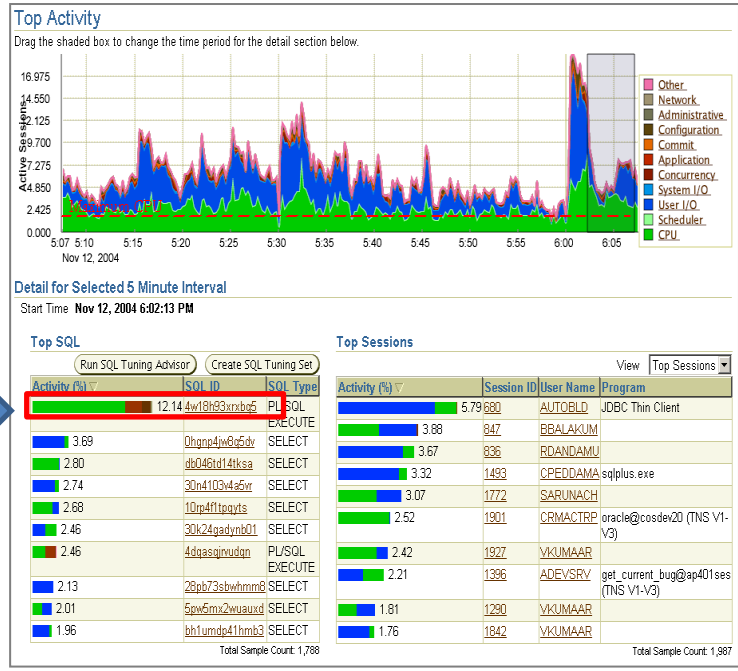
Methodology

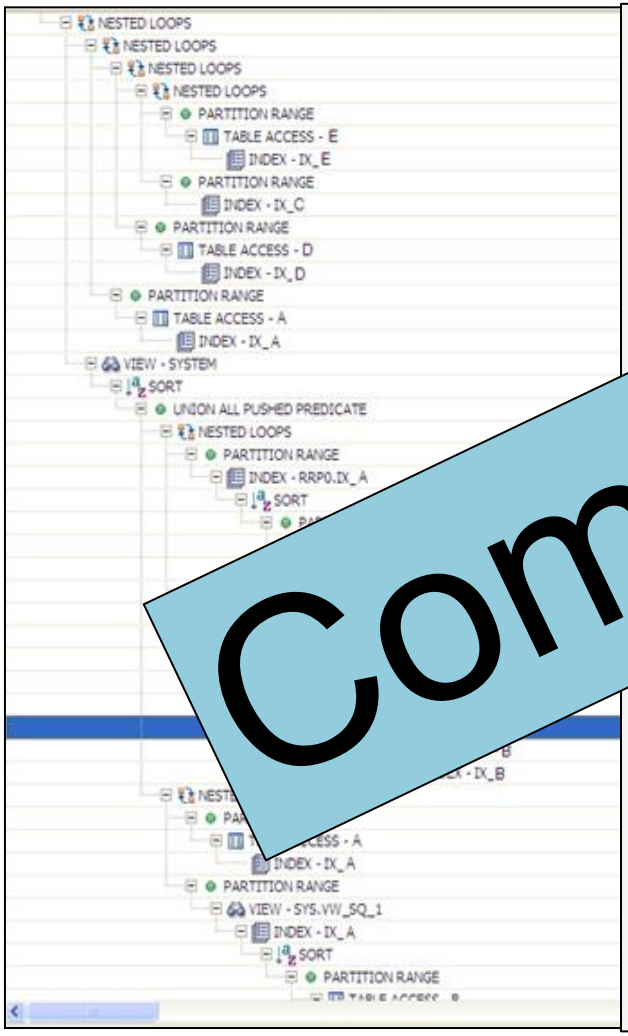
1. Find: Problem SQL
2. Study: SQL Execution Plan
3. Fix: ??

Step 1 : Top SQL

- Users complain
- High Resources
- Monitoring



Step 2: Explain Plan




```
-------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                 | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------------------------
|   1 |  HASH GROUP BY                     |                      |      1 |      1 |      1 |
|*  2 |   FILTER                           |                      |      1 |        |   1909 |
|*  3 |    TABLE ACCESS BY INDEX ROW       |                      |      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 ROW   |                      |      1 |    242 |   2895 |
|   8 |         INDEX RANGE SCAN           |                      |      1 |    233 |   2897 |
|   9 |        TABLE ACCESS BY INDEX ROW   |                      |      1 |      1 |      1 |
|* 10 |         INDEX RANGE SCAN           | PS_CALENDAR          |      1 |      1 |      1 |
|*  11 |       TABLE ACCESS BY INDEX ROW   | PS_RETROPAY_EARNS    |      1 |    286 |   2895 |
|  12 |        INDEX RANGE SCAN            | PS_RETROPAY_RQST     |      1 |  27456 |   122K |
|  13 |      TABLE ACCESS BY INDEX ROW     | PS#RETROPAYPGM_TBL   |      1 |  13679 |  13679 |
|  14 |       INDEX RANGE SCAN             |                      |   3413 |      1 |   3413 |
|  15 |     TABLE ACCESS BY INDEX ROW      |                      |   1791 |      1 |   1791 |
|  16 |      INDEX RANGE SCAN (MIN/MAX)    | WB_JOB_F             |   1791 |      1 |   1579 |
|  17 |     TABLE ACCESS BY INDEX ROW      |                      |   1539 |      1 |   1539 |
|  18 |      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)
```

Complexity !

Step 3: ???

Methodology

- Identify Slow Queries 
- Look at Execution Plan 
- Fix WTF?? 

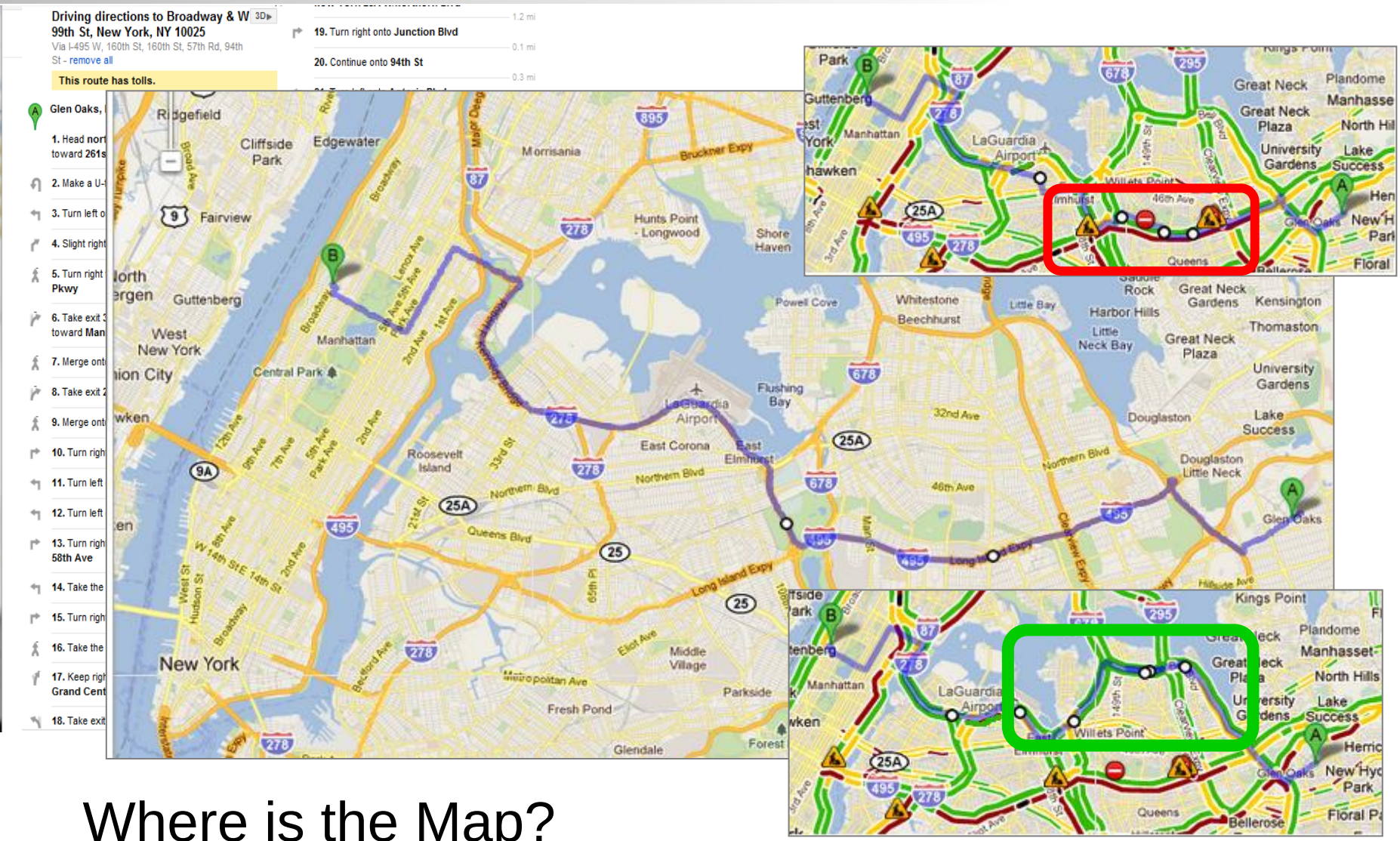
Fix

1. Analyze stats
2. Go to step 1

Fix If you are lucky

1. Look at histograms
2. Full Table Scans
3. Add Some Indexes

Explain Plan: Like Directions



Where is the Map?

SQL Tuning

1. Two Table Join
2. Multi-Table Join
3. Making the Map
4. Methodology
5. Examples

How to Join two tables

Maybe the hardest step in Optimization

- Which table to start with
- What Indexes to use
- What type of join to use HJ, NL

Design issues: partitions, IOT, bitmap indexes

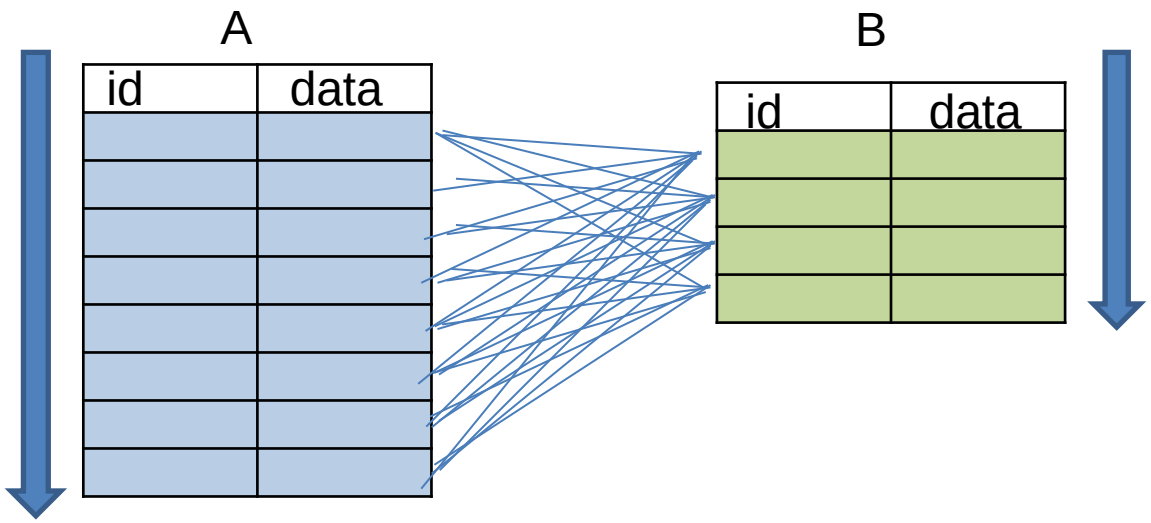
Two Table Join

1. No indexes
2. Indexes
3. One to One
4. One to Many
5. Many to Many

Table Join Order

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

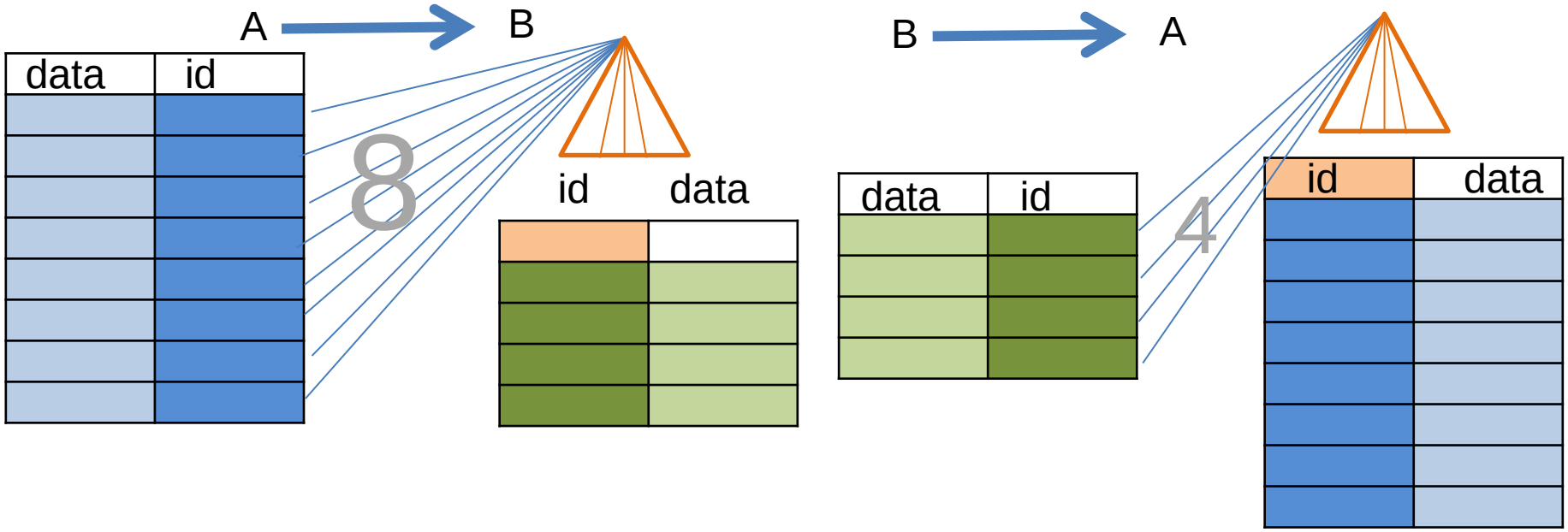
If No Index then order doesn't matter



Every row visits Every row
Work= A-rows x B-rows (8 x 4)
HJ might be optimal to index lookup

2 Table join, with indexes

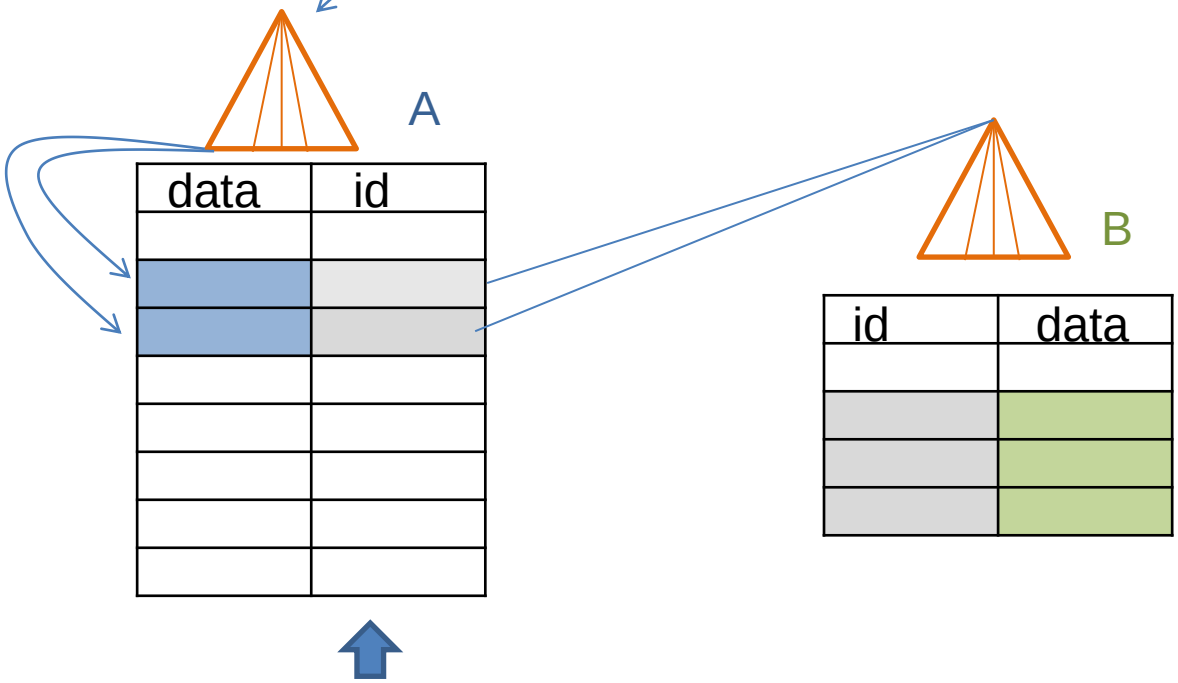
```
select    *
from      a, b
where
          a.id = b.id
```



Start with the least rows

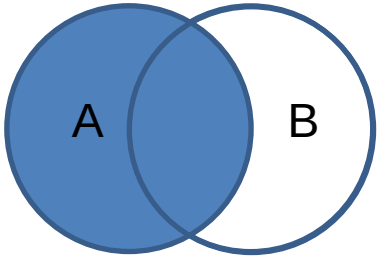
2 table join, index and filters

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

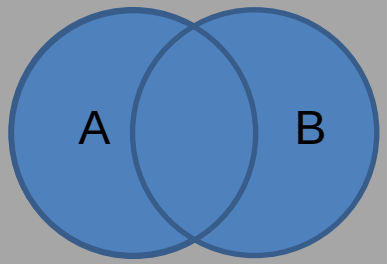


Start on table with least rows after filter

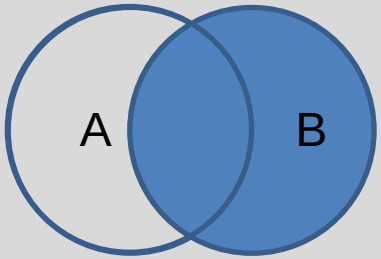
Two Table Joins



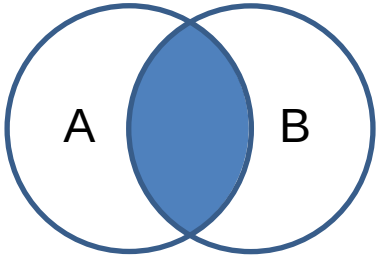
Left join B B(+)



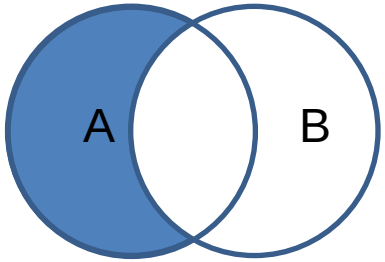
Full outer (union)



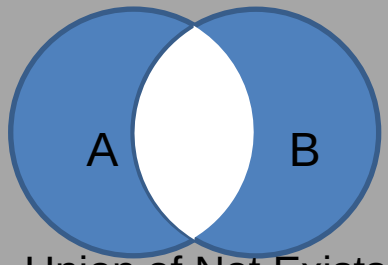
Right join B A(+)



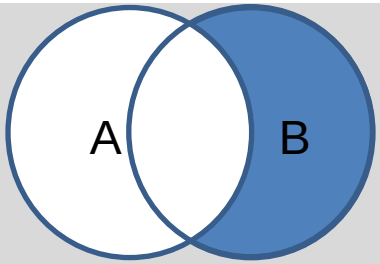
Inner Join



Not exists B



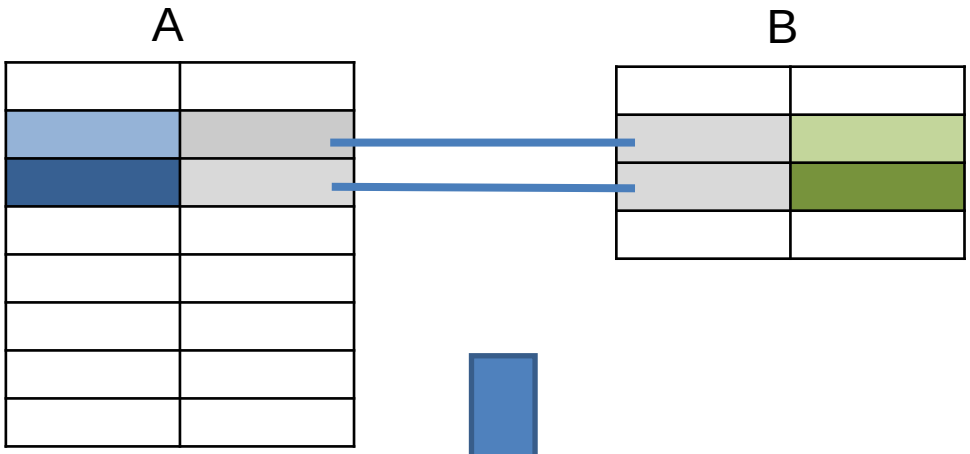
Union of Not Exists

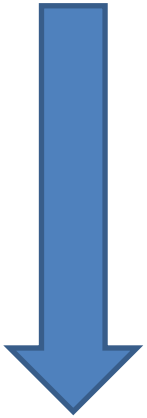


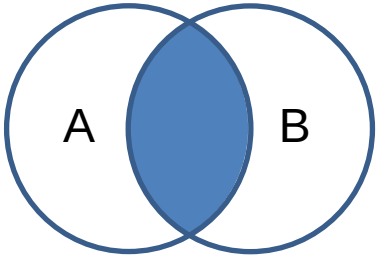
Not Exists A

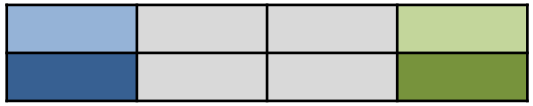
DELPHIX™

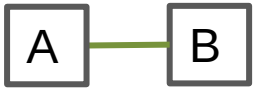
One to One



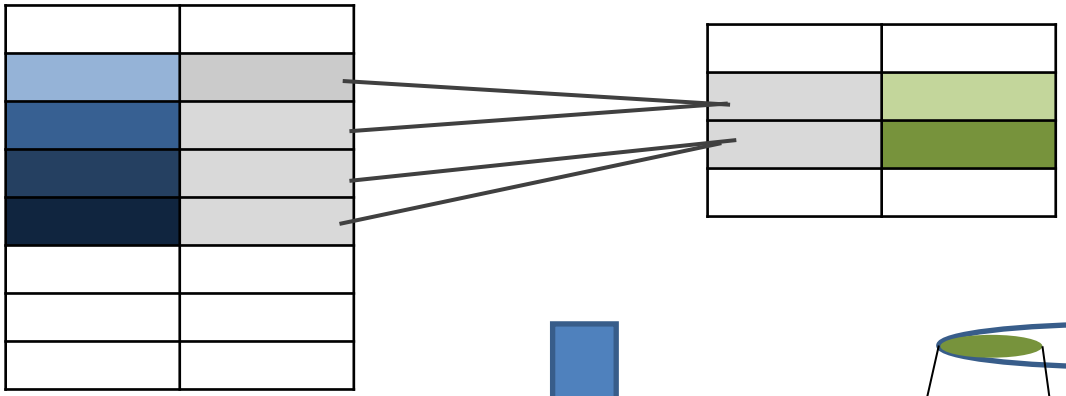


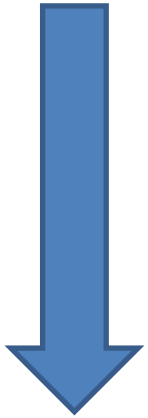


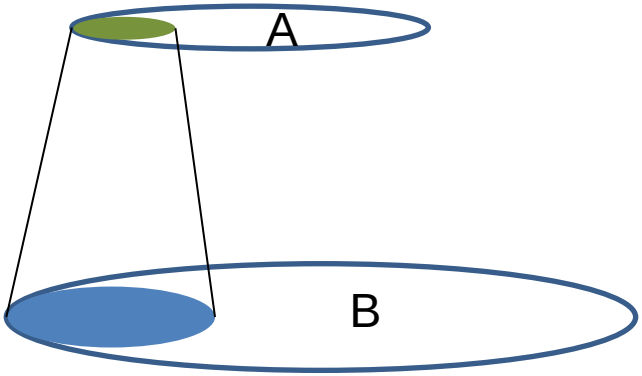


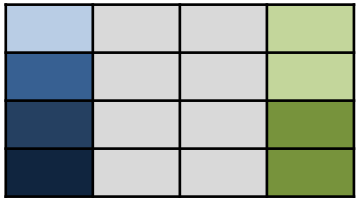


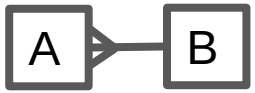
One to Many



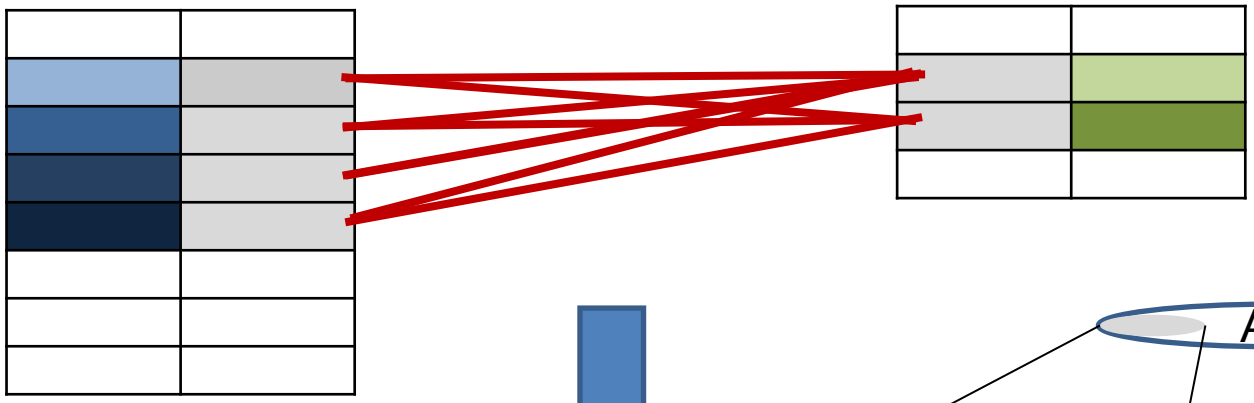


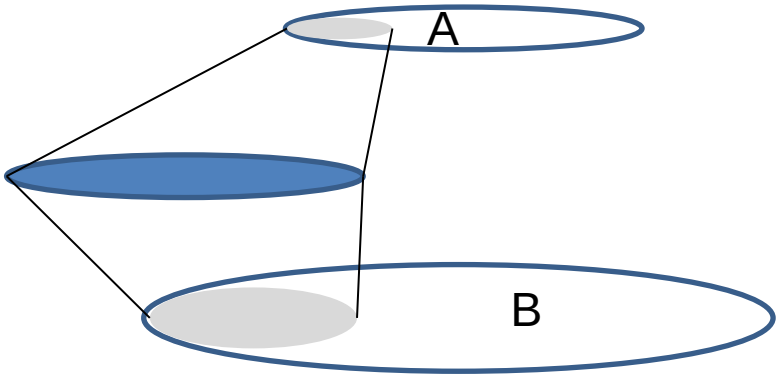


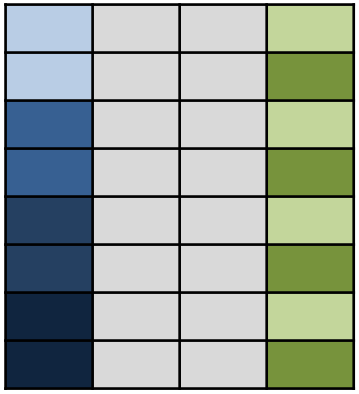


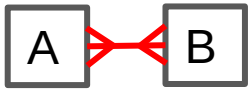


Many to Many

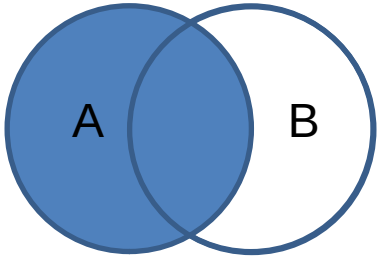








Two Table Joins



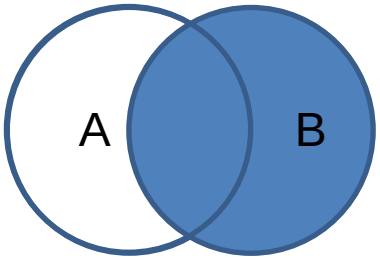
Left join B B(+)

english

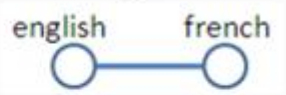
ENGLISH_TEXT	ORDINAL_ID
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

french

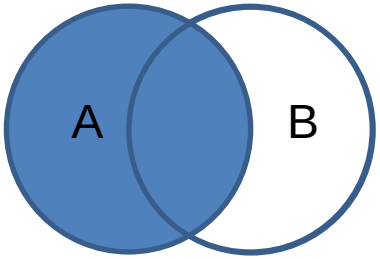
ORDINAL_ID	FRENCH_TEXT
1	Un
3	Trois
4	Quatre
5	Cinq
6	Six
7	Sept
8	Huit



Right join B A(+)

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		

Two Table Joins



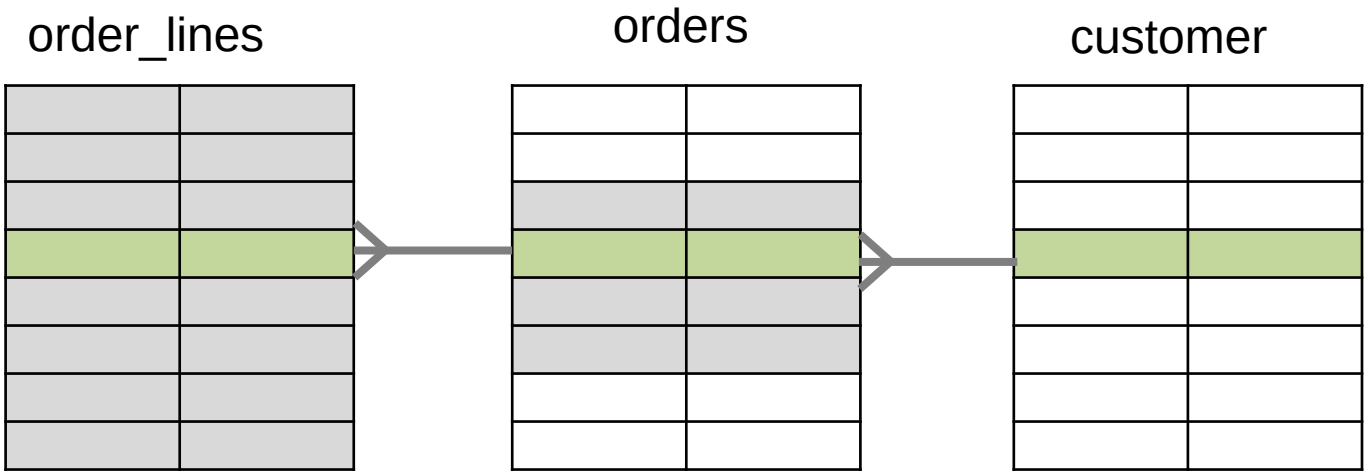
Left join B B(+)

Color nodes that return no data

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

Multiplier – taking B out will change query
But query is probably wrong a

Three table join



Join direction

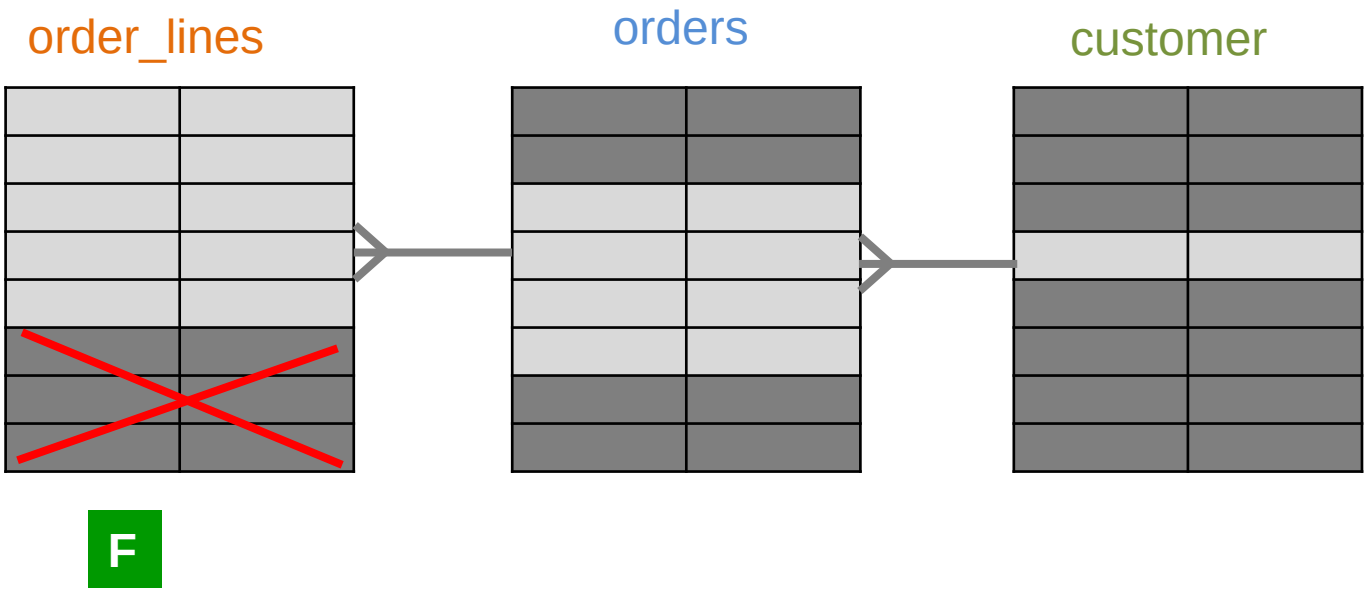
8 rows ← 4 rows ← 1 row

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

Join direction

1 row → 1 row → 1 row

Three table join

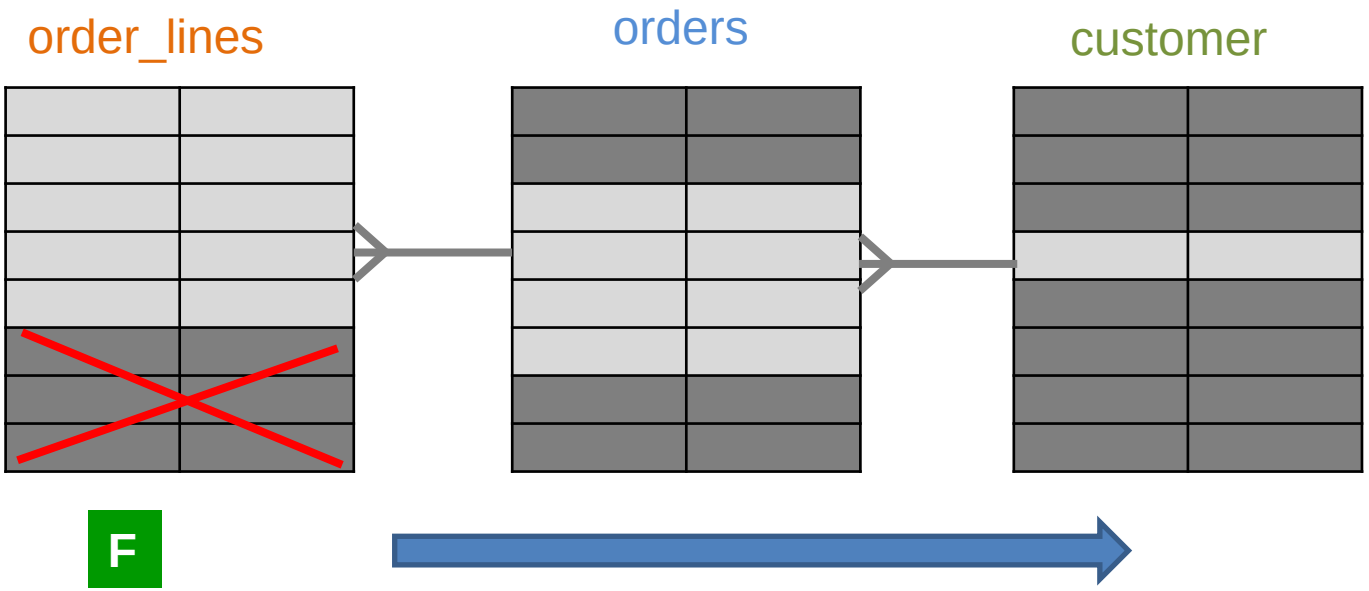


Where

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

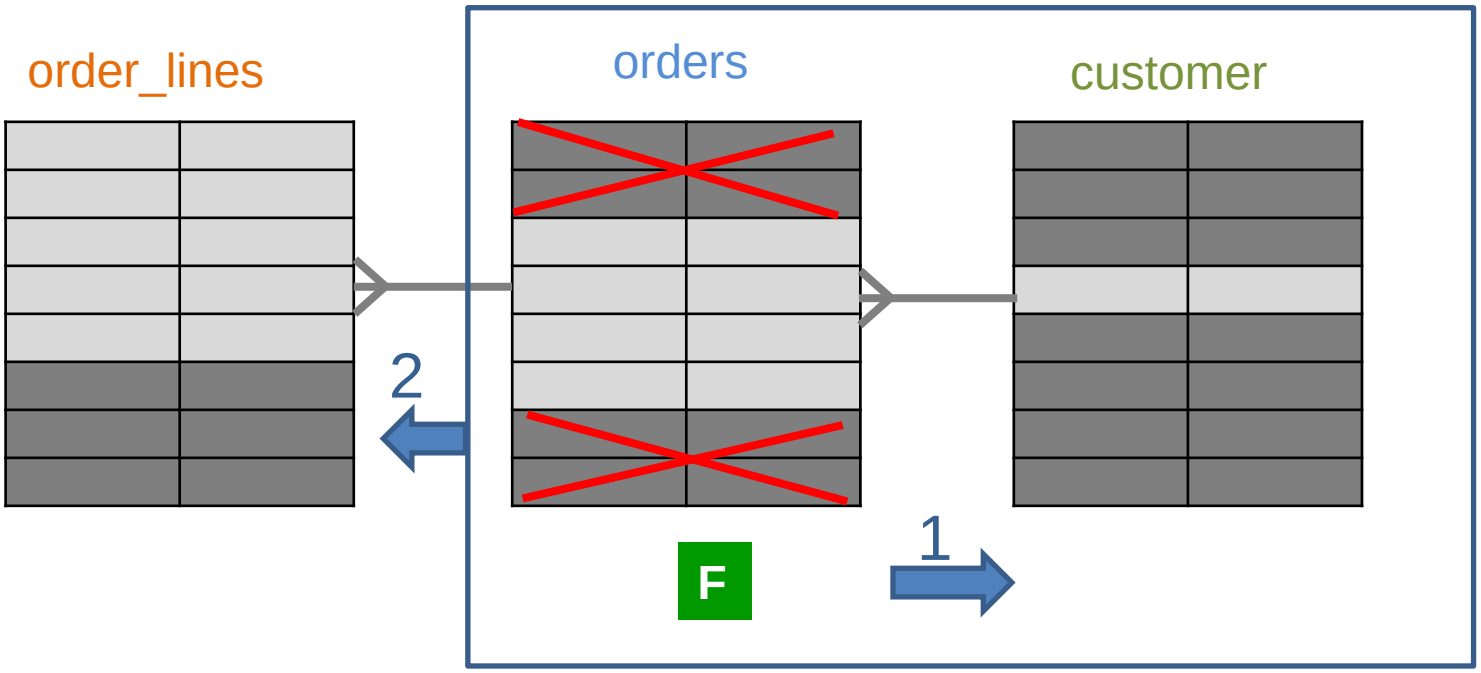


Where

order_lines.field = value

Starting with a filter on order_lines is going to eliminate work

Three table join

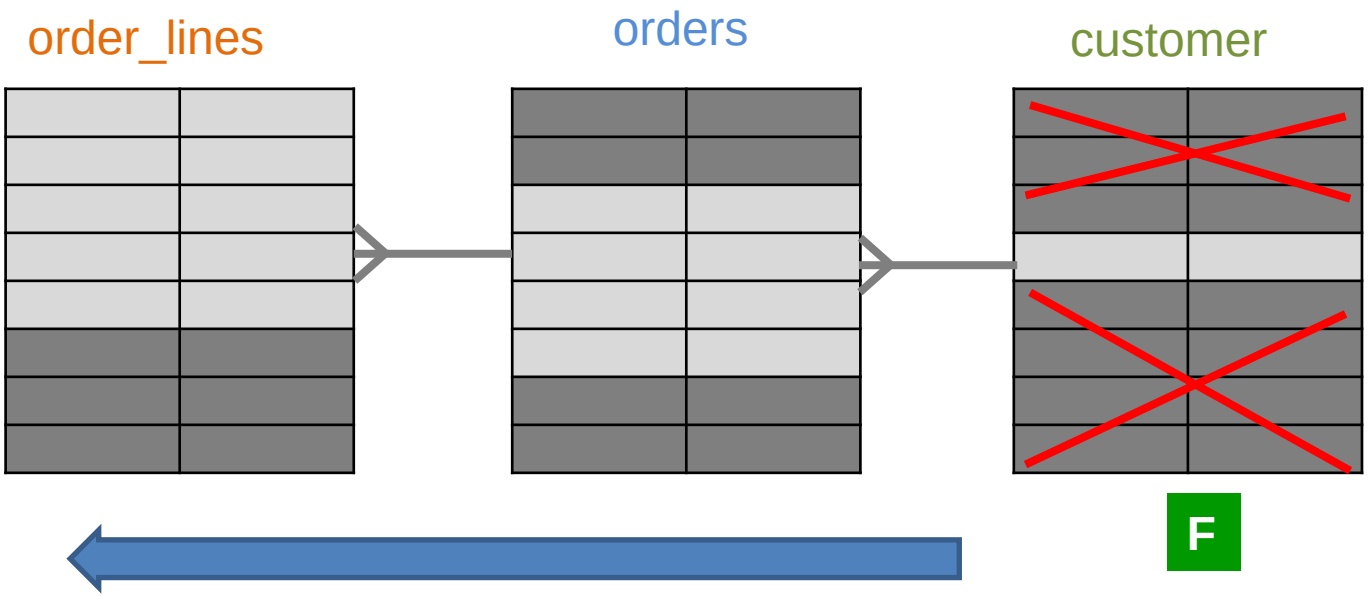


Where
orders.field = value

Starting with a Filter on orders is going to eliminate work

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

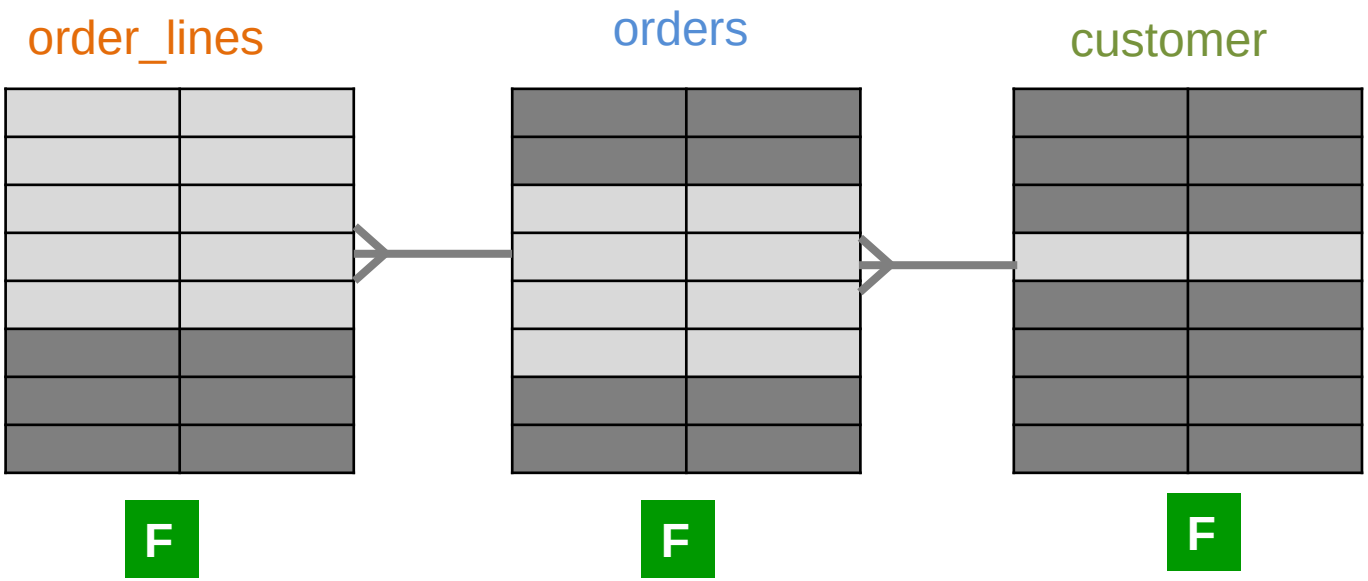
Three table join



Where
customer.field = value

Starting on a filter on customer is going to eliminate work

Three table join

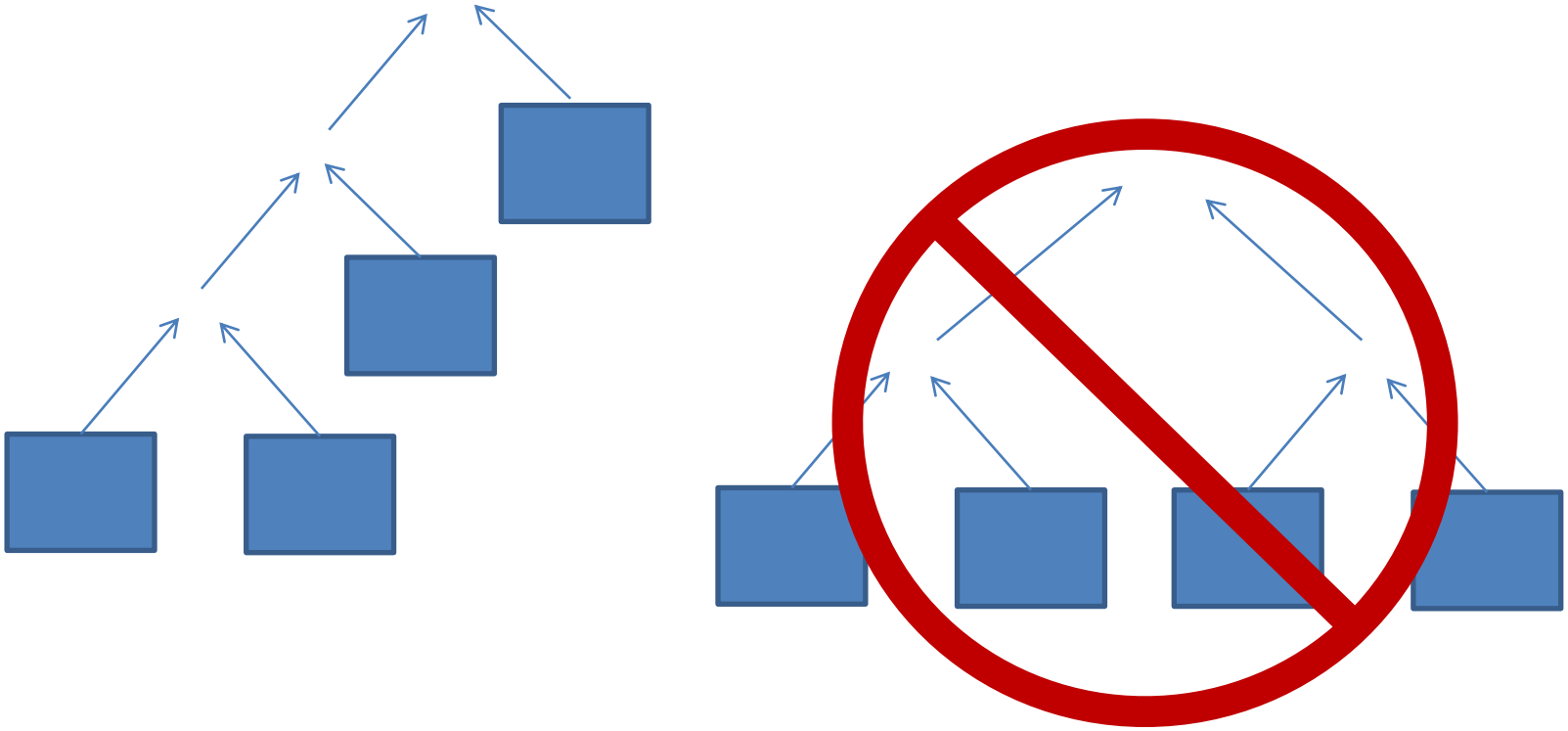


What if a filter on all three?

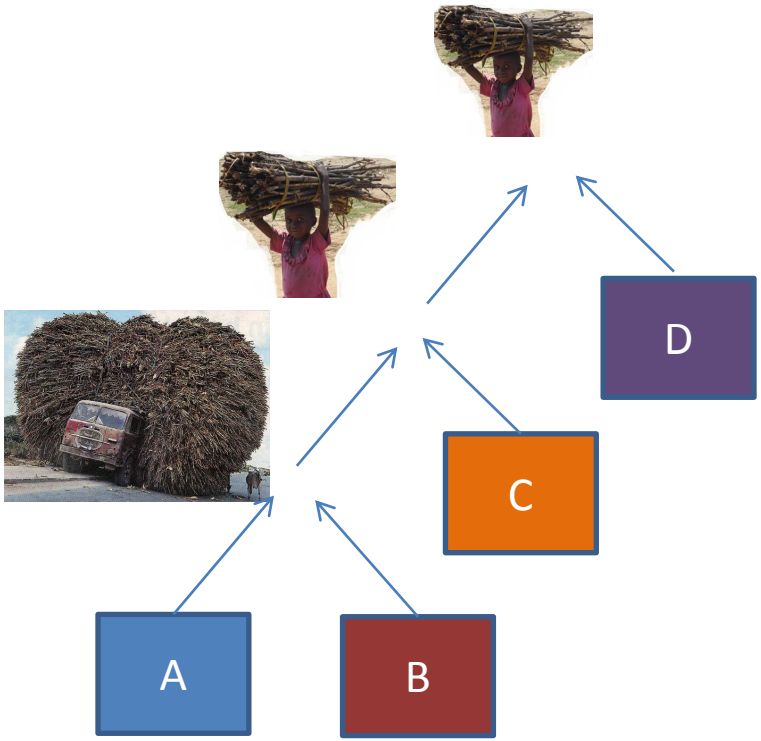
Choose the one that filter's the highest percent of the table.

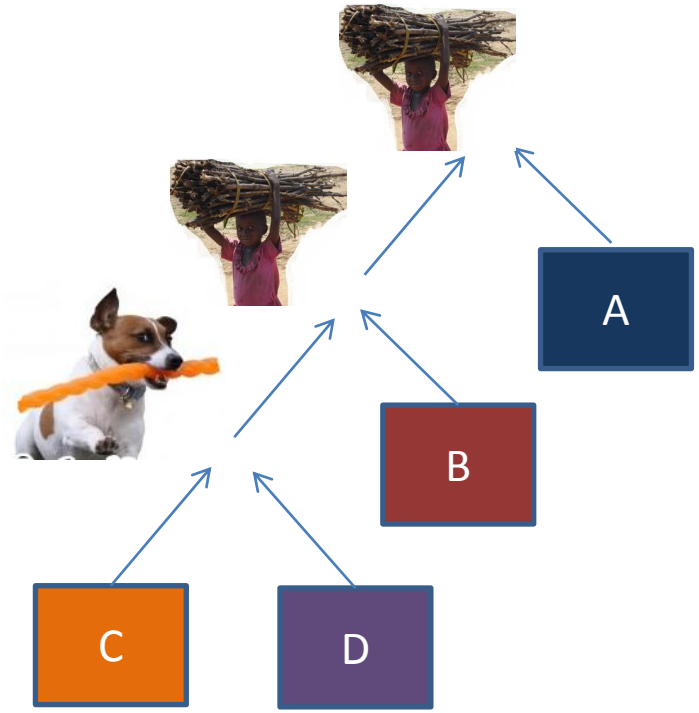
```
100% * (select count(*) from TAB where condition)
--------------------------
(select count(*) from Tab)
```

4 or more table join



4 or more table join





Put it all together : VST

1. Tables
   - drawn as nodes
2. Joins
   - drawn as connector lines
3. Filters
   - mark on each table with filter in where clause

How to VST: Tables and Joins

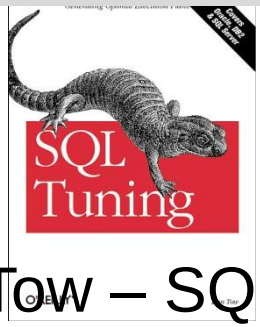
```
SELECT C.Phone_Number, C.Honorific, C.First_Name, C.Last_Name,
       C.Suffix, C.Address_ID, A.Address_ID, A.Street_Address_Li
       A.Street_Address_Line2, A.City_Name, A.State_Abbreviation
       A.ZIP_Code, OD.Deferred_Shipment_Date, OD.Item_Count,
       ODT.Text, OT.Text, P.Product_Description, 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 OT.Code_Type = 'ORDER_STATUS'
  AND OD.Status_Code = ODT.Code
  AND ODT.Code_Type = 'ORDER_DETAIL_STATUS'
  AND O.Order_Date > :Now - 366
ORDER BY C.Customer_ID, O.Order_ID DESC, S.Shipment_ID, OD.Orde
```

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
```



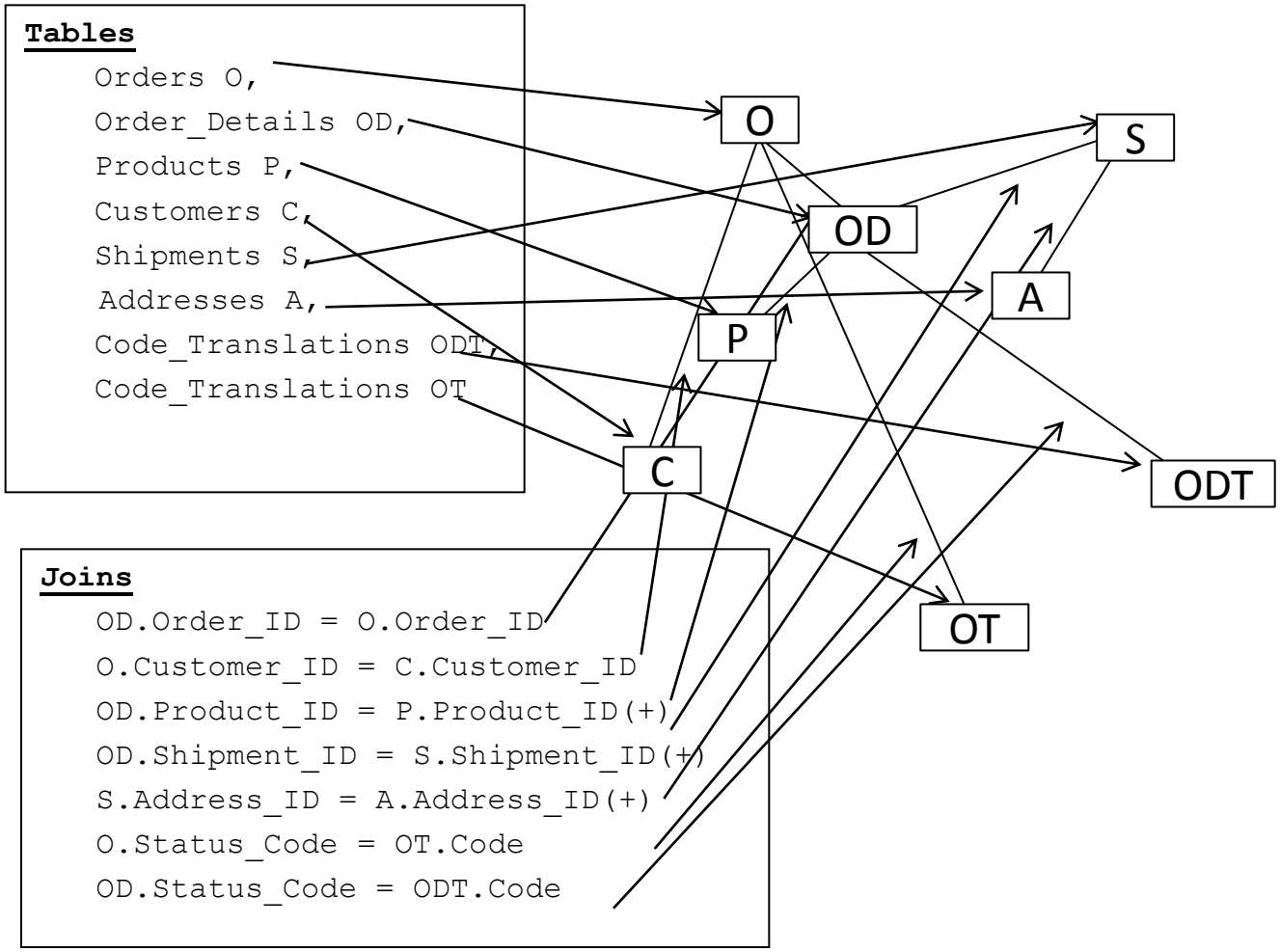
Dan Tow – SQL TUNING

Filters

```
WHERE UPPER(C.Last_Name) LIKE :Last_Name||'%'
  AND UPPER(C.First_Name) LIKE :First_Name||'%'
  AND OT.Code_Type = 'ORDER_STATUS'
  AND O.Order_Date > :Now - 366
  AND ODT.Code_Type = 'ORDER_DETAIL_STATUS'
```

DELPHIX™

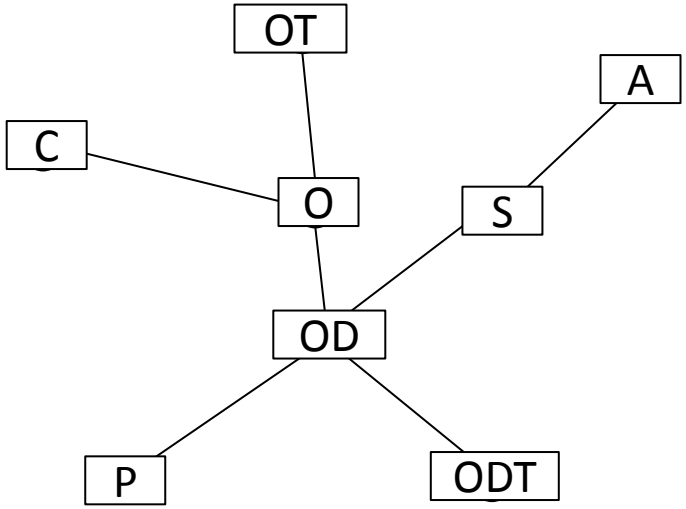
Layout tables and connections



Unstructured

Joins

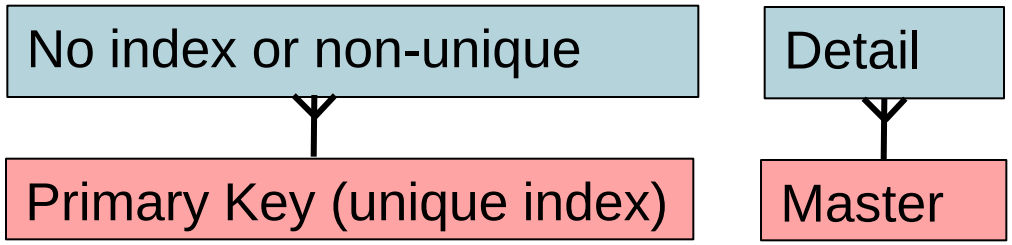
```
OD.Order_ID = O.Order_ID
O.Customer_ID = C.Customer_ID
OD.Product_ID = P.Product_ID(+)
OD.Shipment_ID = S.Shipment_ID(+)
S.Address_ID = A.Address_ID(+)
O.Status_Code = OT.Code
OD.Status_Code = ODT.Code
```

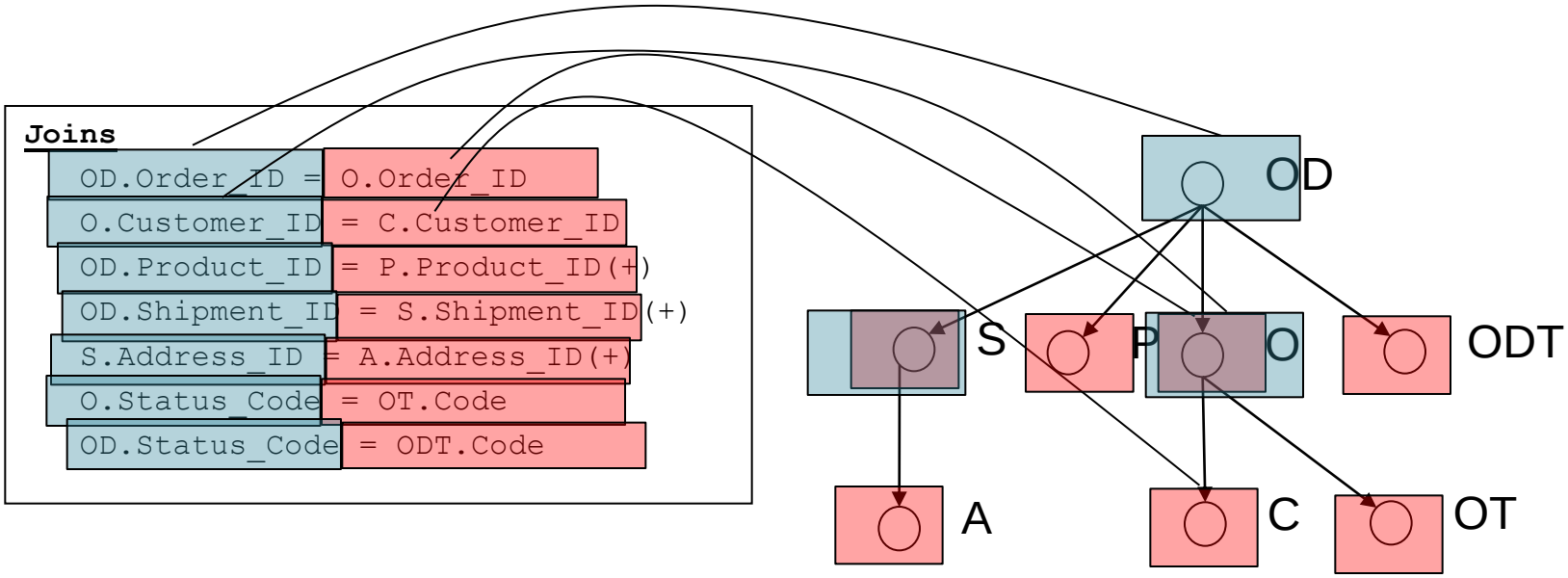


Neater, but can you do anything with it?
What's the optimal execution path?

Parents and Children

Structure the tree

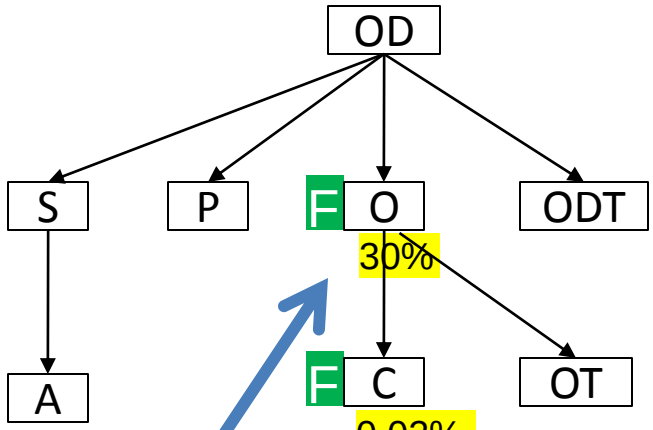




VST – filters and best path

- Filters help determine best path

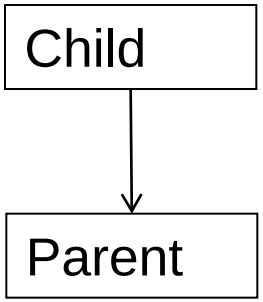
```
Filters
WHERE UPPER(C.Last_Name) LIKE :Last_Name||'%'
  AND UPPER(C.First_Name) LIKE :First_Name||'%'
  AND OT.Code_Type = 'ORDER_STATUS'
  AND ODT.Code_Type = 'ORDER_DETAIL_STATUS'
  AND O.Order_Date > :Now - 366
```

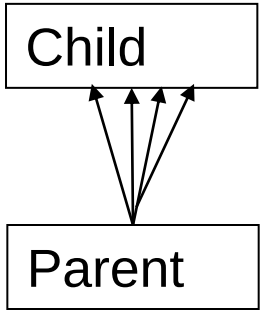



```
100% * (select count(*) from TAB where condition)
---------------------------
(select count(*) from Tab)
```

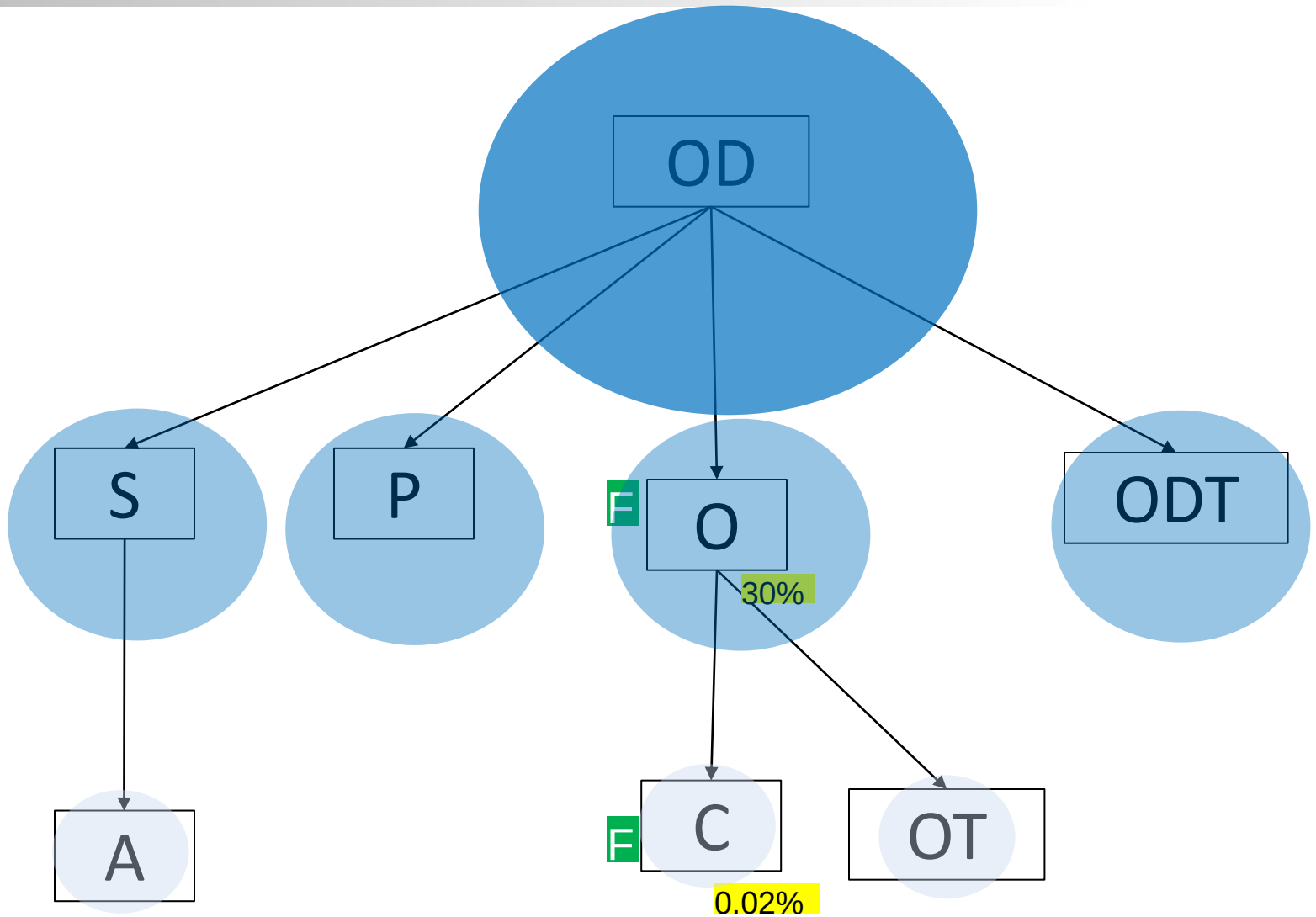
Concept:

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

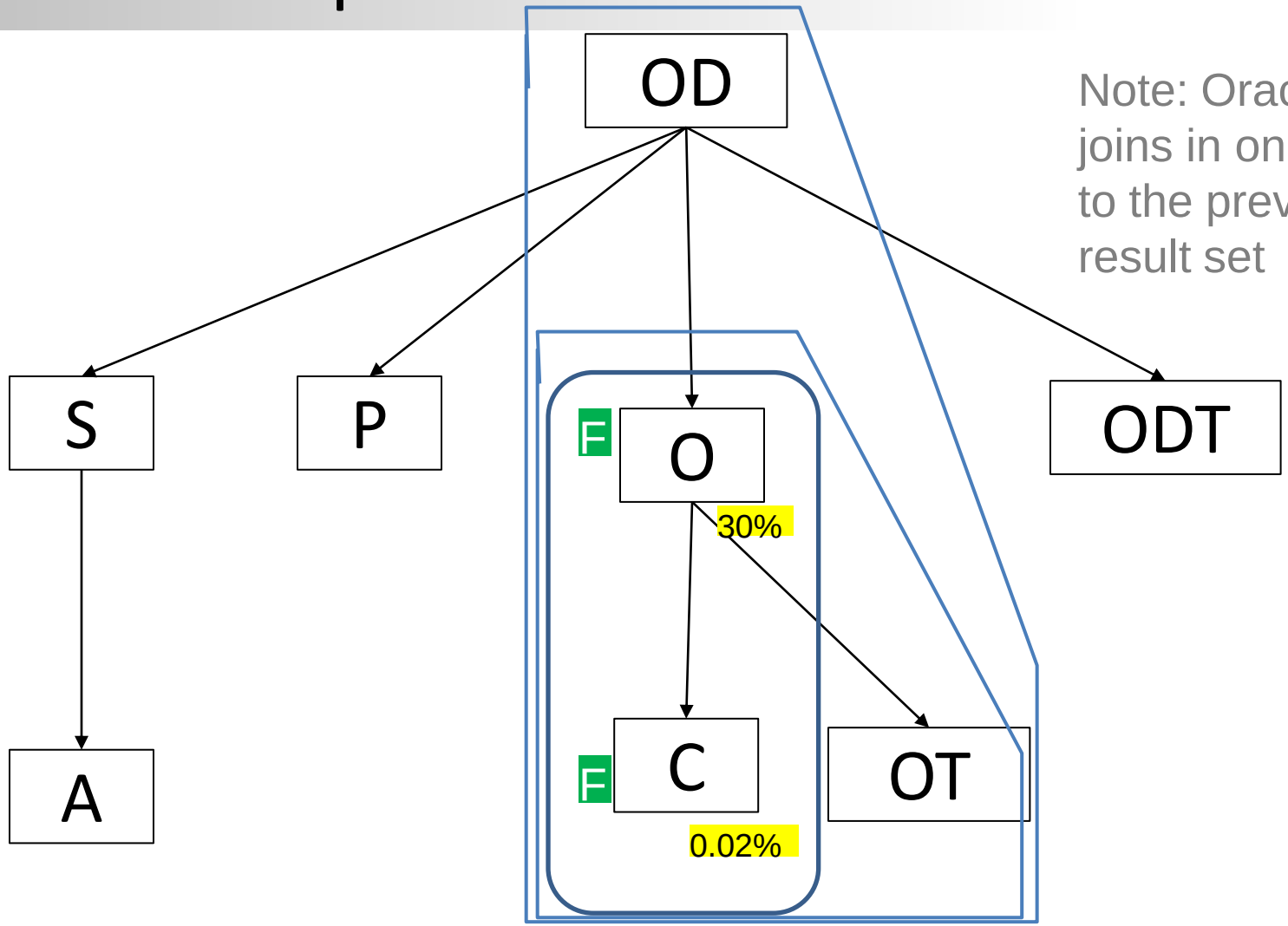




VST – best path



VST – best path

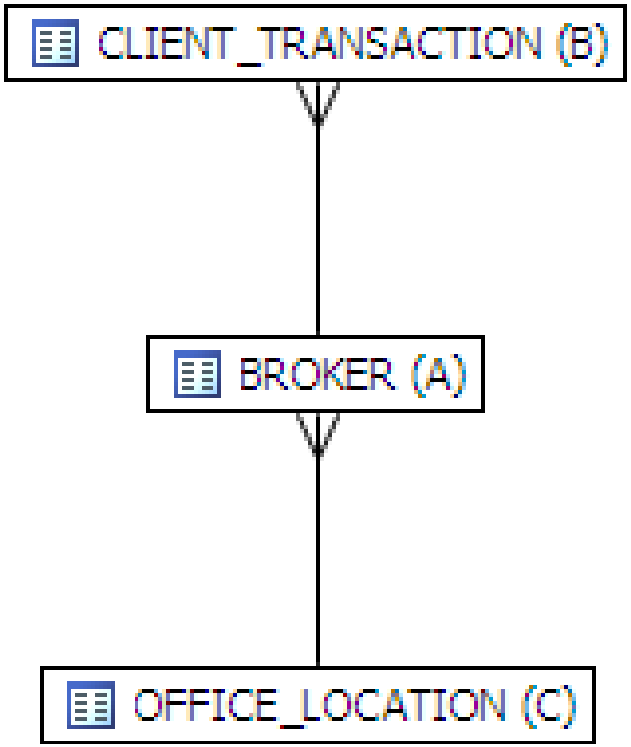


Note: Oracle only joins in one table to the previous result set

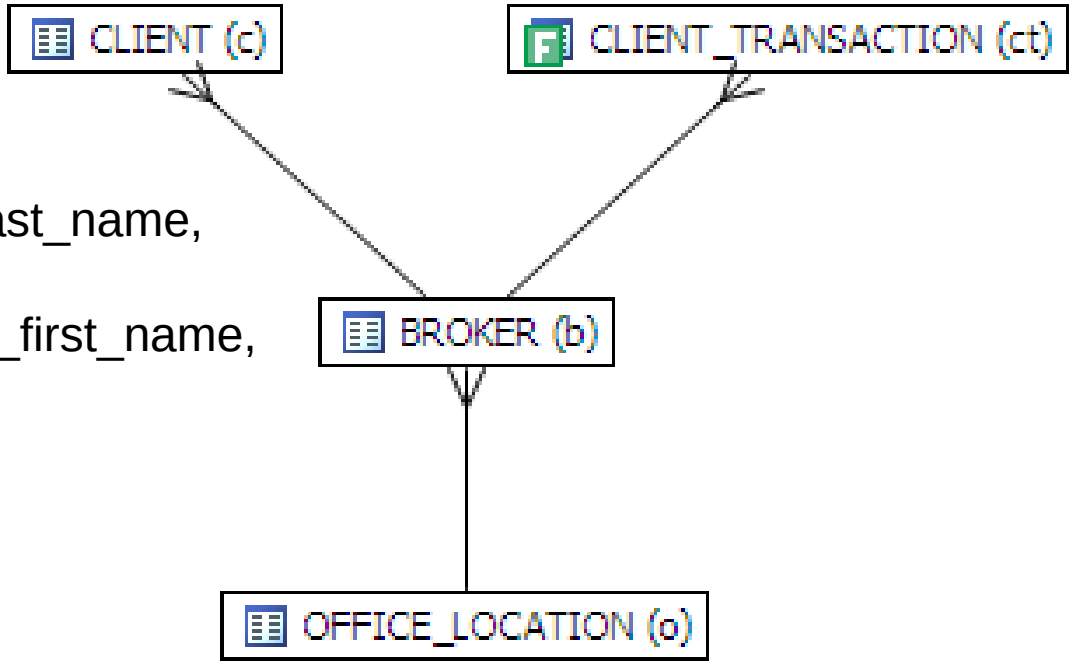
/*+ Leading (C,O,OT, OD) */

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

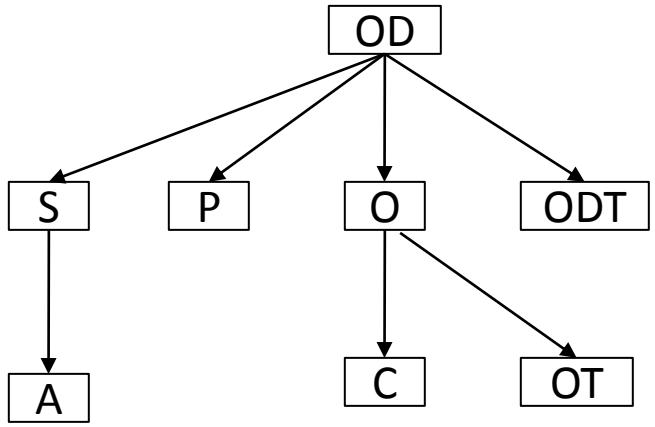
Many to many

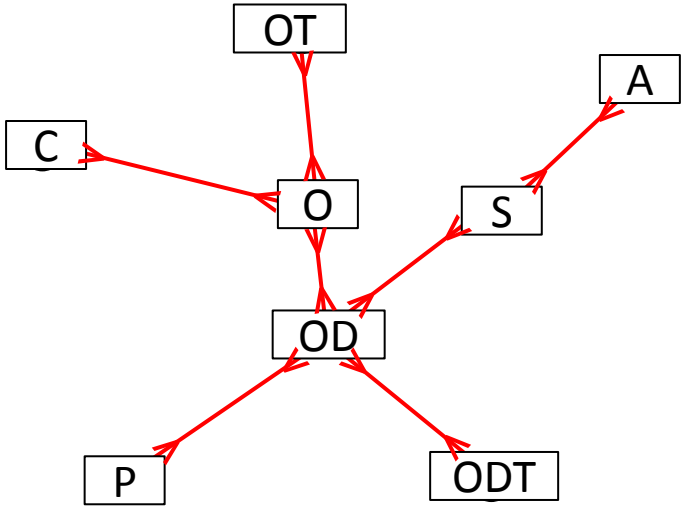
What about many to many?

Unstructured

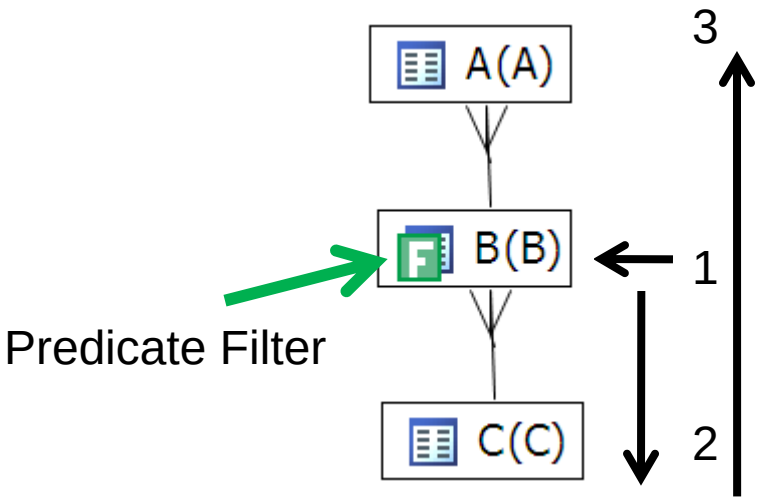
Joins

```
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
```

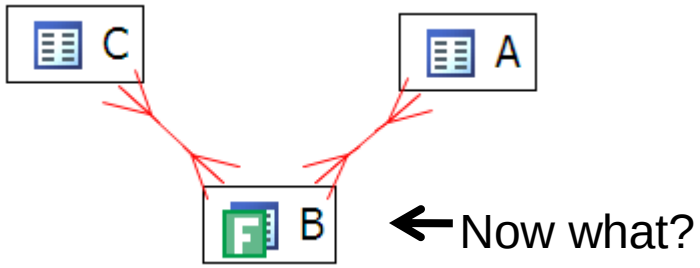




Many-to-One vs Many-to-Many



B -> C -> A

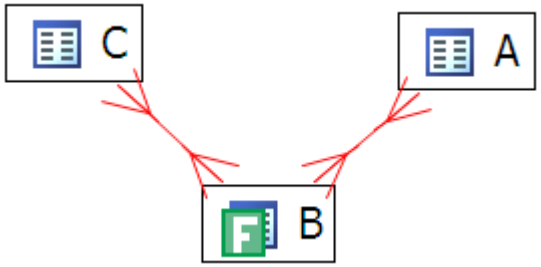


go to A or C?

Adding Constraints

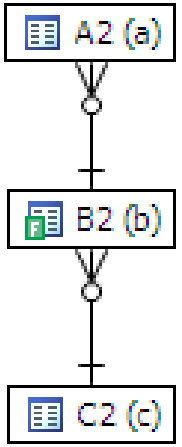
```
SELECT COUNT (*)
FROM
    b,
    c,
    a
WHERE
    b.val2 = 100 AND
    a.val1 = b.id AND
    b.val1 = c.id;
```

58 logical reads



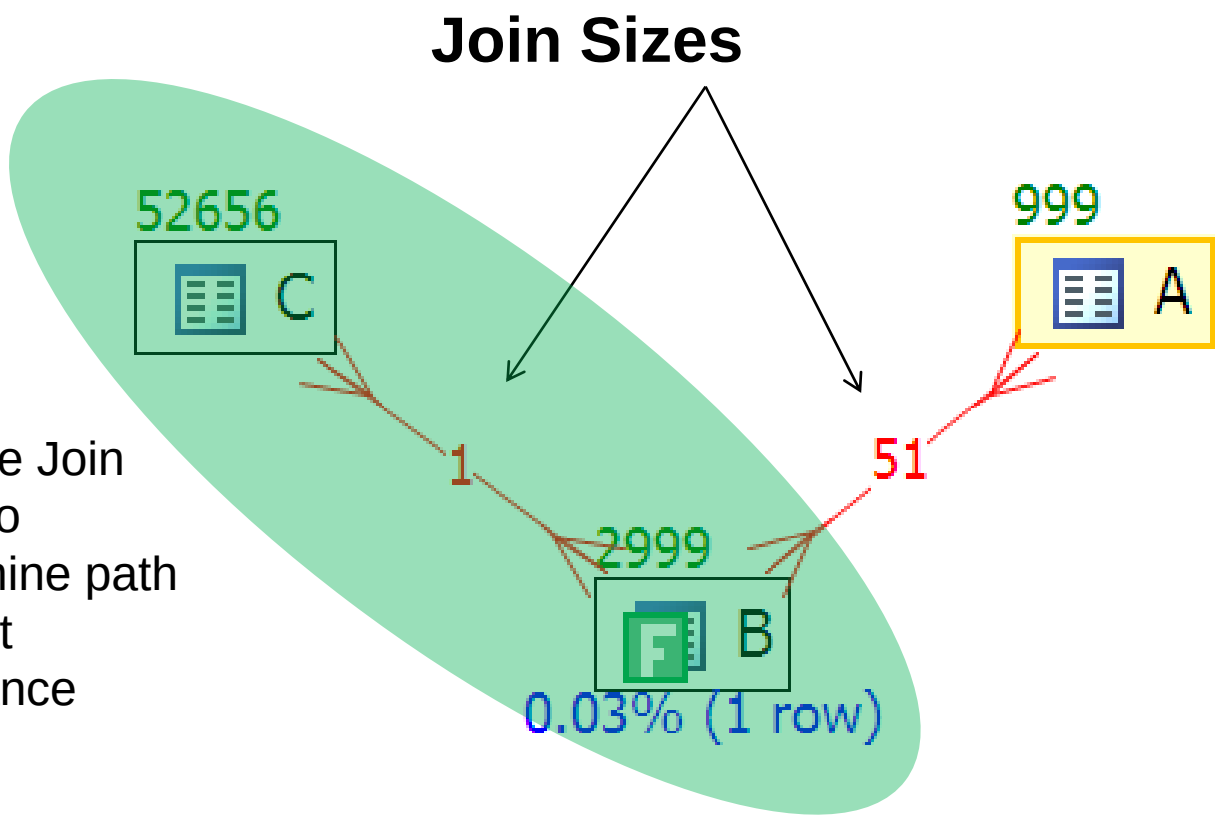

```
alter table c add constraint c_pk unique (id);
alter table b add constraint b_pk unique (id);
```

7 logical reads



Join sizes

Use the Join sizes to determine path of least resistance

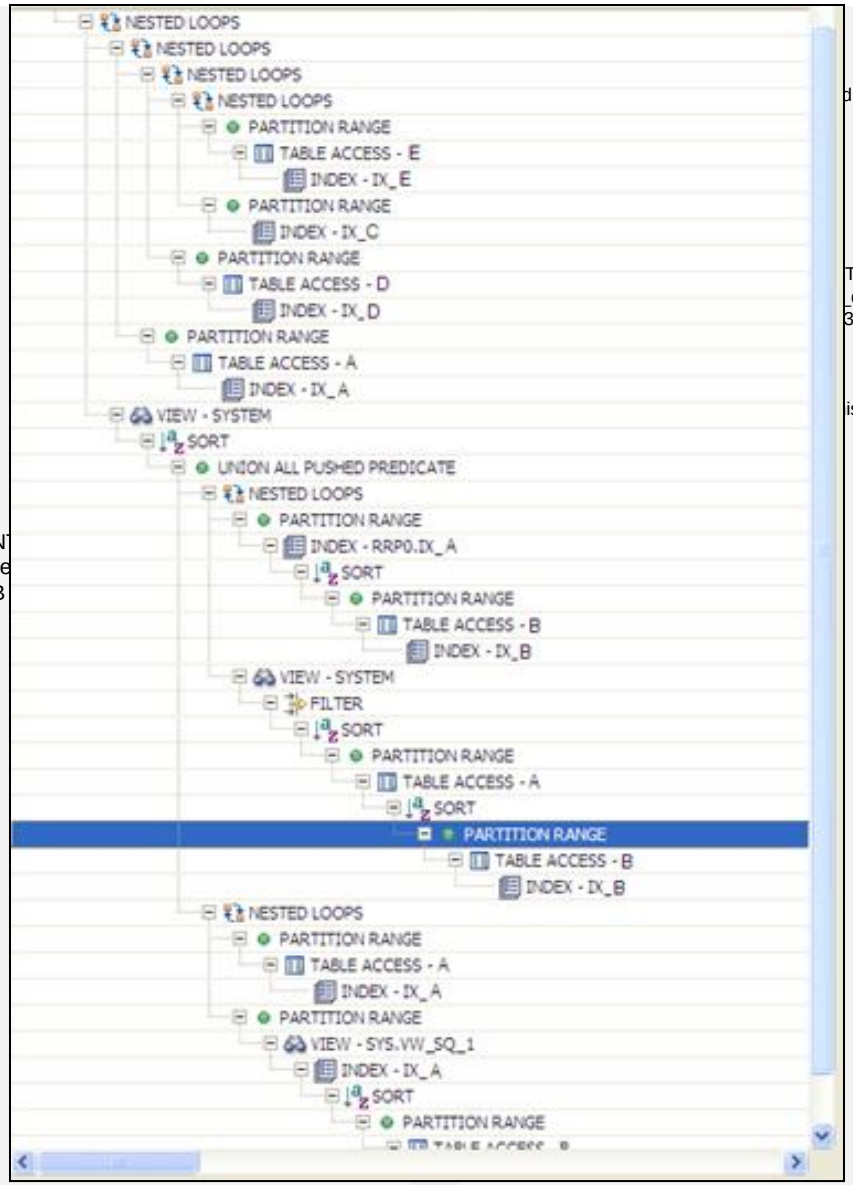


Look at 3 queries

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

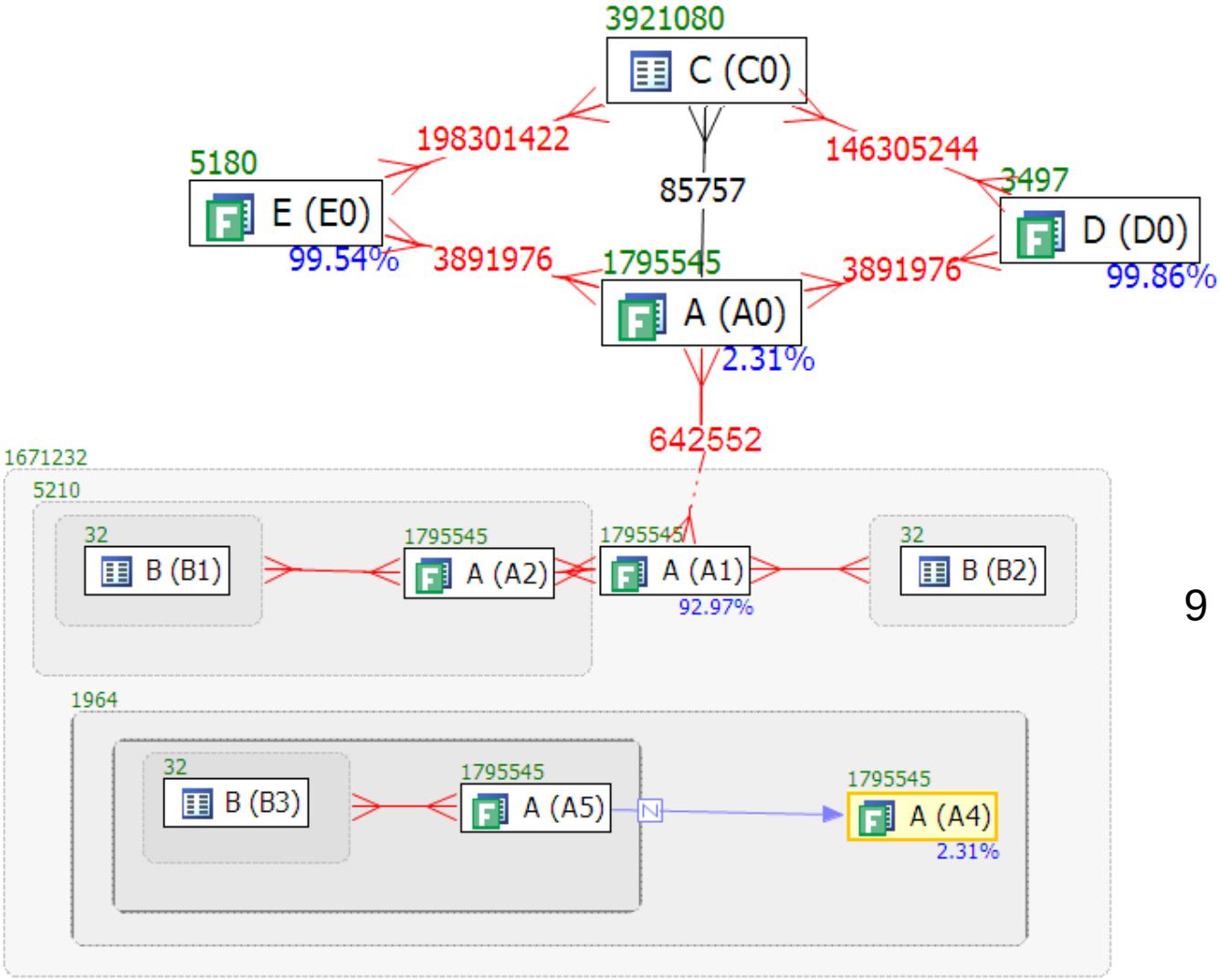
Query 1 : Over 24 hours to run

```
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_MONT
                                   MAX (plante
                                FROM FOO.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,
```



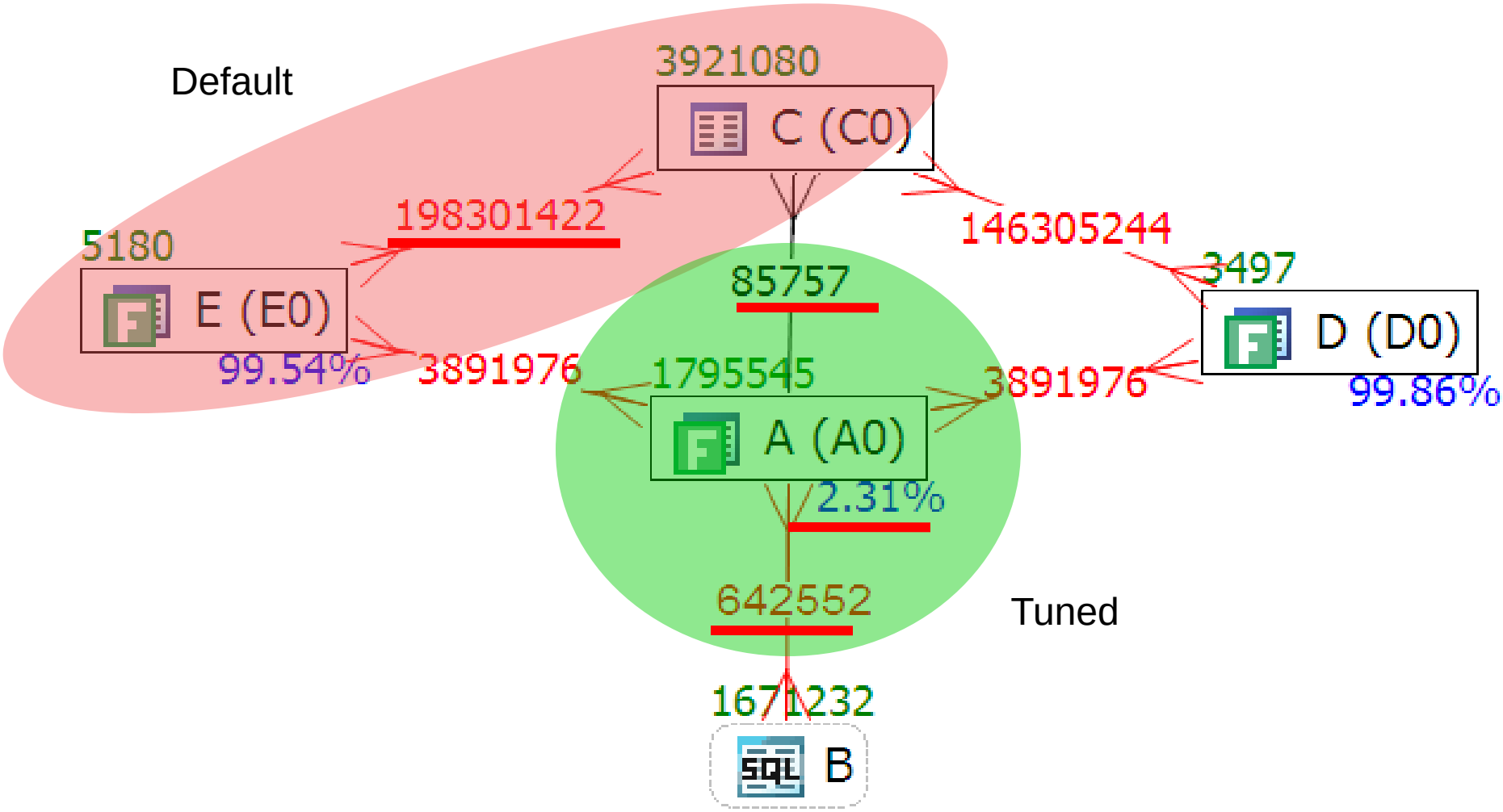

```
d A4.planted_date <'03-OCT-08' AND

T
_date)
3

is AND
```

Visual SQL Diagram

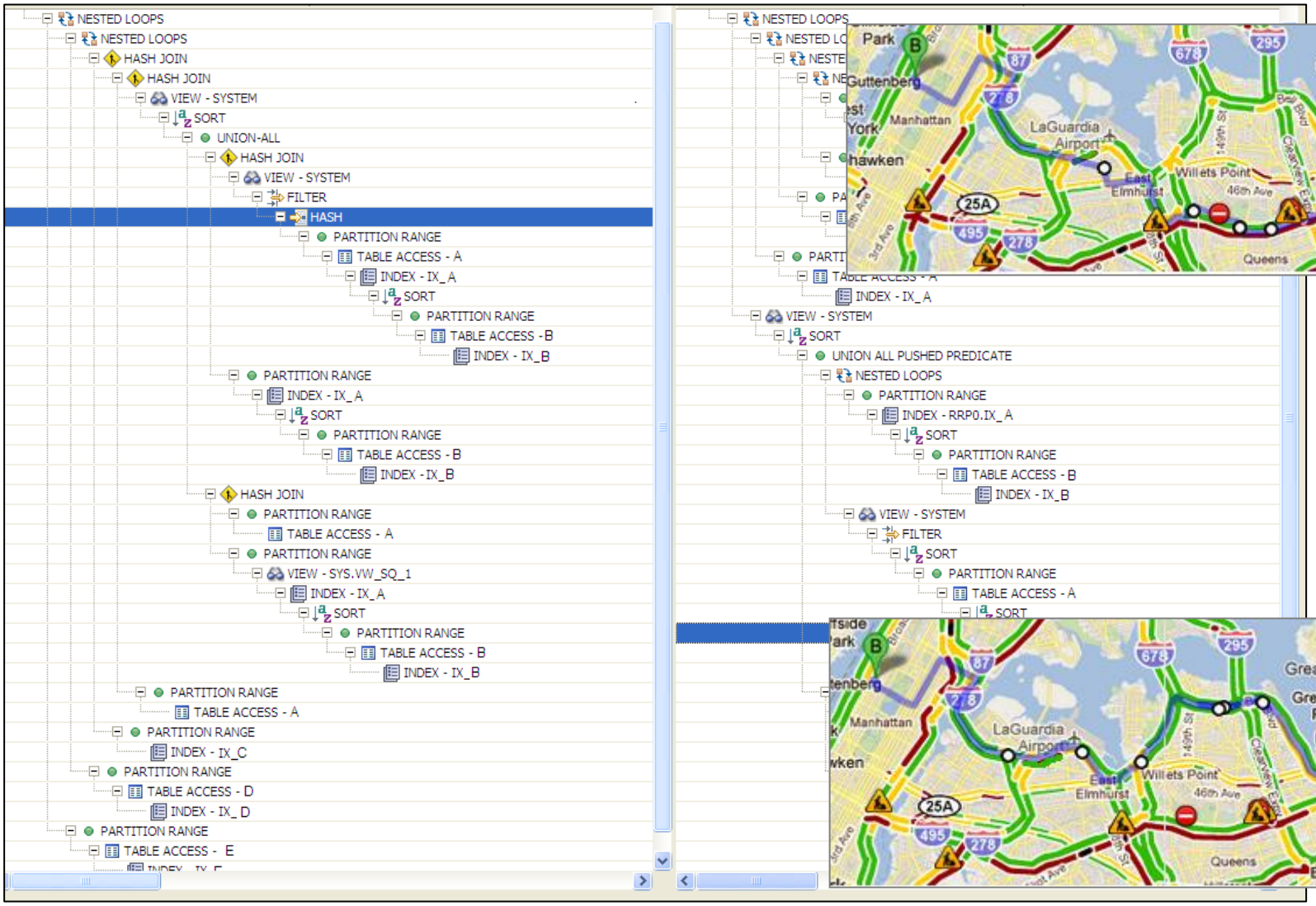


9 secs

Default vs Tuned

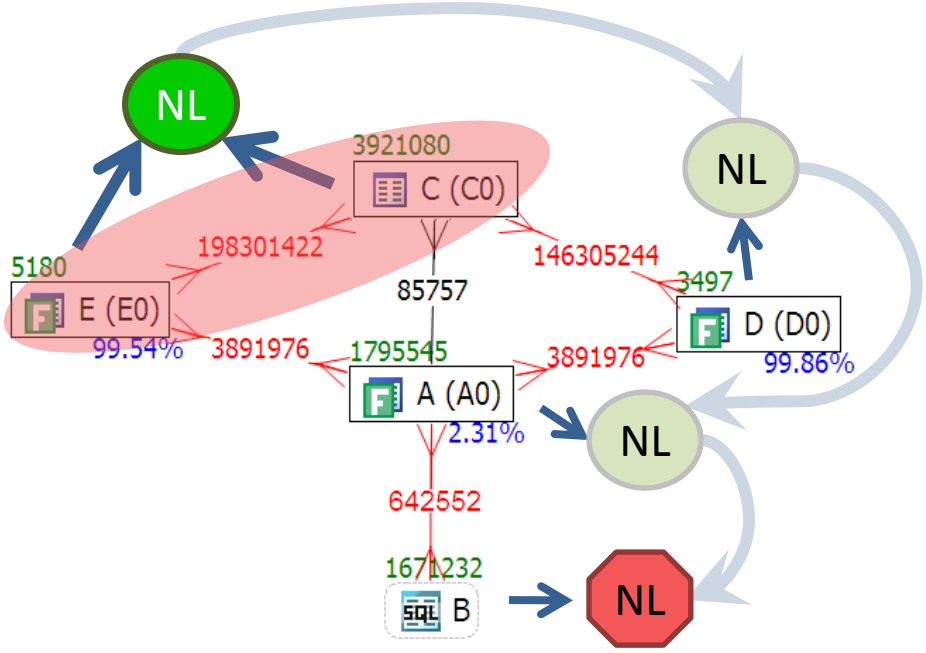


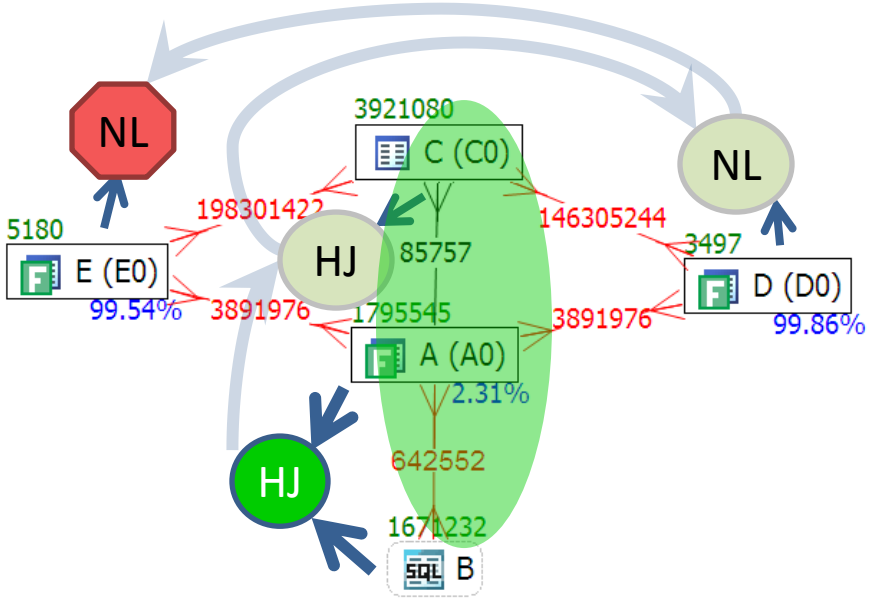
Where is the map?



DELPHIX™

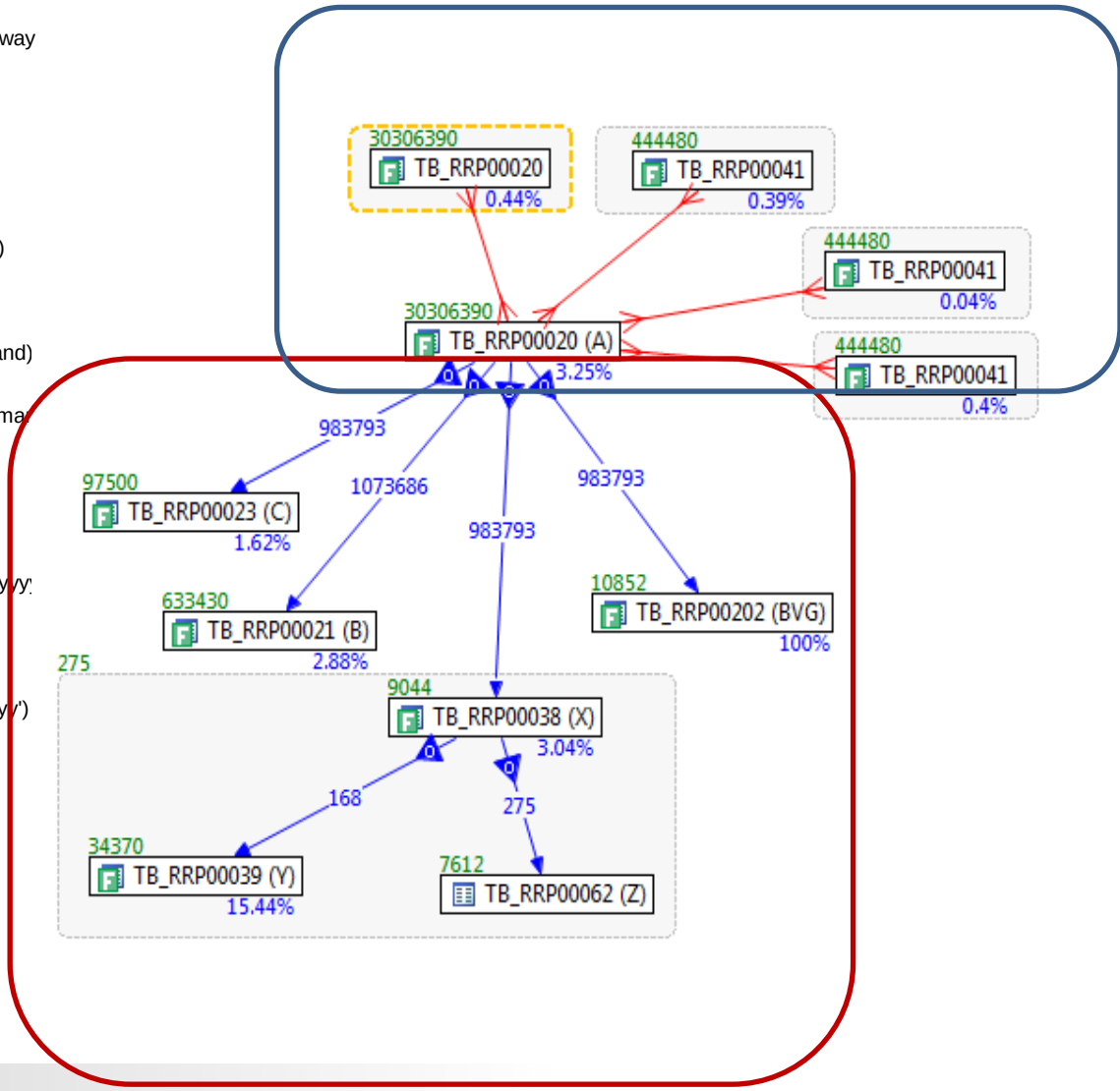
Comparing Plans : 24 hours to 5 mins





Q2

```
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.Denma
            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

682348

30306390
TB_RRP00020
0.44%

845

444480
TB_RRP00041
0.39%

825

444480
TB_RRP00041
0.04%

444480
TB_RRP00041
0.4%

12

30306390
TB_RRP00020 (A)
3.25%

983793

97500
TB_RRP00023 (C)
1.62%

1073686

633430
TB_RRP00021 (B)
2.88%

983793

983793

10852
TB_RRP00202 (BVG)
100%

275

9044
TB_RRP00038 (X)
3.04%

168

34370
TB_RRP00039 (Y)
15.44%

275

7612
TB_RRP00062 (Z)

Q2

The subqueries in the select clause look like

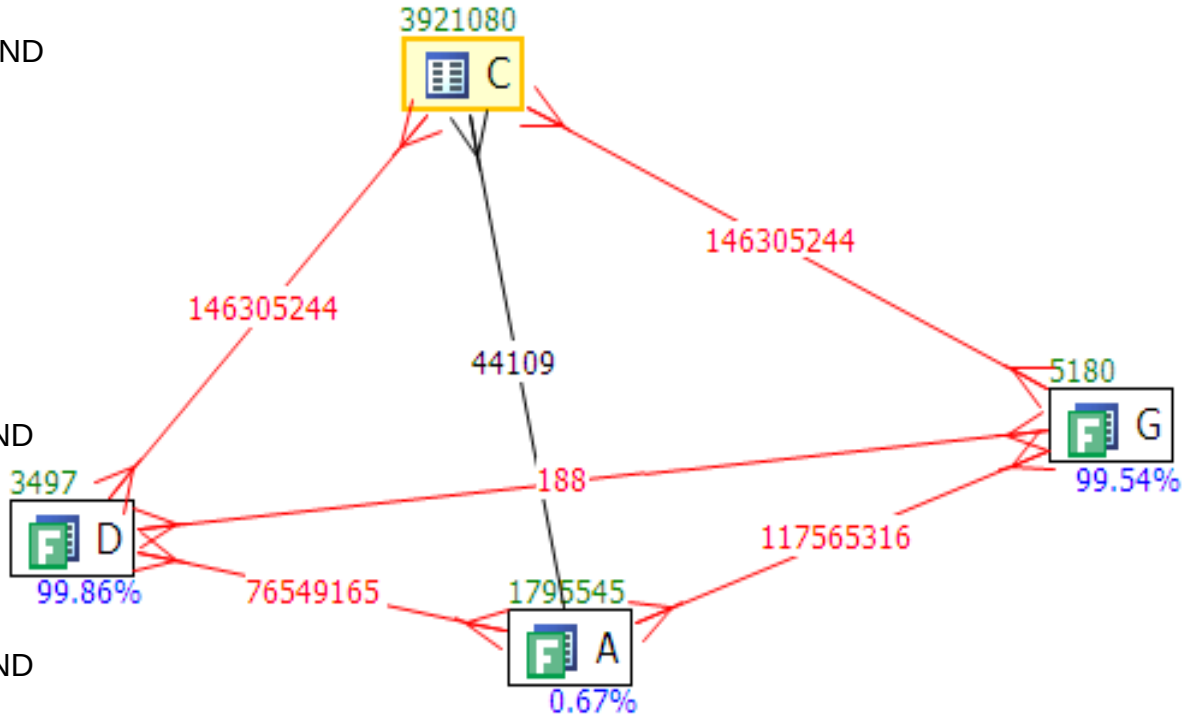
```
select CASE WHEN F.f1 IS NULL
            THEN NULL
            ELSE (SELECT X.f2
                  FROM X
                  WHERE code_vl = F.f2)
            END AS f0
from F;
```

and should be merged into the query like:

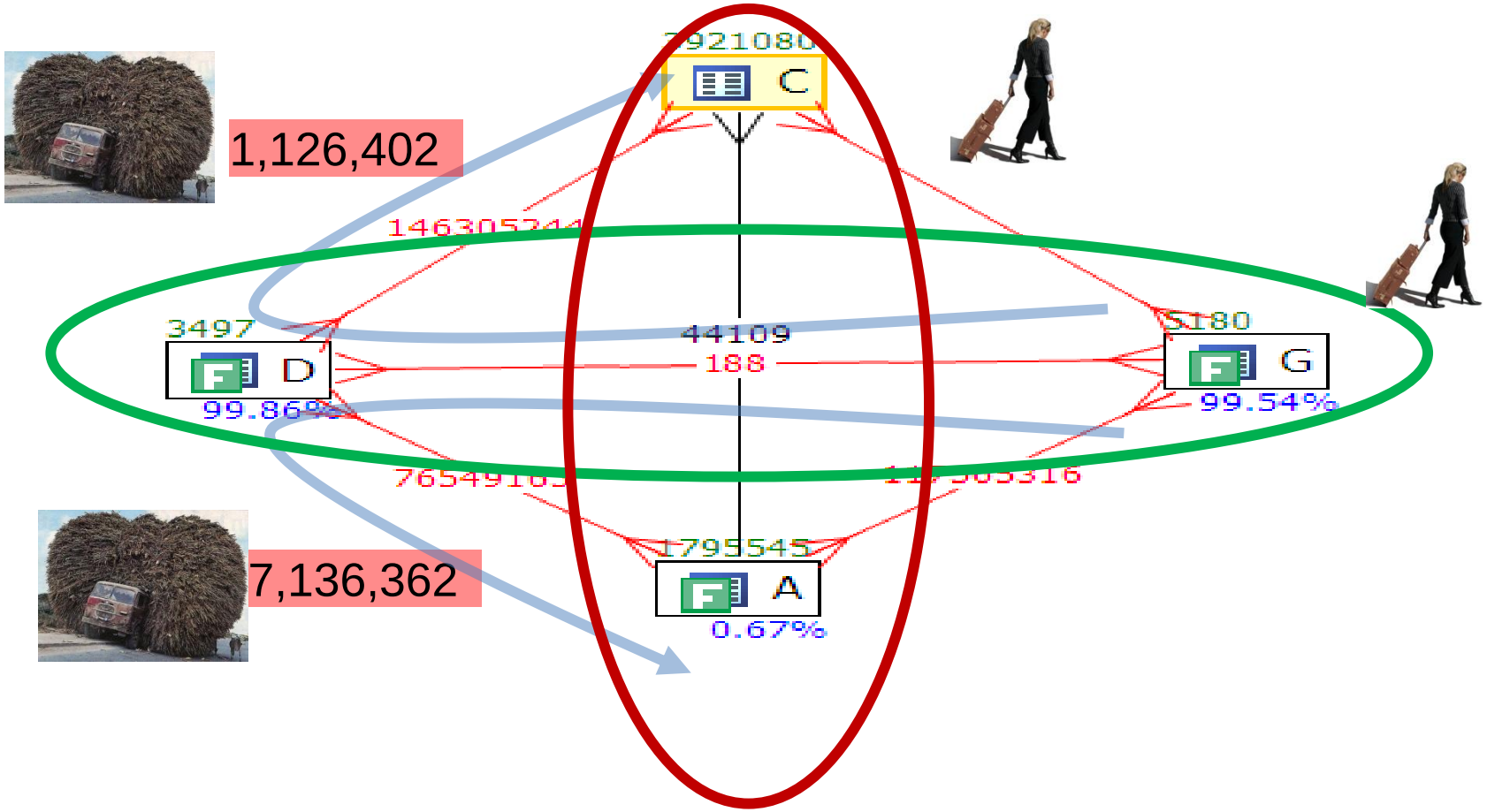
```
 select CASE WHEN F.f1 IS NULL
             THEN NULL
             ELSE ( X.f2)
             END AS f0
 from F , X
where code_vl(+) = F.f1;
```

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;
```

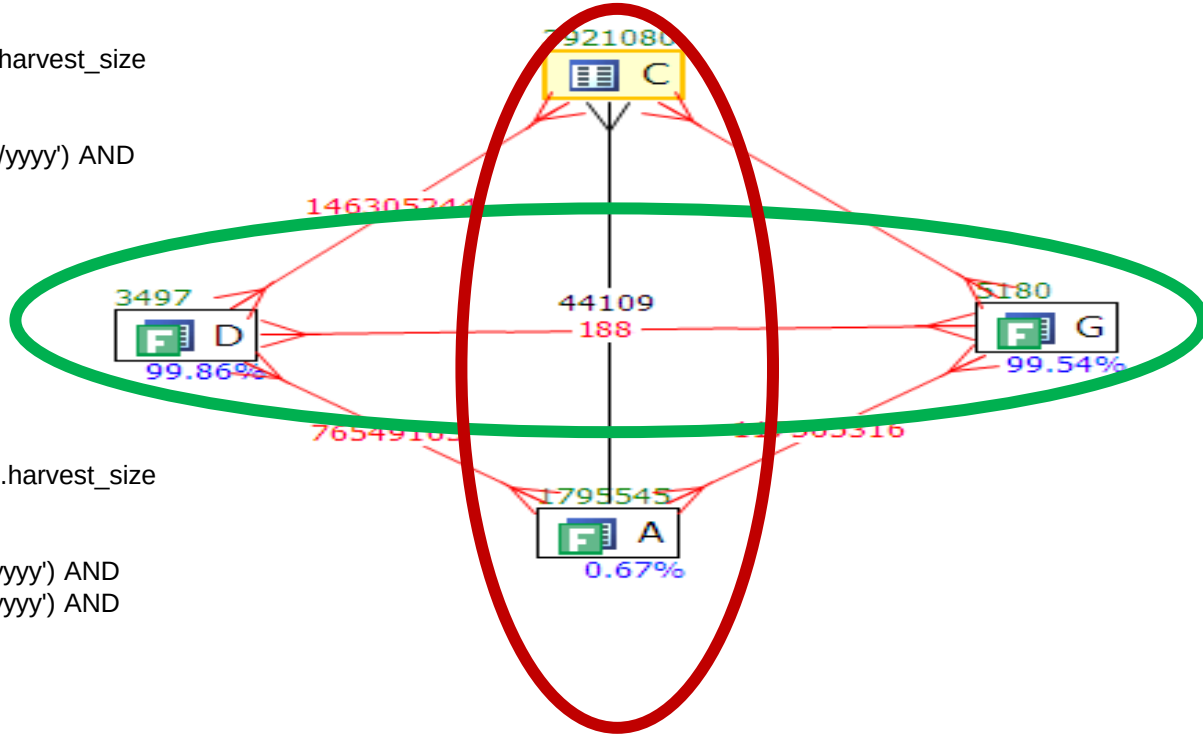


Q3: Transitivity



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

HINTS

- Leading (tab_alias , table_alias …) 10g (9i use ORDERED, not as good)
- USE_NL (table_alias) – Inner Table (2cd in xpla not driving)
- USE_HASH (table_alias) – probe into (1st in xplan)
- 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)

What tools to use to create VST diagrams?

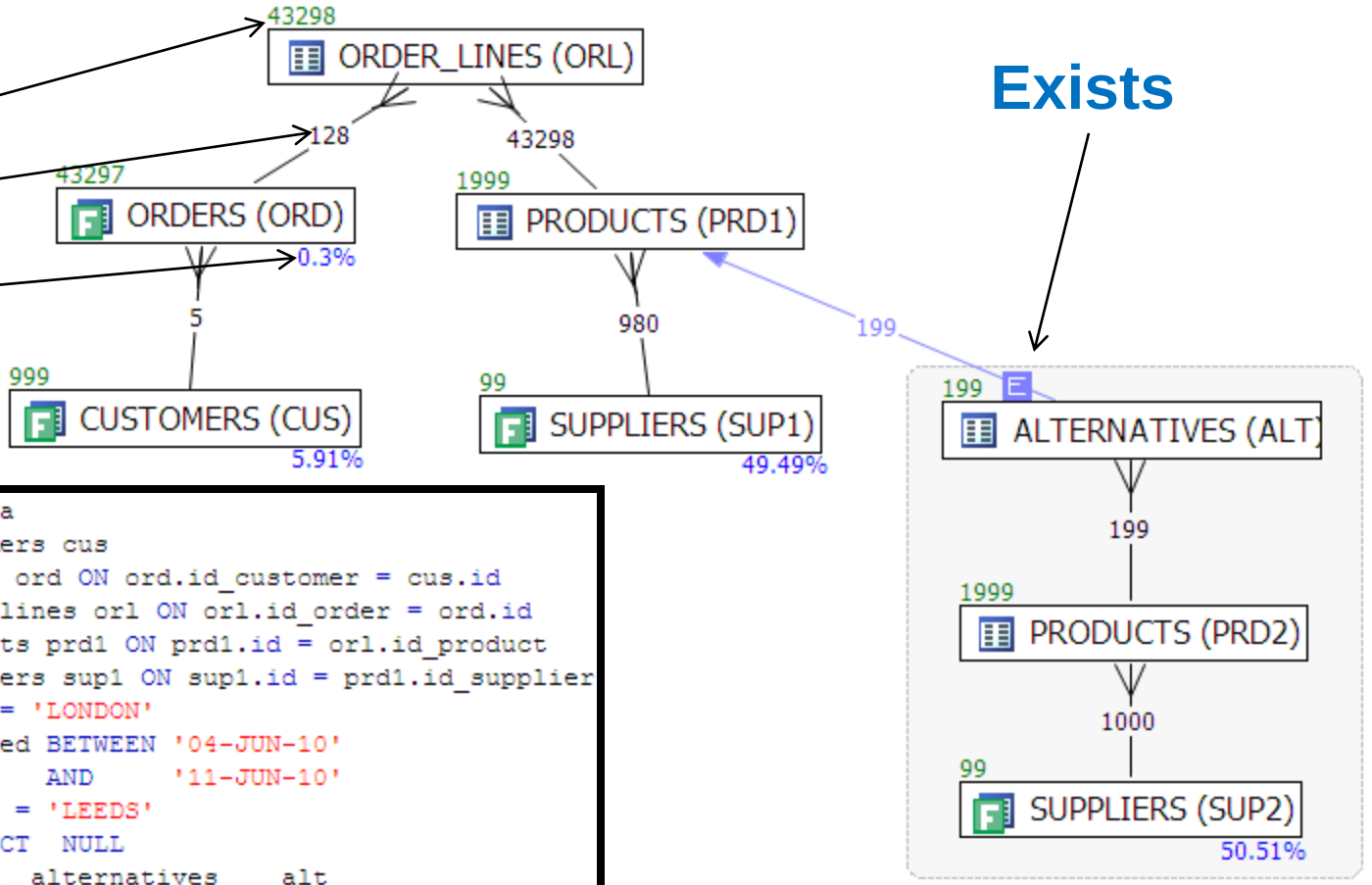
- Paper
  - Modifications messy and difficult
- PowerPoint
  - Can move things around
  - Sticky connectors
  - Easy to modify
- DB Optimizer
  - Automatic and still modifiable

Visual SQL Tuning (VST)

Table Sizes

Join Sizes

Filter Ratios

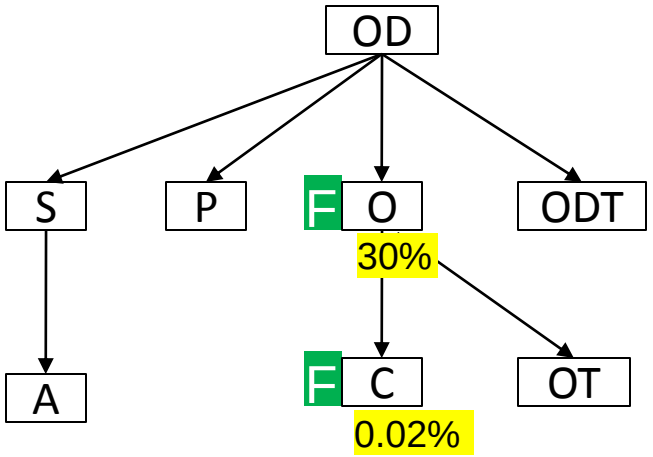



```
SELECT order_line_data
FROM           customers cus
    INNER JOIN orders ord ON ord.id_customer = cus.id
    INNER JOIN order_lines orl ON orl.id_order = ord.id
    INNER JOIN products prd1 ON prd1.id = orl.id_product
    INNER JOIN suppliers sup1 ON sup1.id = prd1.id_supplier
WHERE   cus.location = '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' )
```

DELPHIX™

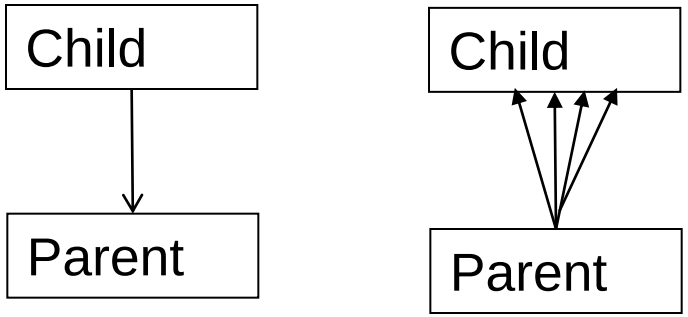
VST Steps Summary

1. Diagram tables
2. Draw connectors for each join
3. Calculate filter ratios
4. Find the table sizes
5. Calculate two table join sizes

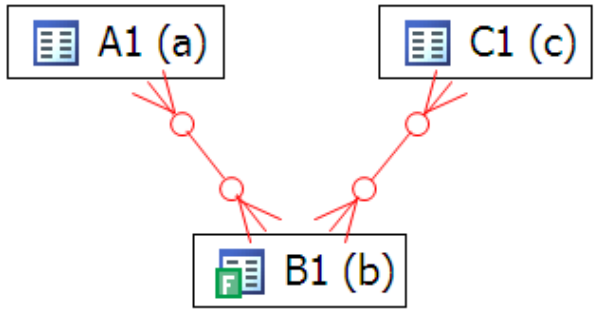


Execution Path

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



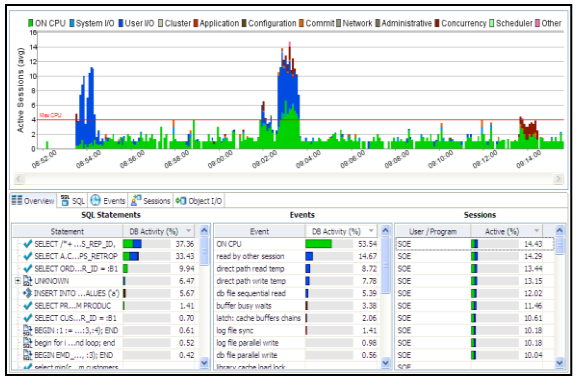
Many to Many relationships = problems



Summary

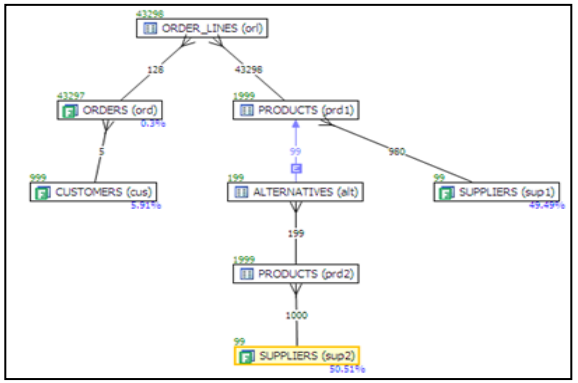
1. Database - AAS

- Profile database
- Use wait interface and graphics
- Identify machine, application, database or SQL



2. SQL - VST

- Indexes, stats, execution path
- Visual SQL Tuning



END