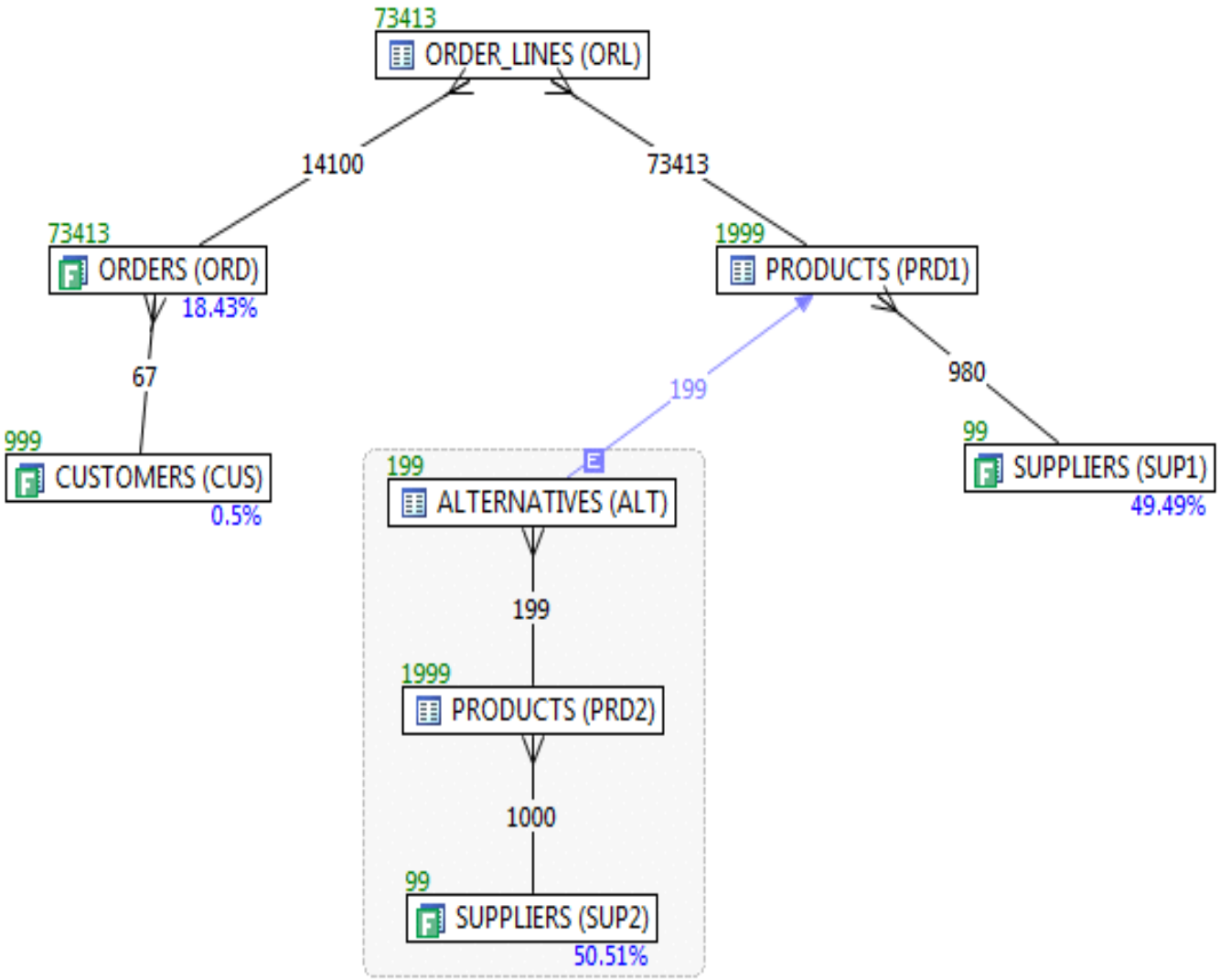
Trace fie



VST

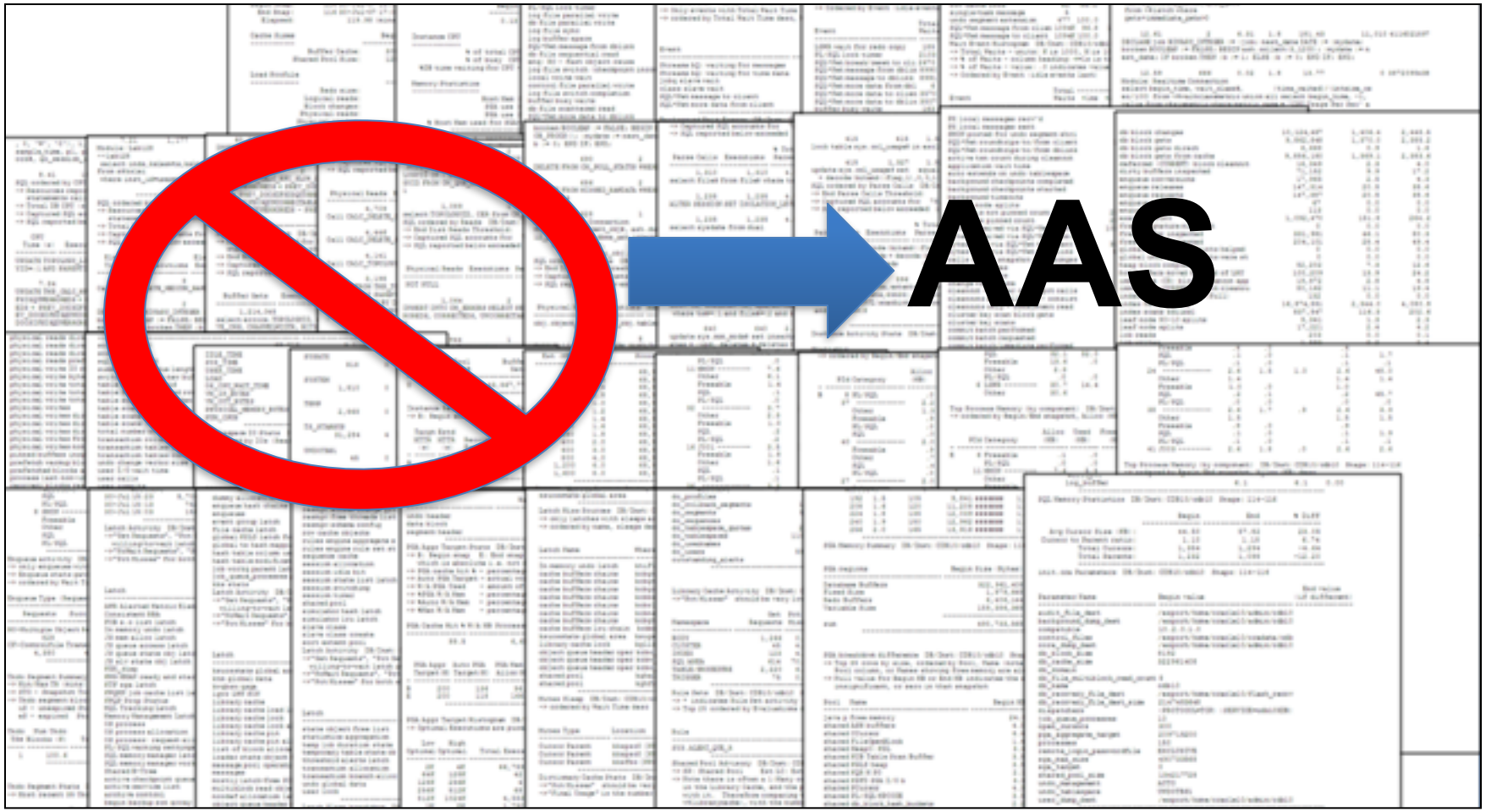
```
-------------------------------------------------------------------------------------------------
| Id  | Operation                            | Name                | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------------------------
|   1 |  HASH GROUP BY                       |                     |      1 |      1 |      1 |
|*  2 |   FILTER                             |                     |      1 |        |   1909 |
|*  3 |    TABLE ACCESS BY INDEX ROWID       | PS_RETROPAYPGM_TBL  |      1 |      1 |   3413 |
|   4 |     NESTED LOOPS                     |                     |      1 |    165 |   6827 |
|*  5 |      HASH JOIN                       |                     |      1 |    165 |   3413 |
|*  6 |       HASH JOIN                      |                     |      1 |    165 |   3624 |
|   7 |        TABLE ACCESS BY INDEX ROWID   | WB_JOB              |      1 |    242 |   2895 |
|     |         NESTED LOOPS                 |                     |      1 |    233 |   2897 |
|     |          TABLE ACCESS BY INDEX ROWID | PS_[illegible]AL    |      1 |      1 |      1 |
|* 1  |           INDEX RANGE [illegible]N   | PS_[illegible]AY_CALEND |      1 |      1 |      1 |
|* 11 |                                      | WB[illegible]       |      1 |    286 |   2895 |
|* 12 |                                      | WP_RETR[illegible]AY_E[illegible]NS |      1 |  27456 |   122K|
|  13 |                                      | PS_[illegible]AY_R[illegible]T |      1 |  13679 |  13679 |
|* 14 |      INDEX RANGE SCAN                | PS#RETROPAYPGM_TBL  |   3413 |      1 |   3413 |
|  15 |     SORT AGGREGATE                   |                     |   1791 |      1 |   1791 |
|  16 |      FIRST ROW                       |                     |   1791 |      1 |   1579 |
|* 17 |       INDEX RANGE SCAN (MIN/MAX)     | WB_JOB_F            |   1791 |      1 |   1579 |
|  18 |        SORT AGGREGATE                |                     |   1539 |      1 |   1539 |
|  19 |         FIRST ROW                    |                     |   1539 |      1 |   1539 |
|  20 |          INDEX RANGE SCAN (MIN/MAX)  | WB_JOB_G            |   1539 |      1 |   1539 |
-------------------------------------------------------------------------------------------------


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

   2 - filter(("B"."EFFDT"= AND "B"."EFFSEQ"=))
   3 - filter("E"."OFF_CYCLE"="A"."PAY_OFF_CYCLE_CAL")
   5 - access("D"."RETROPAY_SEQ_NO"="C"."RETROPAY_SEQ_NO")
   6 - access("C"."EMPLID"="B"."EMPLID" AND "C"."EMPL_RCD#"="B"."EMPL_RCD#")
  10 - access("A"."RUN_ID"='PD2' AND "A"."PAY_CONFIRM_RUN"='N')
  11 - access("B"."COMPANY"="A"."COMPANY" AND "B"."PAYGROUP"="A"."PAYGROUP")
  12 - filter(("C"."RETROPAY_PRCS_FLAG"='C' AND "C"."RETROPAY_LOAD_SW"='Y'))
  14 - access("E"."RETROPAY_PGM_ID"="D"."RETROPAY_PGM_ID")
  17 - access("F"."EMPLID"=:B1 AND "F"."EMPL_RCD#"=:B2 AND "F"."EFFDT"<=:B3)
  20 - access("G"."EMPLID"=:B1 AND "G"."EMPL_RCD#"=:B2 AND "G"."EFFDT"=:B3)
```

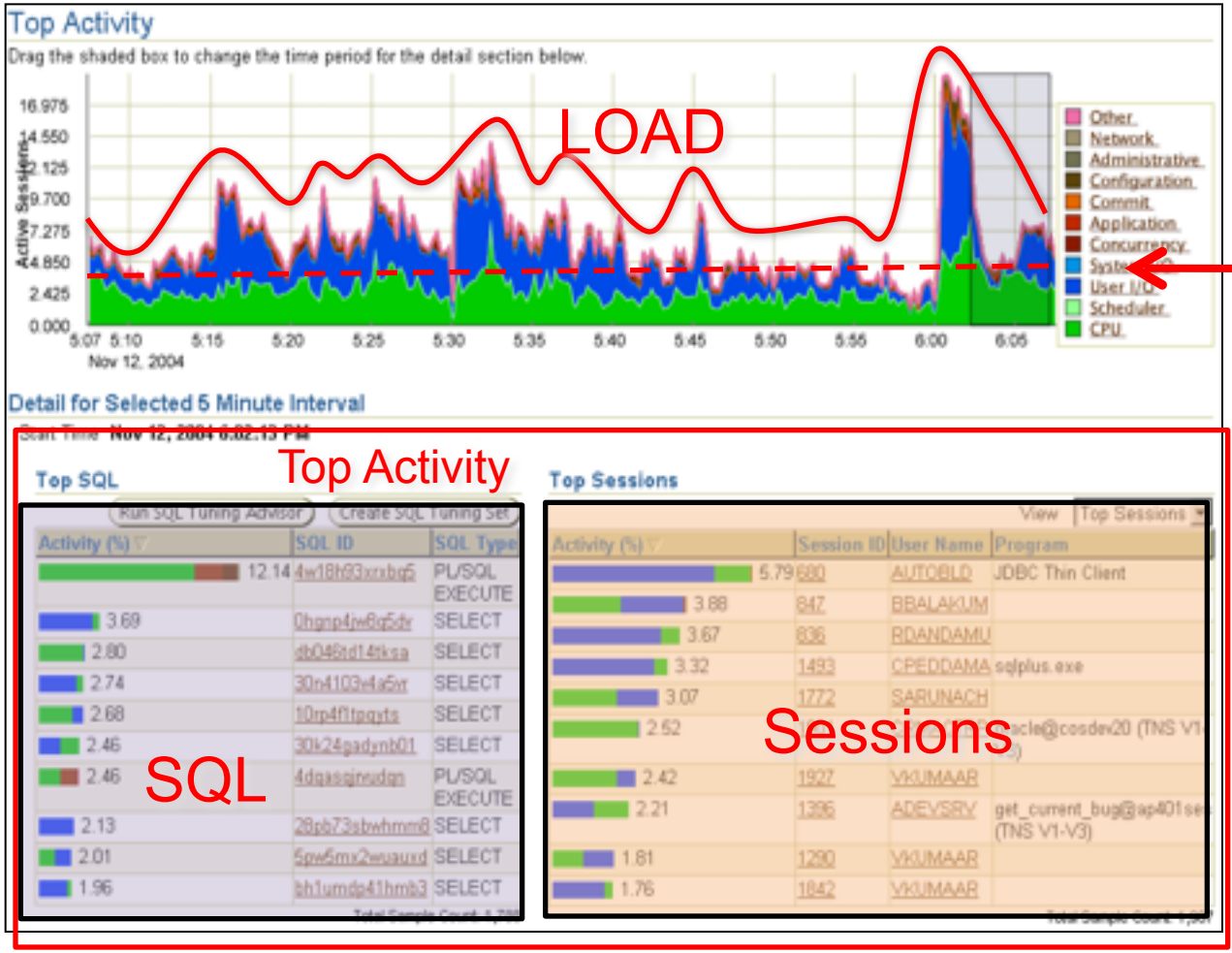
Visual SQL Tuning (VST)



DELPHIX

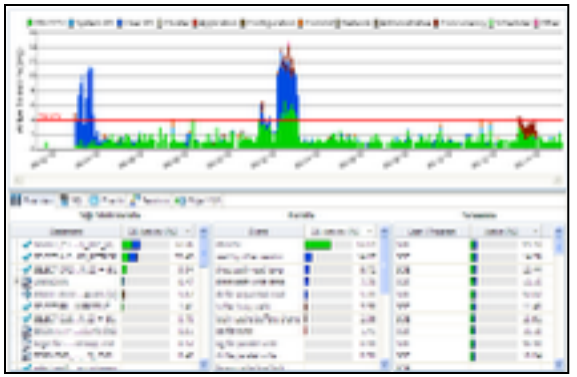
AWR Report / Statspack



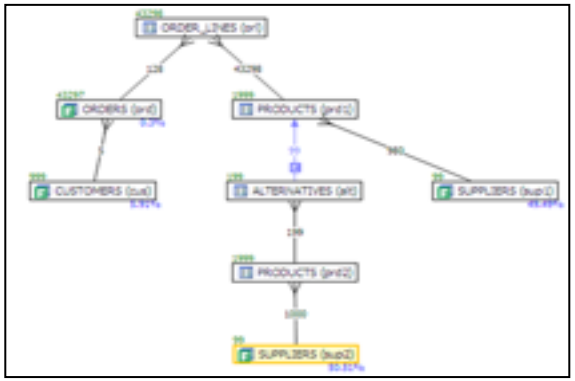

Top Activity
Drag the shaded box to change the time period for the detail section below.
LOAD
Active Sessions
16.975
14.550
12.125
9.700
7.275
4.850
2.425
0.000
5:07 5:10 5:15 5:20 5:25 5:30 5:35 5:40 5:45 5:50 5:55 6:00 6:05
Nov 12, 2004
Other
Network
Administrative
Configuration
Commit
Application
Concurrency
System I/O
User I/O
Scheduler
CPU
Detail for Selected 5 Minute Interval
Top Activity
Top SQL
Run SQL Tuning Advisor
Create SQL Tuning Set
Activity (%)
SQL ID
SQL Type
12.14 4w18h03xrxbq5 PL/SQL EXECUTE
3.69 0hgnp4jw8q5dv SELECT
2.80 db046td14tksa SELECT
2.74 30n4103w4a6wr SELECT
2.68 10rp4f1tpqyts SELECT
2.46 30k24gadynb01 SELECT
2.46 4dqasgjnudgn PL/SQL EXECUTE
2.13 28pb73sbwhmm8 SELECT
2.01 5pw5mx2wuauxd SELECT
1.96 bh1umdp41hmb3 SELECT
SQL
Top Sessions
View
Top Sessions
Activity (%)
Session ID
User Name
Program
5.79 680 AUTOBLD JDBC Thin Client
3.88 847 BBALAKUM
3.67 836 RDANDAMU
3.32 1493 CPEDDAMA sqlplus.exe
3.07 1772 SARUNACH
2.52
2.42 1927 VKUMAAR
2.21 1386 ADEVSRV
1.81 1290 VKUMAAR
1.76 1842 VKUMAAR
Sessions
Max CPU
(yard stick)

Summary

1. Database - AAS



2. SQL - VST



END

10053

```
Table: A  Alias: A
  Best:: AccessPath: TableScan
        Cost: 92.18  Degree: 1  Resp:
92.18  Card: 999.00
```

A

$: 92
#: 999

```
Best:: AccessPath: IndexRange
 Index: B_V2
        Cost: 4.00  Degree: 1  Resp:
4.00  Card: 1.00
```

B

$: 4
#: 1

```
Best:: AccessPath: IndexFFS
 Index: C_PK_CON
        Cost: 35.57  Degree: 1  Resp:
35.57  Card: 59999.00
```

C

$: 5428
#: 59999

B A C

B C A

C A B

C B A

A C B

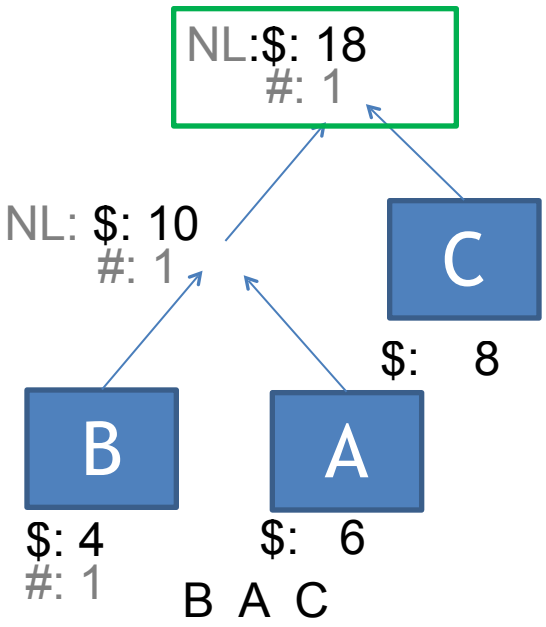
A B C

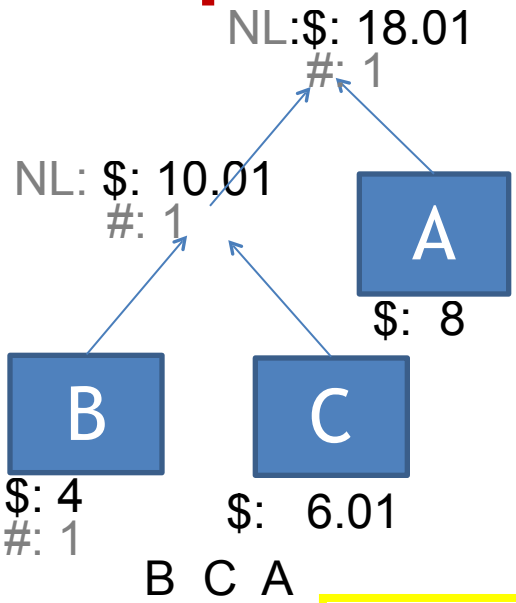
```
Join order[1]:  B[B]#0  A[A]#1  C[C]#2
***************
Now joining: A[A]#1
***************
Best:: JoinMethod: NestedLoop
      Cost: 6.00  Degree: 1  Resp: 6.00  Card: 1.00 Bytes: 16
***************
Now joining: C[C]#2
***************
Best:: JoinMethod: NestedLoop
      Cost: 6.00  Degree: 1  Resp: 6.00  Card: 1.00 Bytes: 21
***********************
Best so far:  Table#: 0  cost: 4.0020  card: 1.0000  bytes: 12
              Table#: 1  cost: 6.0031  card: 1.0000  bytes: 16
              Table#: 2  cost: 6.0032  card: 1.0000  bytes: 21
```

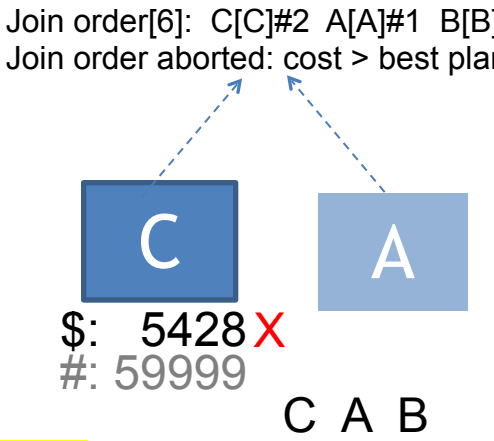
Total: 14.0083

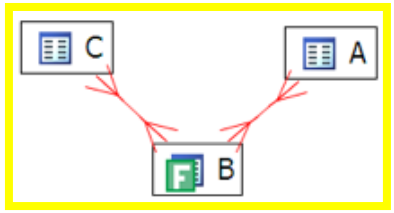
```
alter session set events='10053 trace name context forever';
```

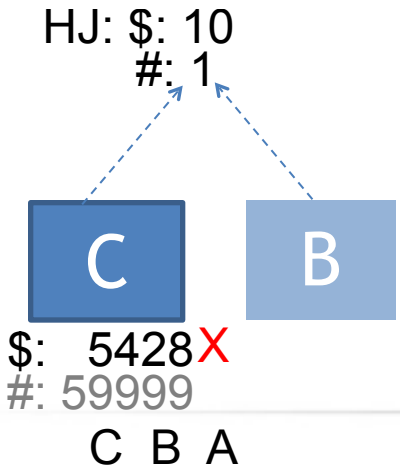
No Unique Constraint

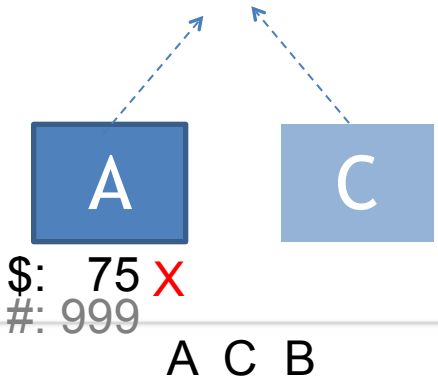


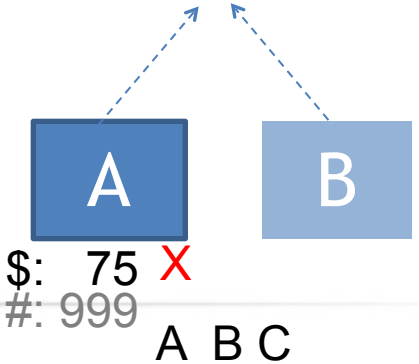




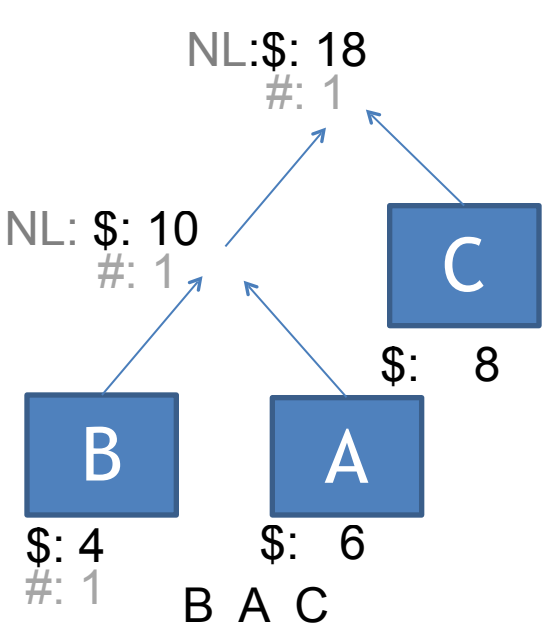


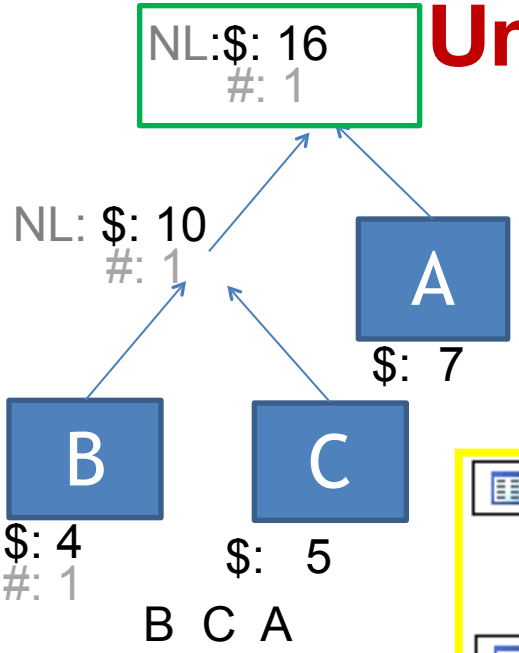


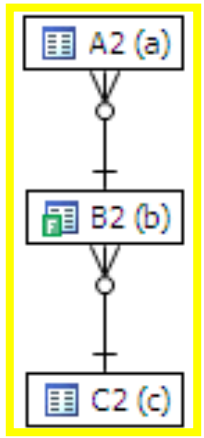




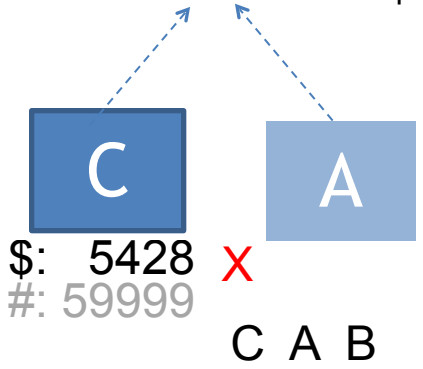
Unique Constraint

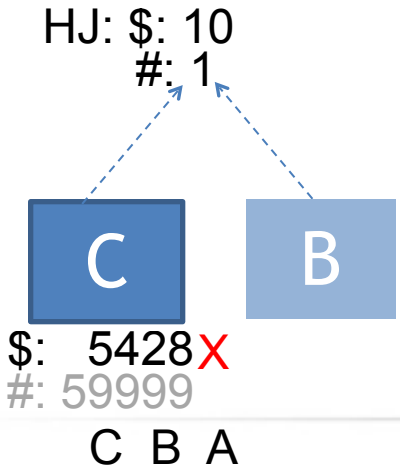


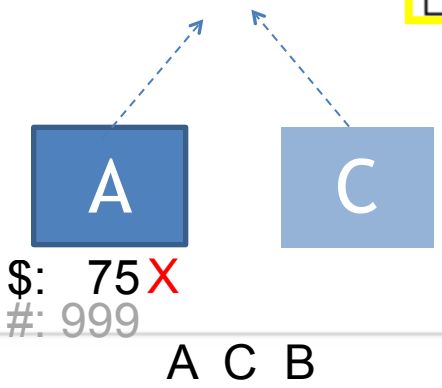


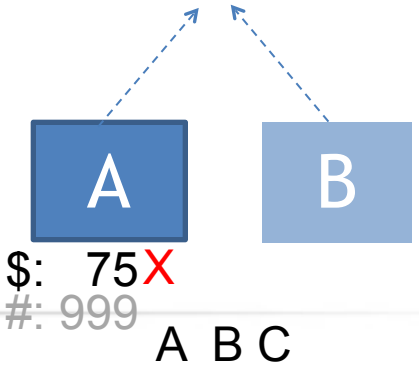


Join order[6]: C[C]#2 A[A]#1 B[B]#0
Join order aborted: cost > best plan cost

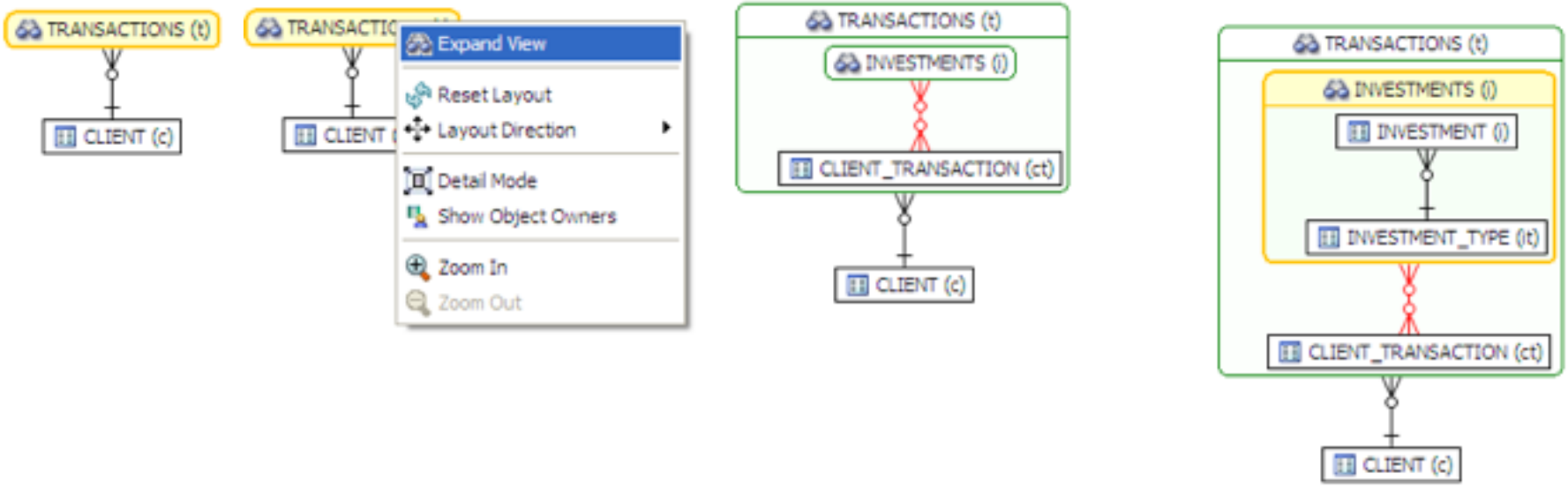




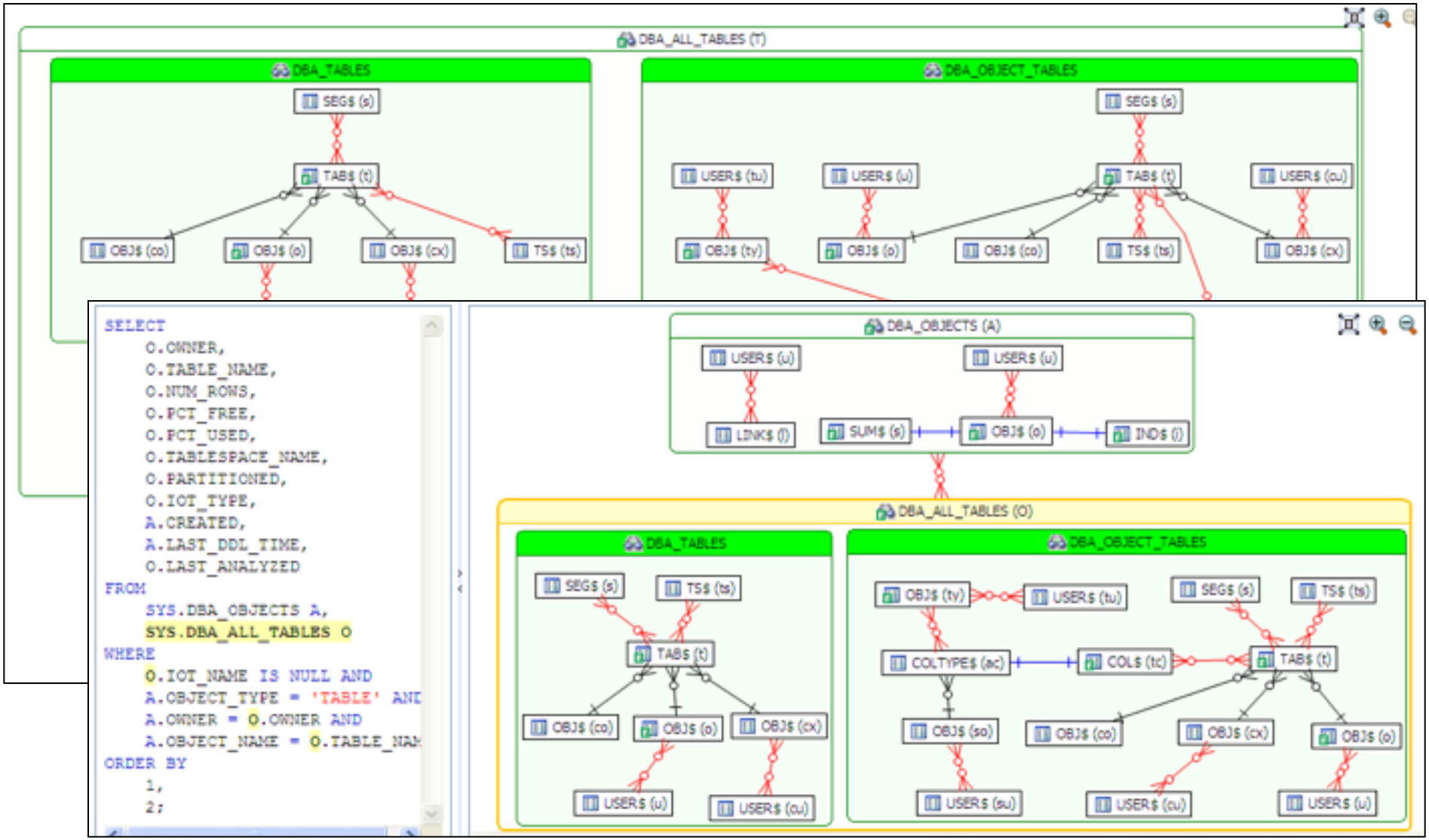




View Expansion



View Expansion

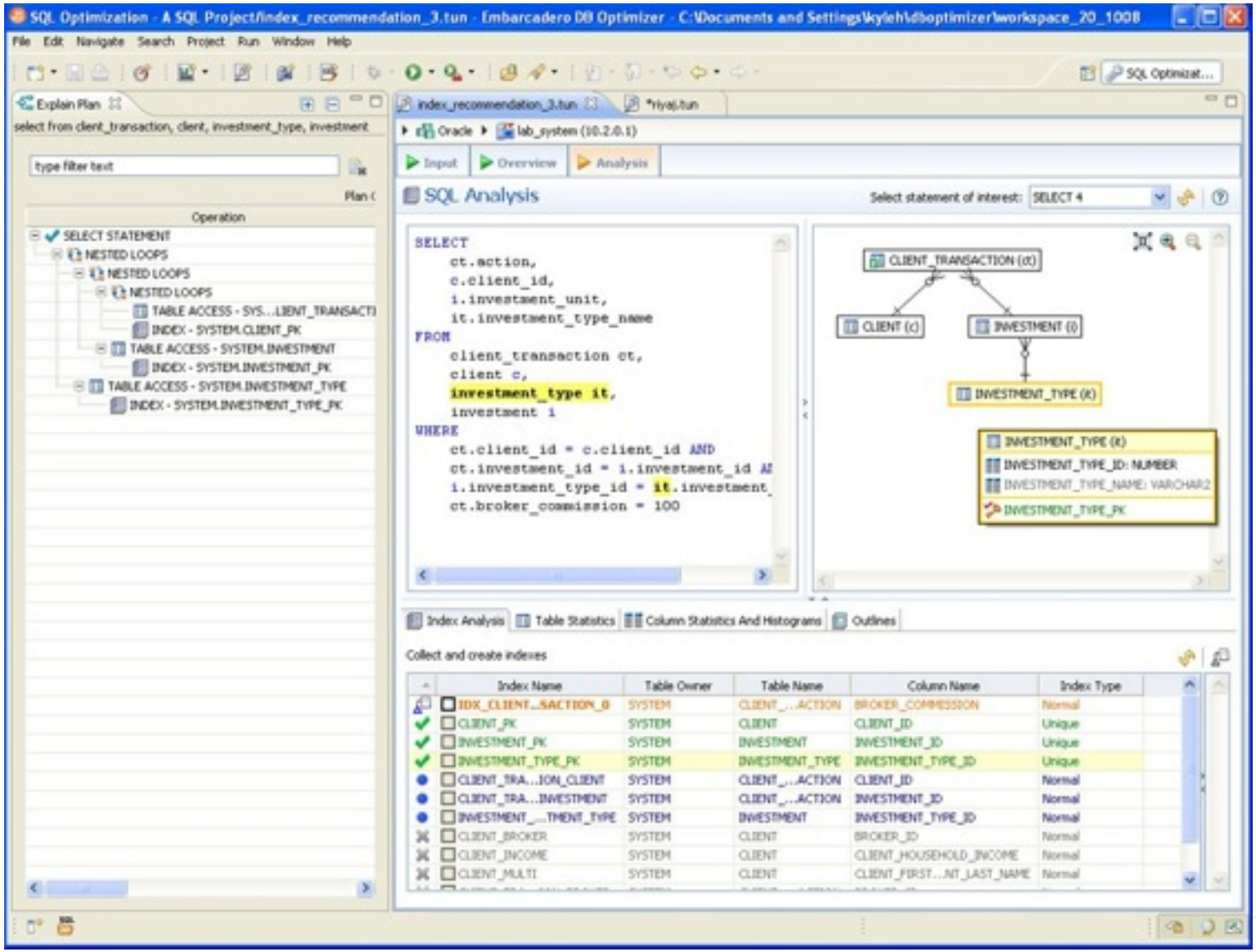


Types of Subqueries

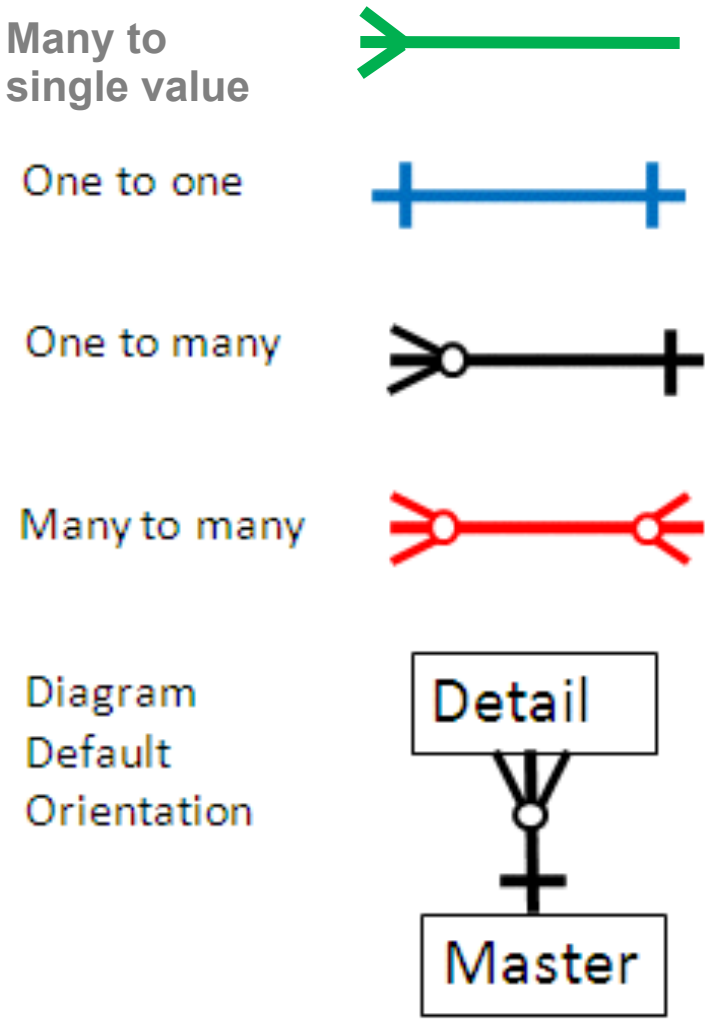
- subquery
  - query in where clause
  - Select 1 from dual where 1 = (select 1 from dual);
- correlated subquery
  - Subquery uses fields from outer query
  - Select 1 from dual a where 1=(select 1 from dual b where a.dummy=b.dummy)
- scalar subquery
  - Query in select list
  - Returns 1 value or null , ie scalar
  - Select (select 1 from dual) from dual;
- inline views
  - Query in from clause
  - Select 1 from dual a, (select 1 from dual) b;

DELPHIX

Visual SQL Tuning



- Layout tables Graphically
  - Details tables above Masters
  - One to one relations side by side
  - Many to many any which way



Join Set Sizes

Join	type	max result set size possible	notes
	one-to-scalar	A	this is equivalent to a filter on A table B returns one value like a max(), min(), count() etc
	one-to-one	min(rA,rB)	
	one-to-many	rA	Joining from A to B will not increase the result set size
	many-to-many	rA*rB ---------- min(ndv(A),ndv(B))	The more duplicates in both tables the greater chance the result set size will explode

We can say a lot about the join set size between two tables just by the join type and the number of rows in the table and NDV in the join column, but the easiest way most often is just to extract the two table joins and run a count(*) on that subset of the query

Join Filter ratios

FK

Jd ~> 1

Jm=1

PK

detail

Jd ~> 1

Jm <= 1

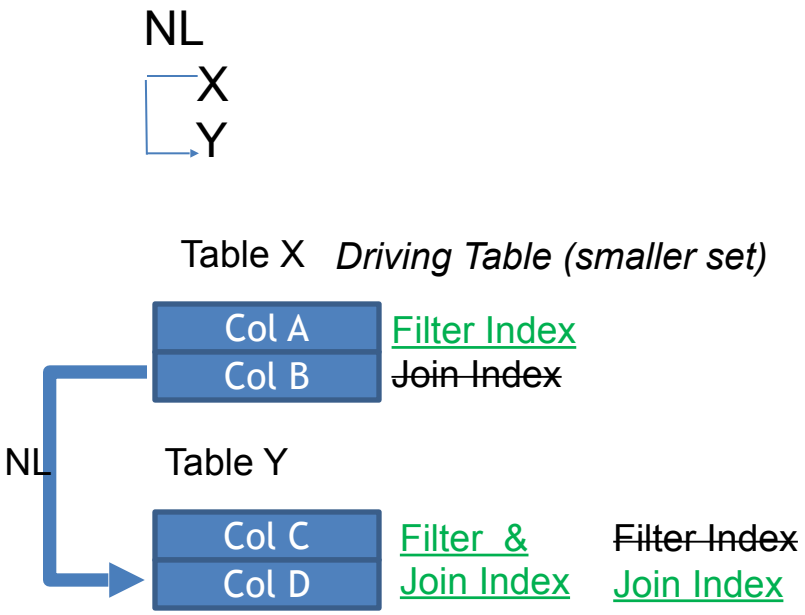
Master

Unique index or unique constraint

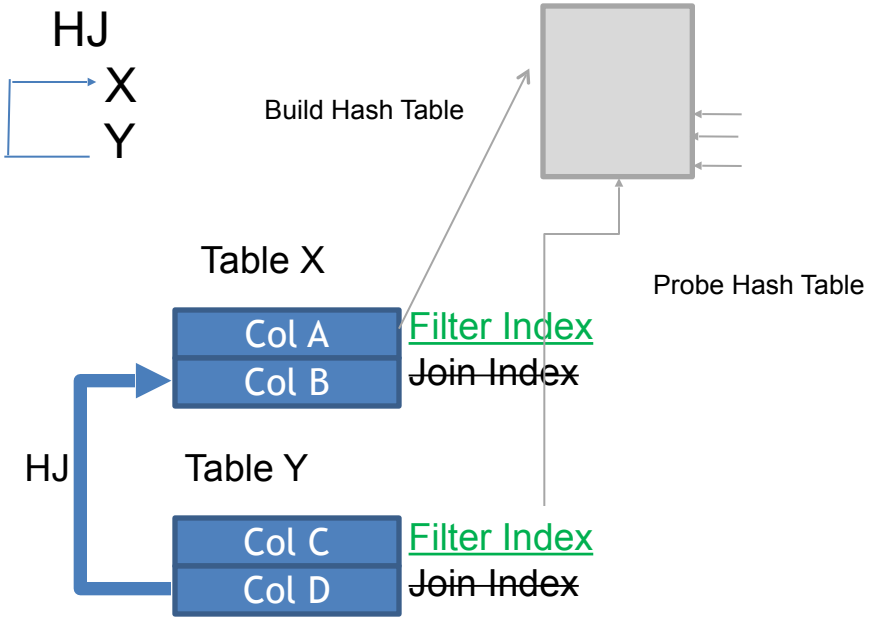
Jd, = join detail ratio , the avg number of rows returned when joining from the master to child
Jm = master join ratio, the avg number of rows returned joining from the child into the master.
Jm = 1 when PK/FK relations,
Jm < 1 possible when unique indexes or unique constraints instead of PK/FK.

Jd < 1 rare but acts as filter instead of a multiplier

Hash Join vs NL

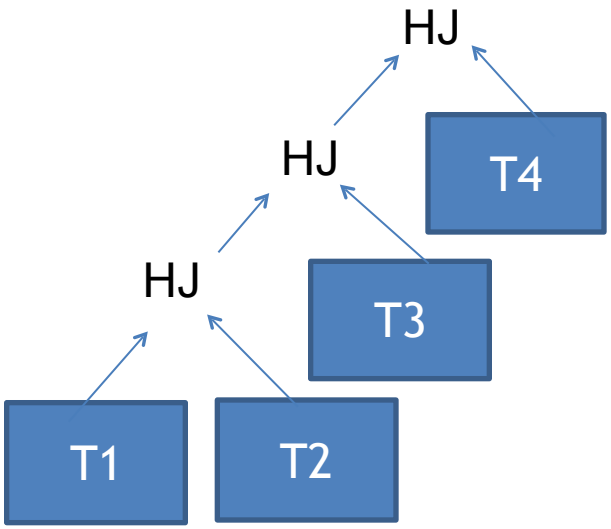


Drive from Table Y off of set of rows returned from filter on column C
Nested loops into Table X on Index on Join
Filter results without Index even though index on filter column

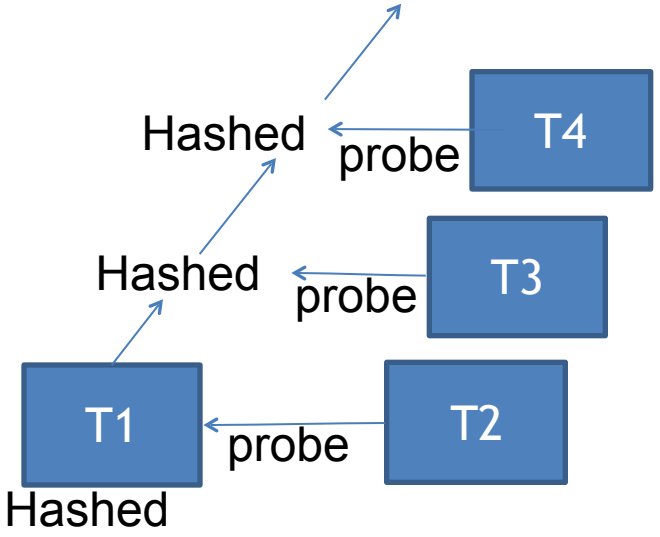


Create hash result set on Table Y from filter on column C
Probe hash result set with rows from filter on table X on column A

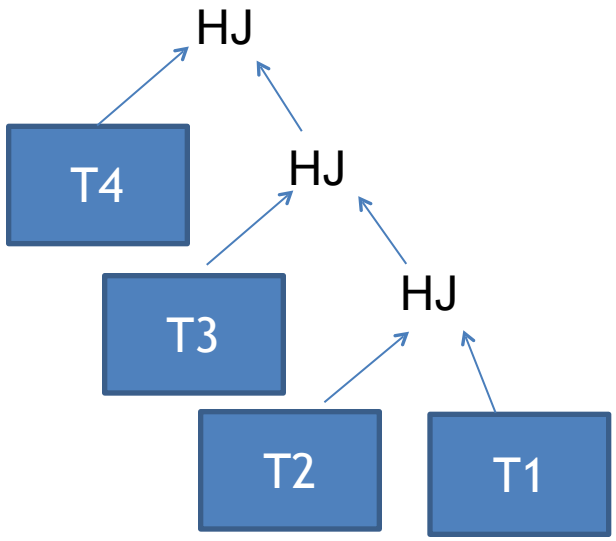
Left Deep HJ



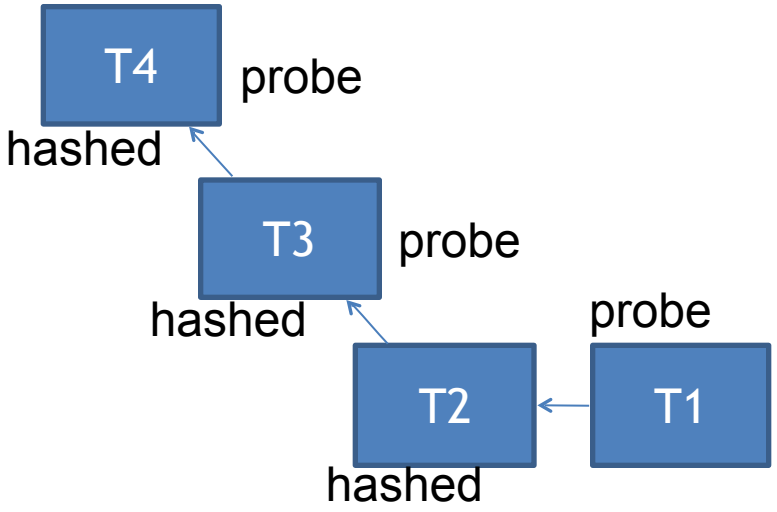

```
HJ
  HJ
    HJ
      T1
      T2
    T3
  T4
```



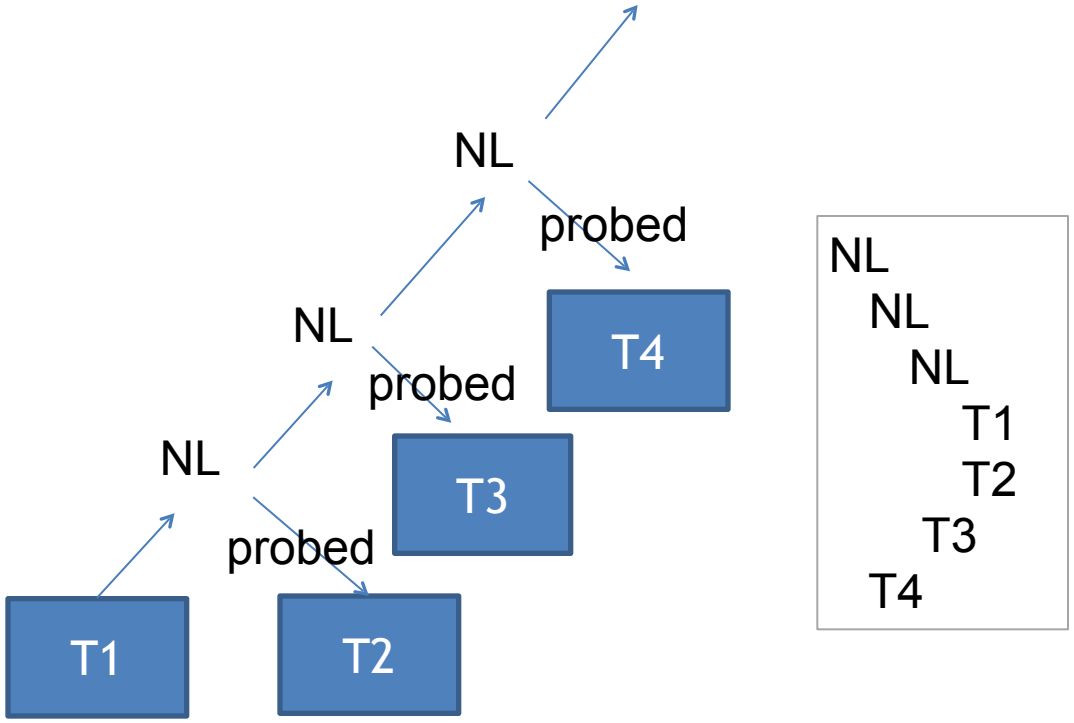
Right Deep HJ




```
HJ
  T4
  HJ
    T3
    HJ
      T2
      T1
```

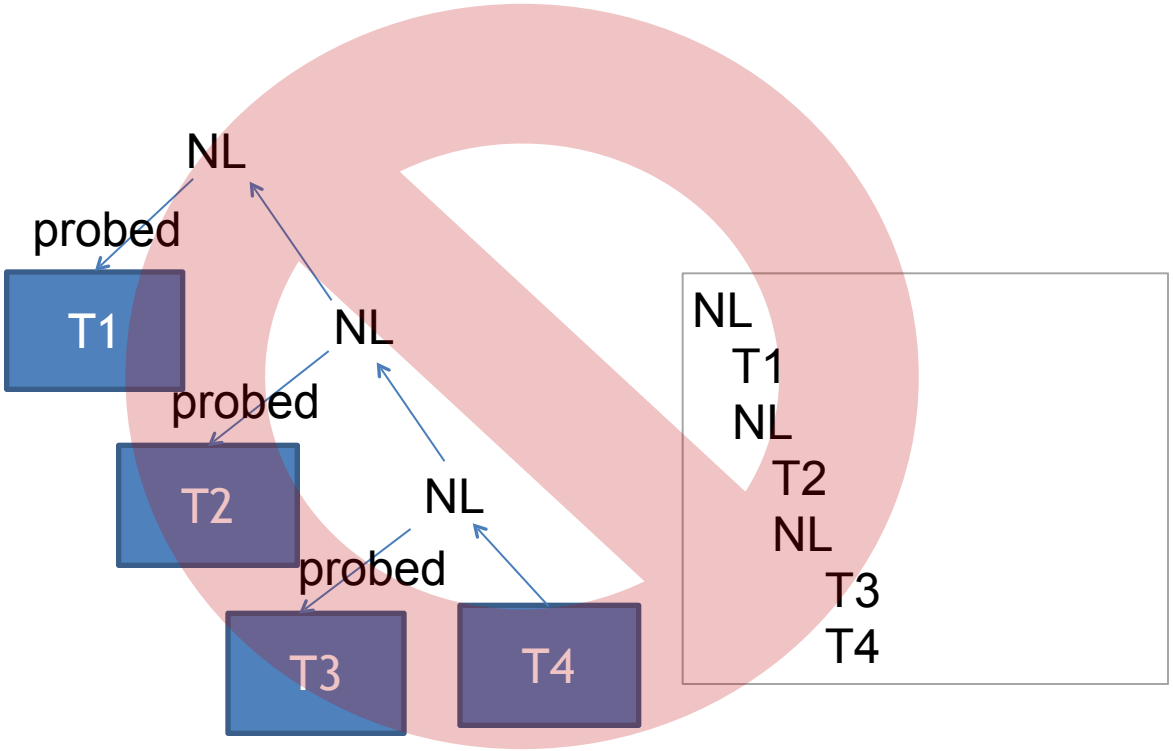


Nest Loops left deep



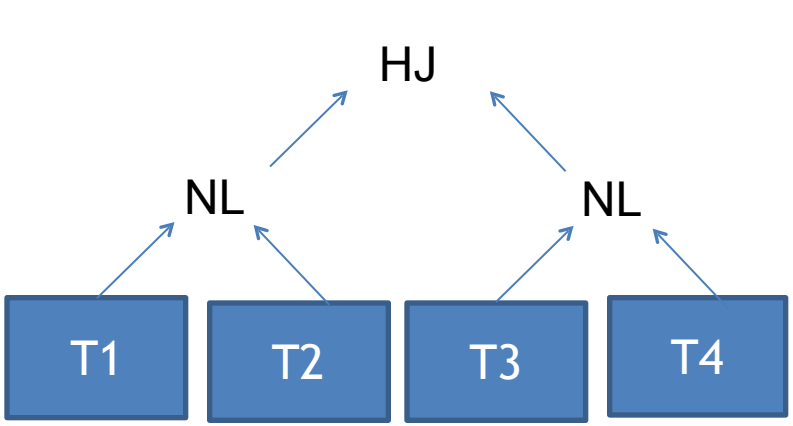

```
NL
  NL
    NL
      T1
      T2
    T3
  T4
```

Nest Loops can't be right deep



Bushy

```
select t3id, a.*, b.* from
  ( select  /*+  no_merge */
          t2.id t2id, t1.data t1data, t2.data t2data
    from  t1, t2
    where t1.id = t2.id ) a,
  ( select  /*+  no_merge */
          t3.id t3id, t3.data t3data, t4.data t4data
    from  t3, t4
    where t3.id = t4.id ) b
where  a.t2id=b.t3id
/
```



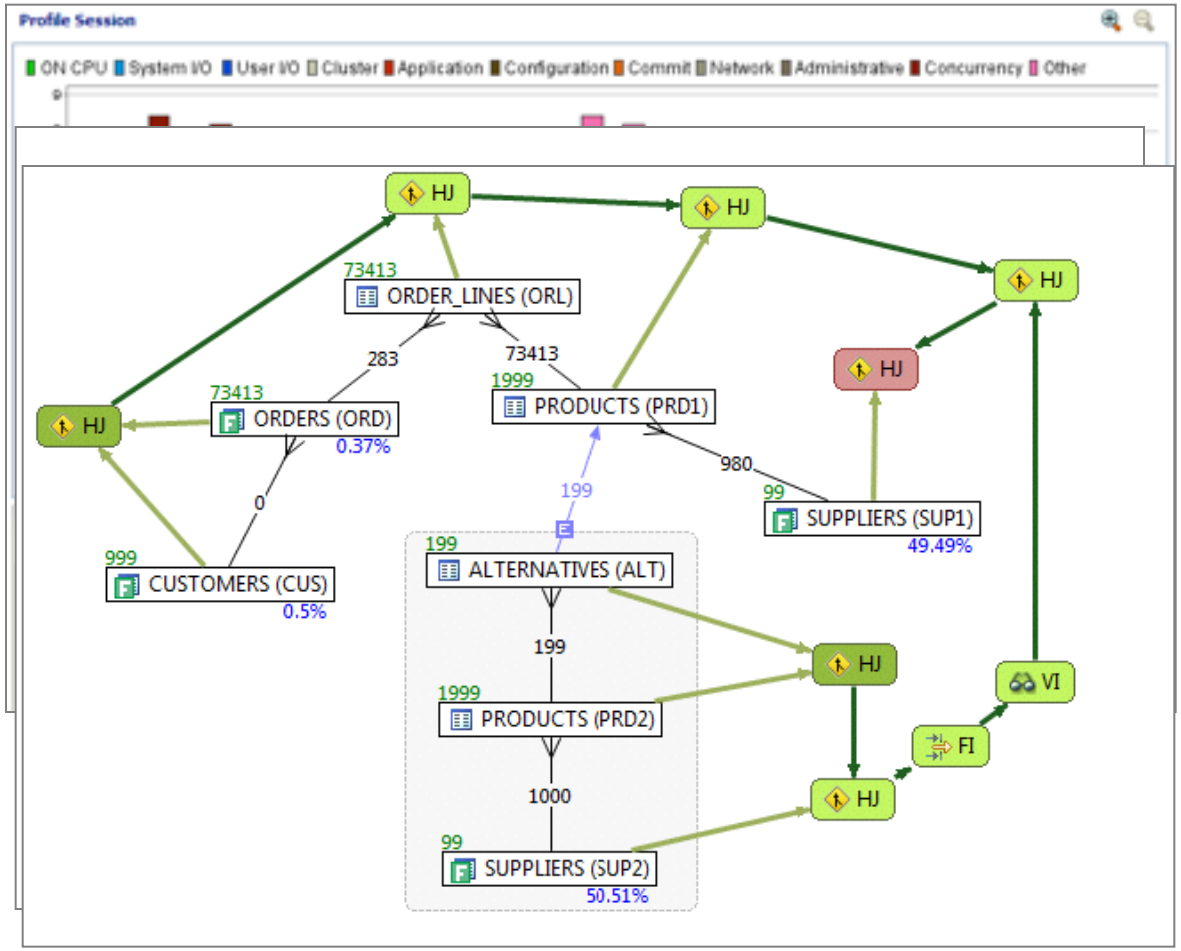

```
HJ
  NL
    T1
    T2
  NL
    T3
    T4
```

Who is Kyle Hailey

- 1990 Oracle
  - 90 support
  - 92 Ported v6
  - 93 France
  - 95 Benchmarking
  - 98 ST Real World Performance
- 2000 Dot.Com
- 2001 Quest
- 2002 Oracle OEM 10g

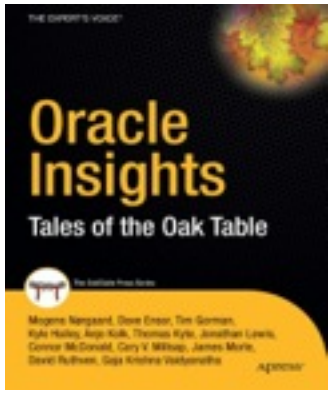
Who is Kyle Hailey

- 1990 Oracle
  - 90 support
  - 92 Ported v6
  - 93 France
  - 95 Benchmarking
  - 98 ST Real World Performance
- 2000 Dot.Com
- 2001 Quest
- 2002 Oracle OEM 10g
- 2005 Embarcadero
  - DB Optimizer



Who is Kyle Hailey

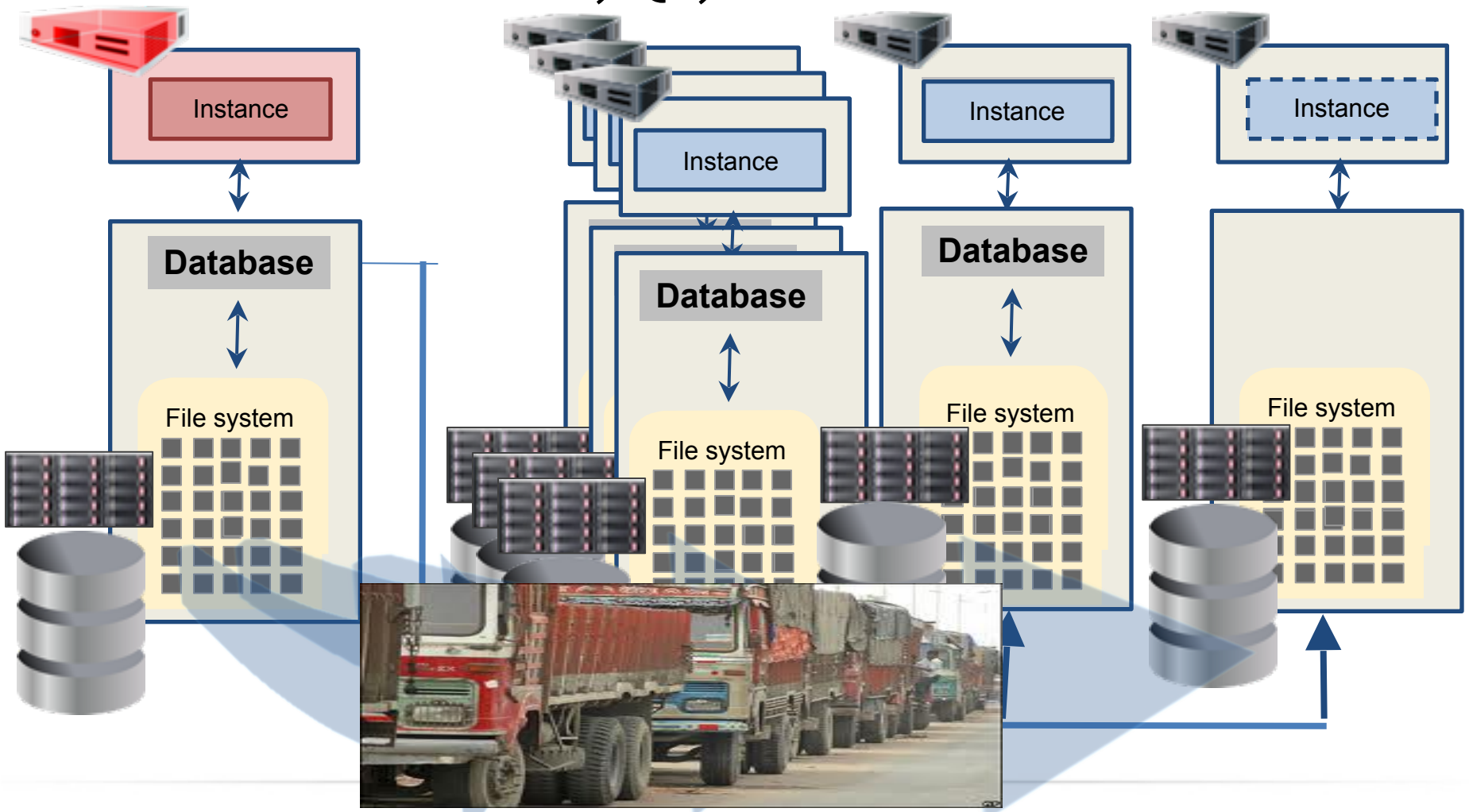
- 1990 Oracle
  - 90 support
  - 92 Ported v6
  - 93 France
  - 95 Benchmarking
  - 98 ST Real World Performance
- 2000 Dot.Com
- 2001 Quest
- 2002 Oracle OEM 10g
- 2005 Embarcadero
  - DB Optimizer
- Delphix



When not being a Geek

- Have a little 4 year old boy who takes up all my time

Typical Architecture



With Delphix