Memory Structures

(1) System Global Area (SGA)
- Shared SQL Area

(2) Program Global Area (PGA)
- Private SQL Area

SQL statement processing

(1) parsing
(2) binding
(3) exec
(4) fetching

(1) Library Cache Latch
(2) Shared Pool Latch
(3) Transforom
(4) Estimate
(5) Plan Generator

Plan

CBO

(1) Query Transformer

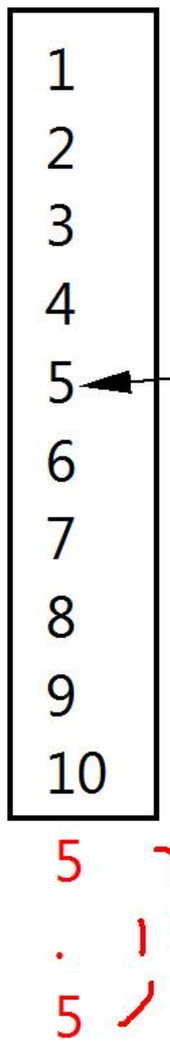
- to rewrite the original SQL statements to find out more possible access paths

- OR --> UNION ALL
- Suvquery with IN --> Join form
- etc

CBO Estimator

(1) Optimizer statistics

- rows
- index
- B*Tree or Bitmap
- NDV
- etc



Plan 1 Plan 2 Plan 3

Histogram

(1) selectivity: % of data to be filtered (0 ~ 1)

- where col = 5 10% --> 0.1

= 1/NDV (assumption: even distribution)

= 1/10

(2) cardinality: number of rows to be processed

- 1 row

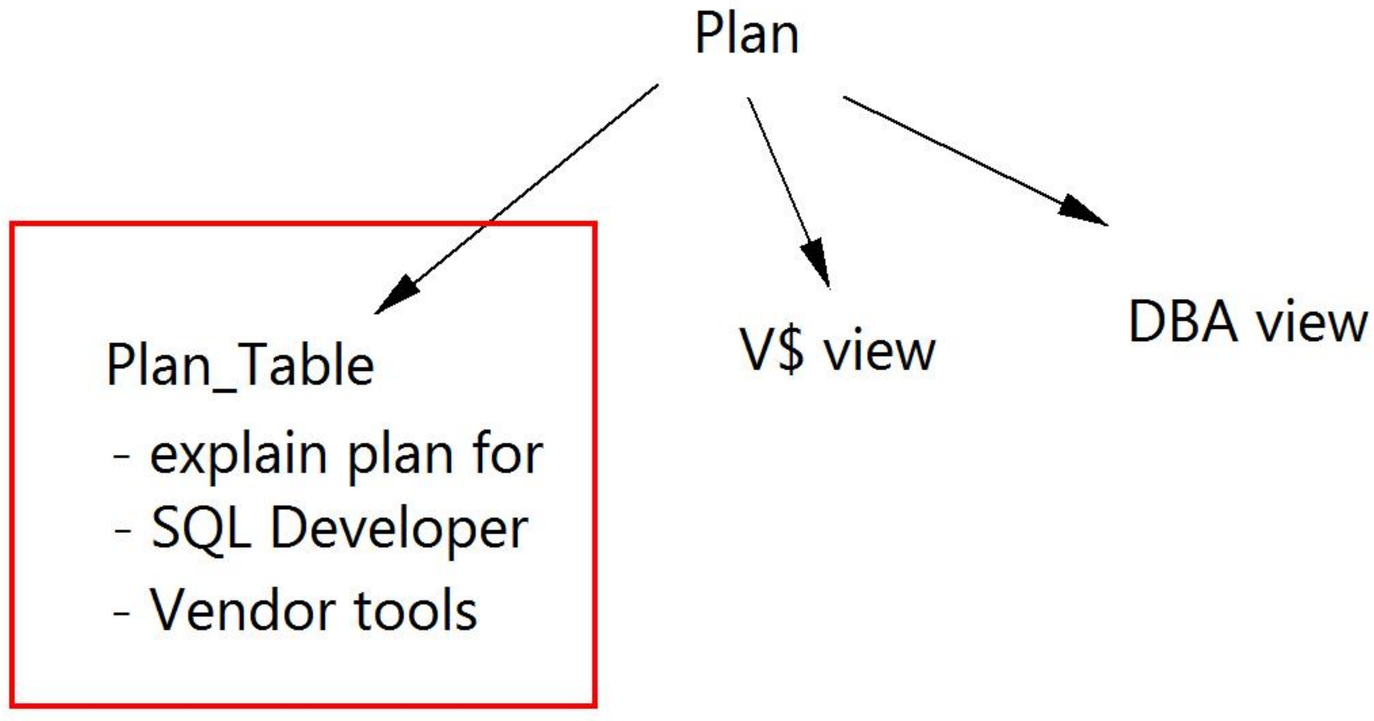
= rows x selectivity

= 10 x 1/10

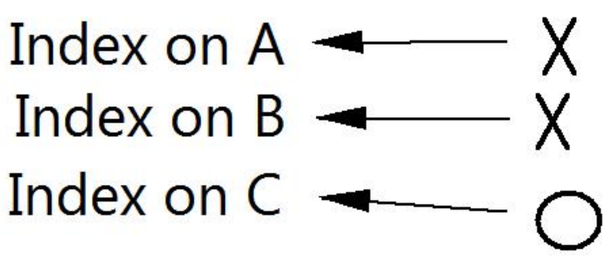
= 1 row → Cost (System Statistics)

Plan Cost Unit = Estimated Time to Read a Single Block

dbms_xplan
- display
- display_cursor
- display_awr



A, B, C



Index on (A, B, C)

Single Column Index vs Composite Column Index
- Which one is better? Single Column Index

If multiple columns appear in the WHERE clause, then you can possible take advantage of
Composite Key Index

How should I create a composite key index?
- Index on (A, B, C)
- What should be in A position?
  (1) The most selective column should be in A.
  (2) If all three columns are equaly selective, then choose
  the column with the best selectivity.

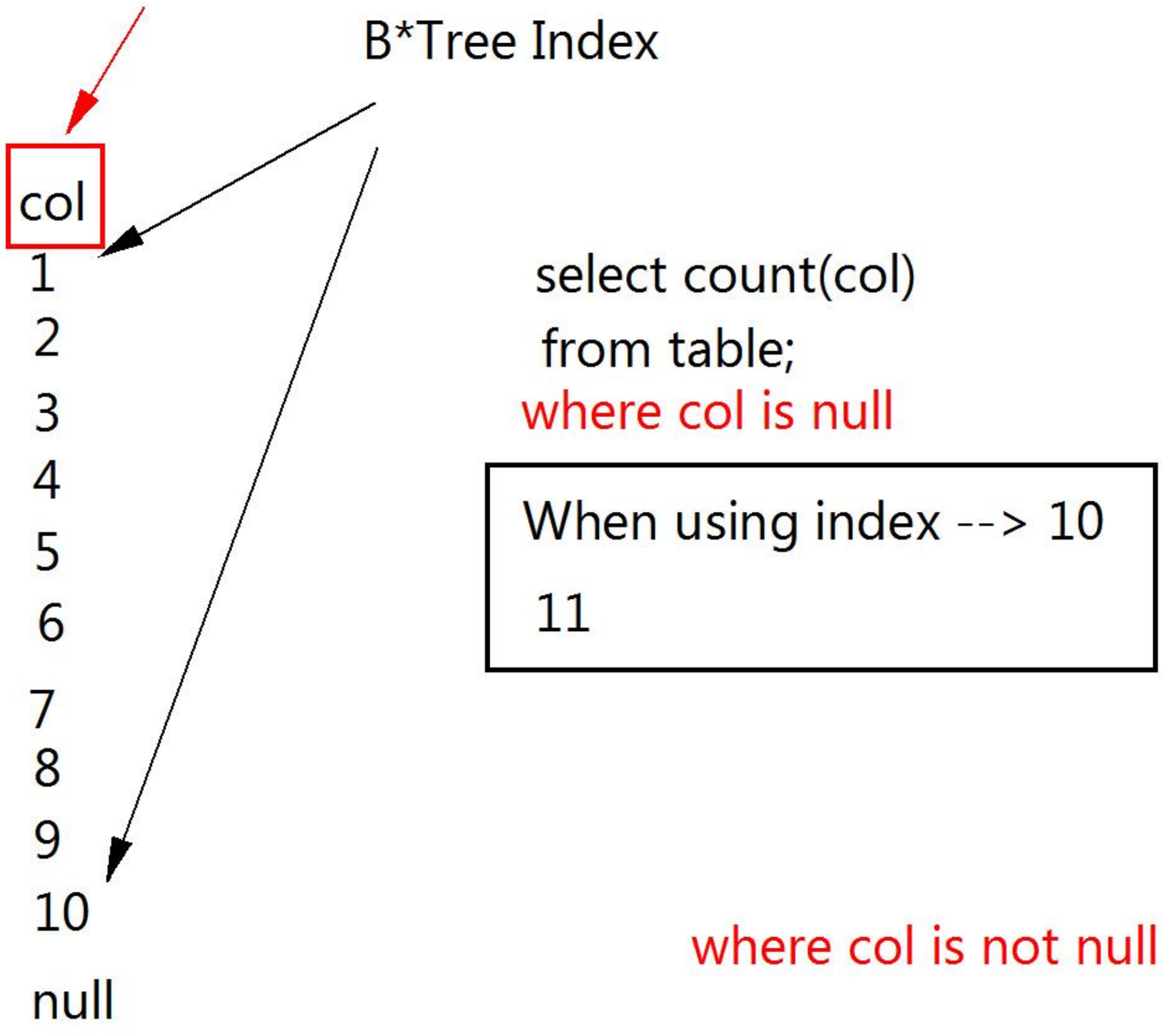
frequently

unique or non unique

gender

postal_code

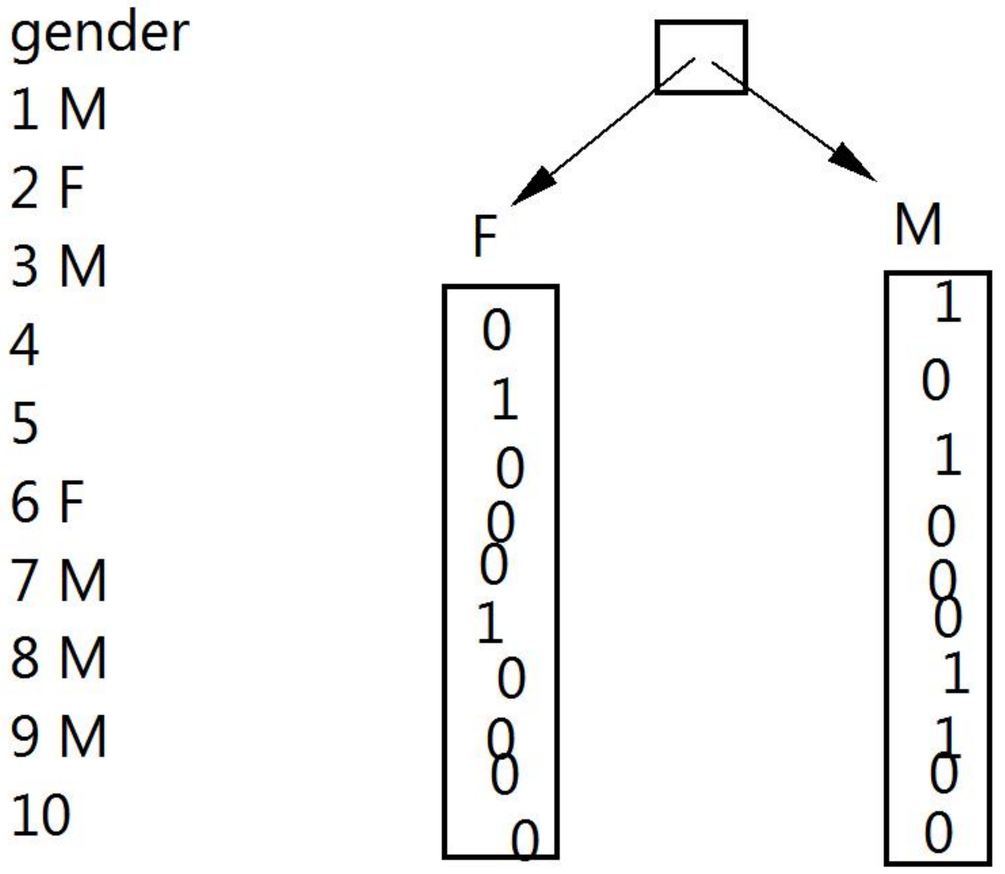
Index Skip Scan

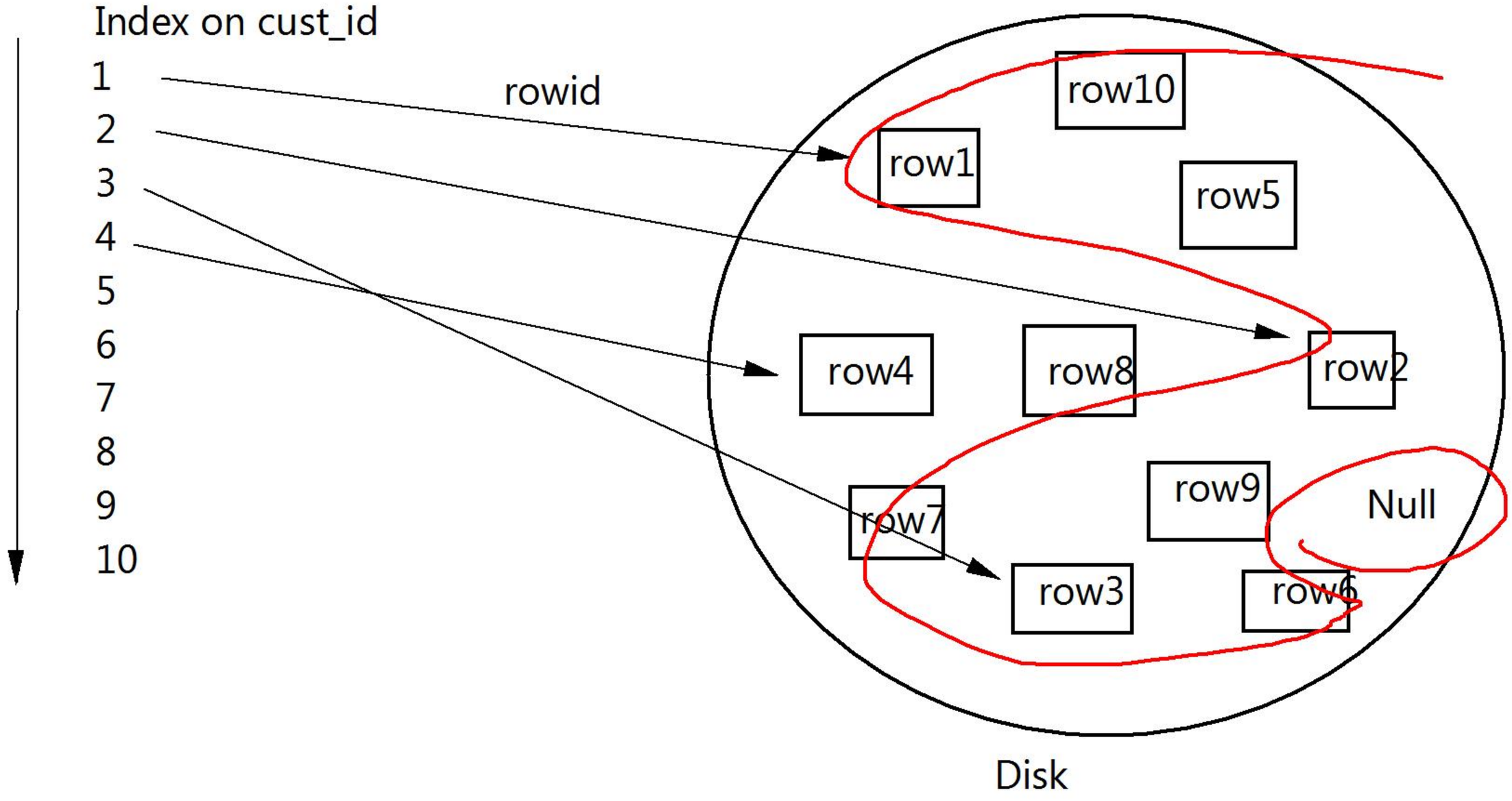


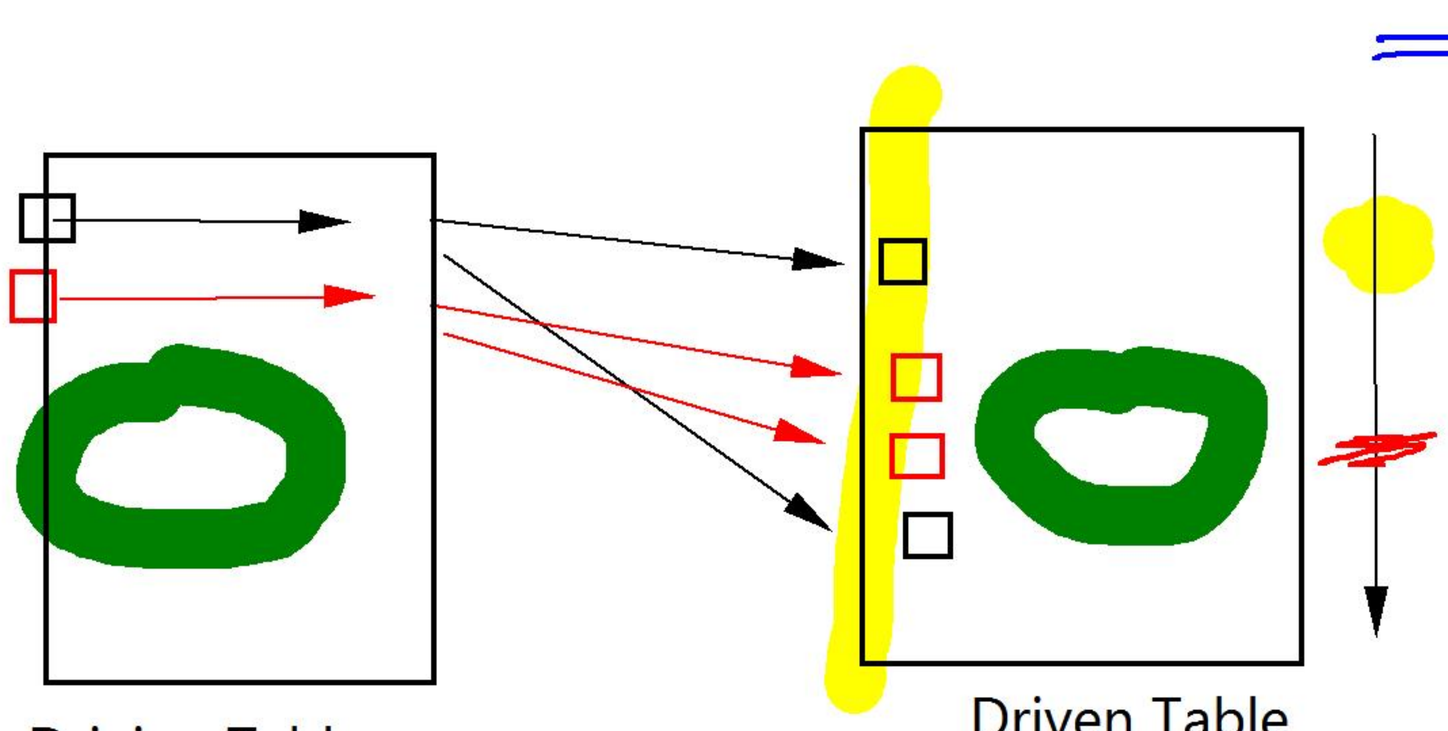
select count(col)
from table;
where col is null

When using index --> 10
11

where col is not null




Index on cust_id
1
2
3
4
5
6
7
8
9
10
rowid
row10
row1
row5
row4
row8
row2
row9
Null
row7
row3
row6
Disk



Nested Loop

(1) small data sources

(2) good filter conditions

- use indexes

(3) OLTP

(4) SGA

Driving Table

10

100

100B

Driven Table

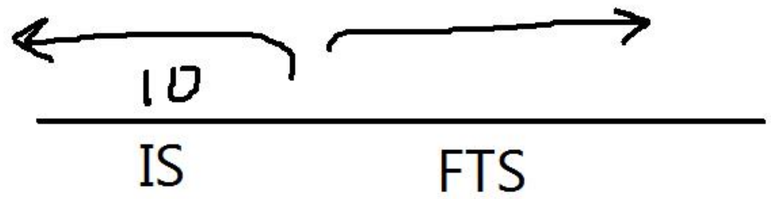
100

1000

1B

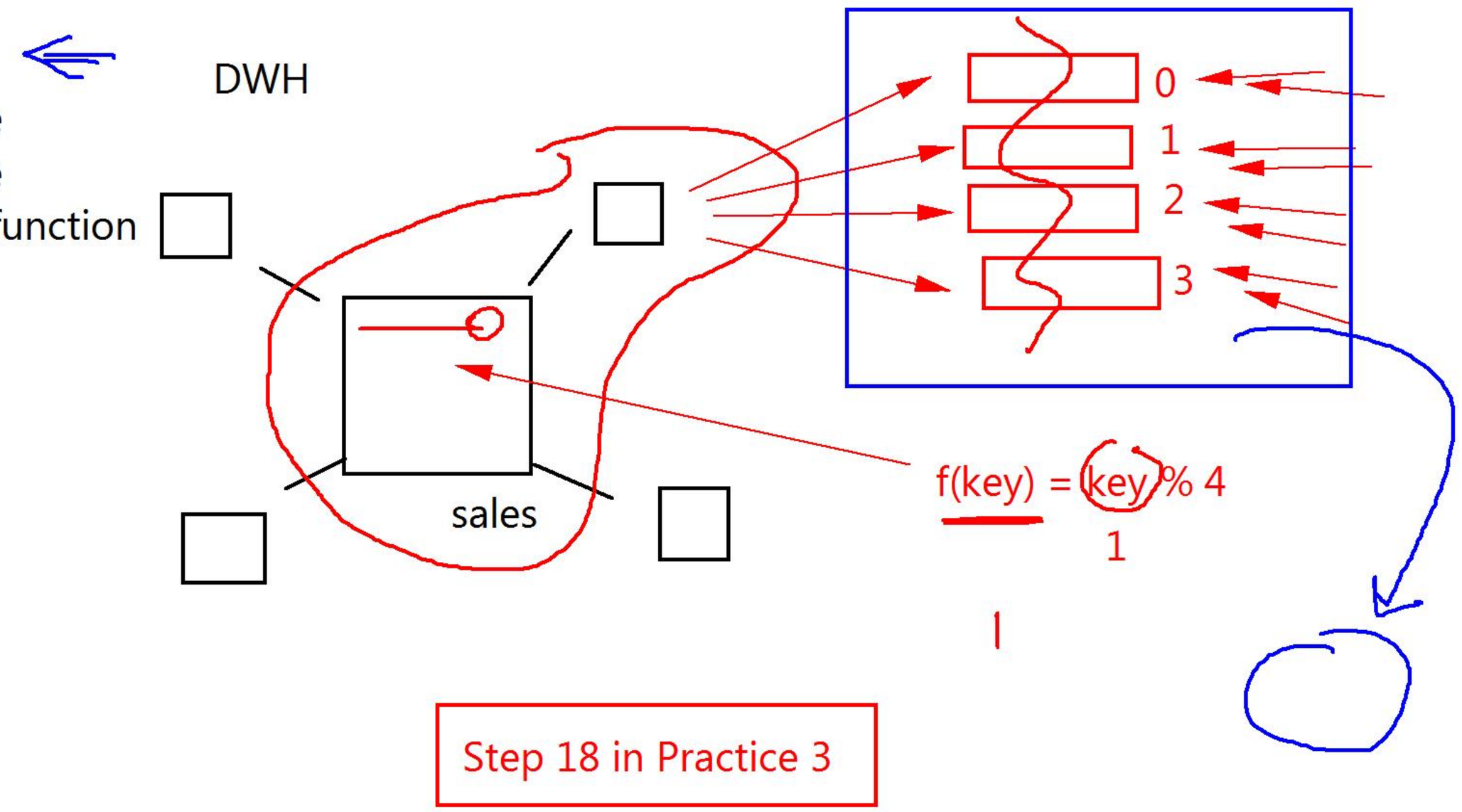
Index

selectivity < 0.1



Hash Join

- small-large
- large-large
- uses Hash function
- uses PGA



Step 18 in Practice 3

Merge Join

- Large Data Sources

- case where HJ is NOT possible (<> between)
  - pre-sorting