CCDUG

Sept 18/19th 2023
Toronto, Canada

IDUG

Explain explained…

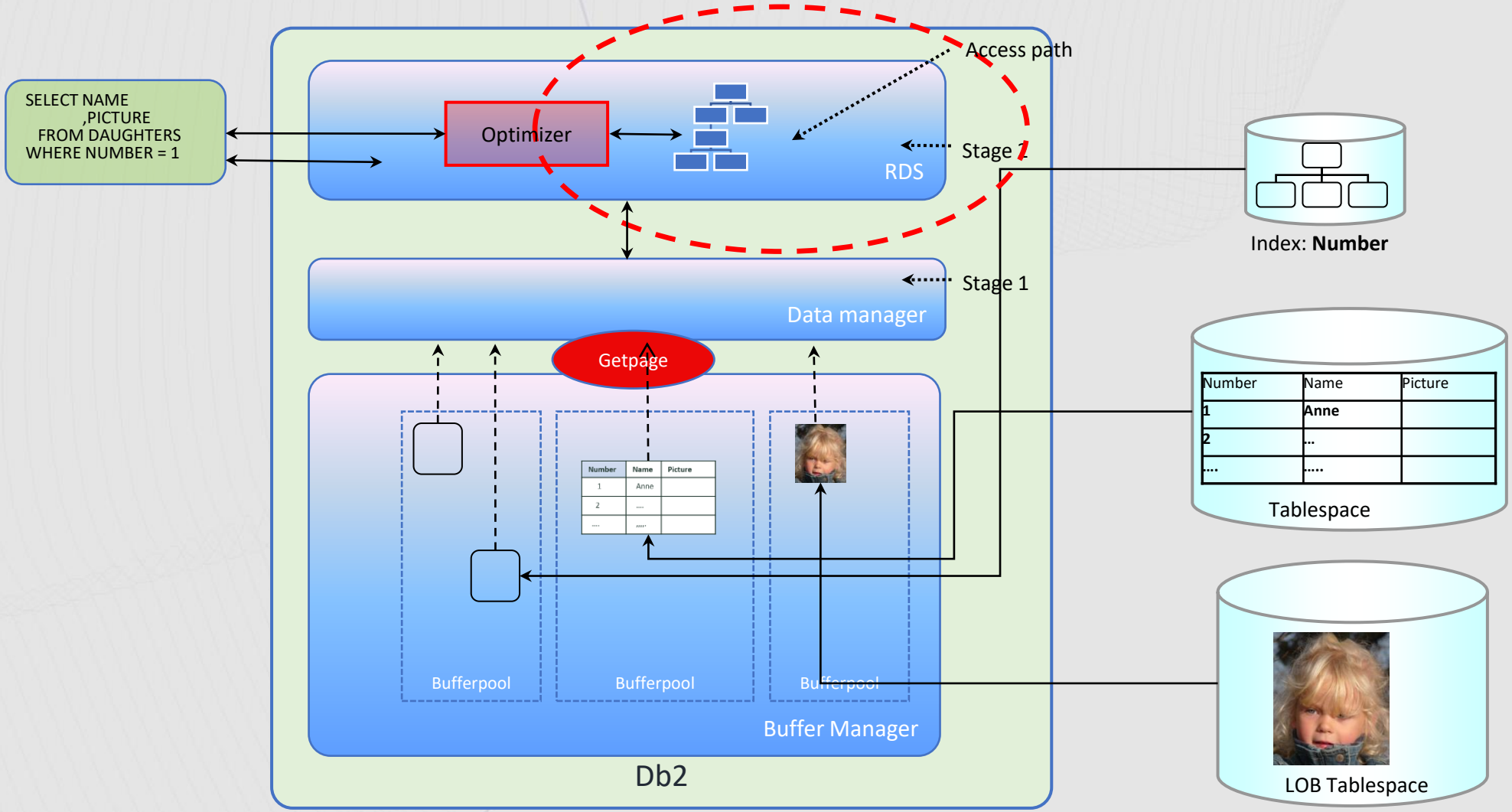
Toine Michielse, Broadcom

toine.michielse@broadcom.com

Agenda

- Executing a query
- Access Path base elements
- Final Remarks

Executing a query



Executing a query

- A key characteristic of relational DBMSs is that the user does not have to develop a strategy for accessing data
  - User only defines the desired result set using SQL syntax
  - The DBMS (optimizer) determines optimal access strategy to create this result set. This is called the access plan and consists of one or more access paths.
  - Internal representation for dynamic SQL created at runtime (prepare) if not already present in local or global statement cache
- Note the term plan. Db2 will seriously attempt to follow the plan
  - Environmental reasons may cause the plan to fail.
    - Processing involving RID lists may run into thresholds and fallback to tablespace scan
    - Parallel plans may run into less resources than expected and reduce the number of parallel groups or even fallback to sequential plan

Executing a query : Optimizer

- To determine the access plan, the optimizer breaks down the query in query blocks.
- An informal definition of a query block could be ‘the part of a query that deals with a single from clause’.
- Optimization involves:
  - Query rewrite
  - Predicate generation (transitive closure)
  - Statistics gathering
  - Filter factor calculation
  - Possible access path determination
  - Optimal access path selection
  - Optionally attempt to build parallel access path

Executing a query : Optimizer decision

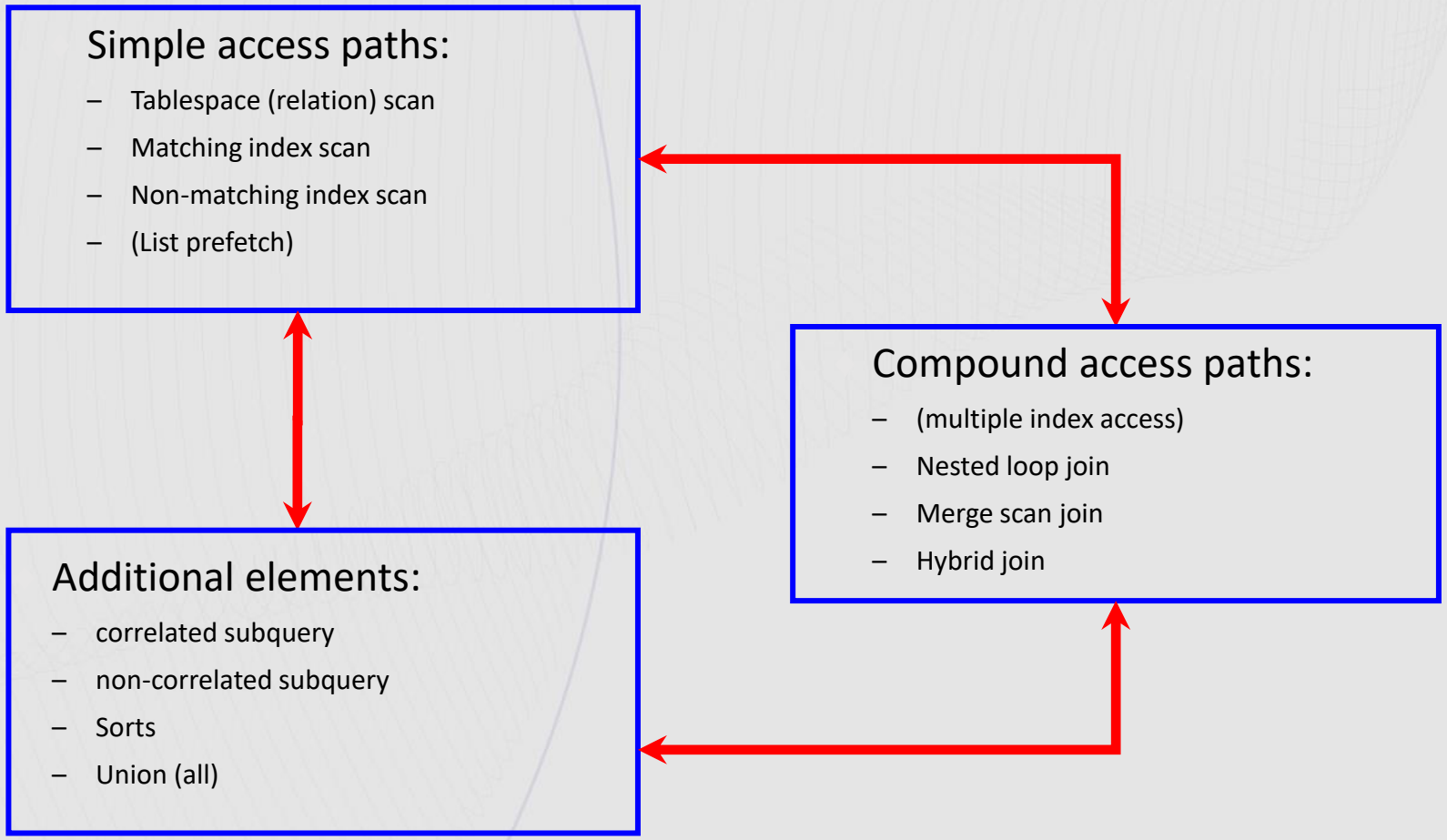
- The access plan chosen by the optimizer can be made available to the user through the EXPLAIN facility.
  - Explain provides possibility for early verification of the expected performance as well as table and index design
  - It is a good attitude to explain statements and verify this output
  - Strongly recommended to EXPLAIN statements before writing the program
- Explain invoked through:
  - EXPLAIN command for a single SQL statement
  - Using EXPLAIN(YES) parameter on bind
- Beware of the difference between dynamic and static SQL
  - Hostvariable vs. literals → use ? as placeholder for host variables (e.g. where location = ? <-> where location = ‘Madrid')

Executing a query : Optimizer output

- Explain output stored in a set of Db2 tables:
  - < creator>.**PLAN_TABLE** (must exist for Explain)
    - Contains the actual access path information. One or more rows per query.
- Explain tools:
  - all "explain tools" (Broadcom Easy Explain, Toad explain and others) use data of these tables, show an access plan graph or only a text description
  - for a better understanding WHY the optimizer chose the access plan.
  - But a the end of the day you have to understand the output of the plan_table otherwise you cannot understand the output of any tool
  - more or less general recommendations may be given (think at sorts, statistics,…)
  - the tool would never take a "bad" statement and transform it somehow in a good statement

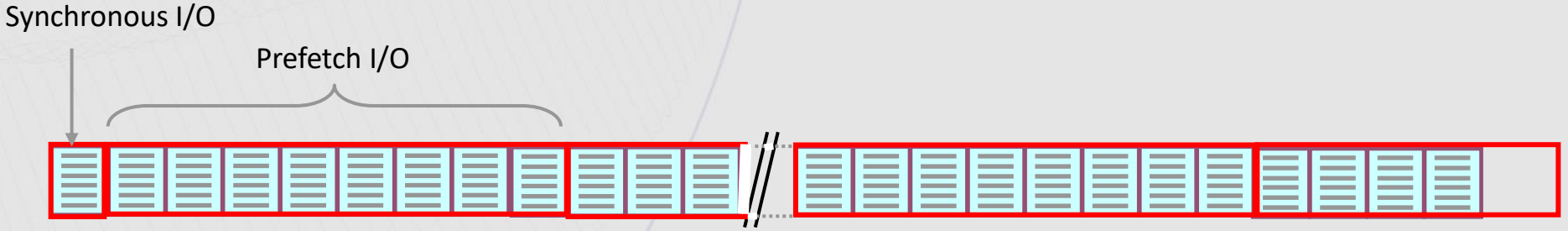
Access Path : base elements

Access Path Overview



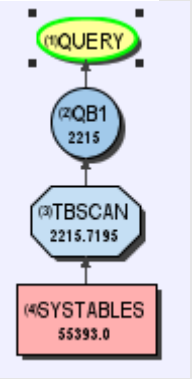
Access Path: Table(space) scan

- At execution time, Db2 will read all pages of the physical dataset
- To reduce elapsed time, sequential prefetch will always be used
  - The only access strategy if no **usable** index exists
  - A good access strategy if a large percentage of the rows (> 20%) needs to be accessed or if the table is very small.
  - Predicates will be applied to each and every row
  - Unit of I/O is always multiple, physically sequential pages (up to 32)
    - Average cost per page a fraction of synchronous I/O



Access Path: Table(space) scan

- How is this recorded in PLAN_TABLE
  - **Accesstype = 'R', Prefetch = 'S'** (sequential)
  - Accessname = blank



Query:

```
EXPLAIN ALL SET QUERYNO = 1 FOR
SELECT * FROM SYSIBM.SYSTABLES
WHERE CHAR(DBNAME) = 'D'!!USER
WITH UR
```

Plan_table contents (important attributes only):

QNO	CREATOR	TNAME	ACCESSTYPE	ACCESS CREATOR	ACCESSNAME	PREFETCH
1	SYSIBM	SYSTABLES	R			S

Why did this query use tablespace scan, there is an index on DBNAME?

Column function CHAR used. This is a STAGE2 predicate and therefore not indexable

How to quickly check the your predicates

- DSN_FILTER_TABLE
  - Contains a row for every predicate ***that is used during execution***
  - ORDERNO indicates order of evaluation
  - Filter factor
  - STAGE
- DSN_PREDICAT_TABLE
  - Contains a row for every predicate in the SQL statement, including generated ones
  - SEARCHARG

Learning more about your predicates

```
 SELECT
    A.QUERYNO                                  AS Q#
   ,A.QBLOCKNO                                 AS QB#
   ,B.ORDERNO                                  AS EVAL#
   ,CAST(A.FILTER_FACTOR AS DEC(5,4))          AS FF
   ,A.BOOLEAN_TERM                             AS BT
   ,A.SEARCHARG
   ,B.STAGE
   ,CAST(A.PREDNO AS SMALLINT)                 AS P#
   ,A.TYPE
   ,SUBSTR(A.LEFT_HAND_SIDE,1,12)              AS LHS
   ,SUBSTR(A.RIGHT_HAND_SIDE,1,12)             AS RHS
 FROM DSN_PREDICAT_TABLE A LEFT JOIN
      DSN_FILTER_TABLE   B
  ON  A.QUERYNO          = B.QUERYNO
  AND A.QBLOCKNO         = B.QBLOCKNO
  AND A.PREDNO           = B.PREDNO
WHERE A.QUERYNO          = 1
  AND TYPE <> 'COMPOUND'
ORDER BY Q#, QB#, EVAL#
```

ORDERNO:

- Indicates in which order predicates are evaluated
- Within a STAGE group, in order written in SQL
- For optimal cost/performance, order in ascending FILTER-FACTOR order within your SQL text

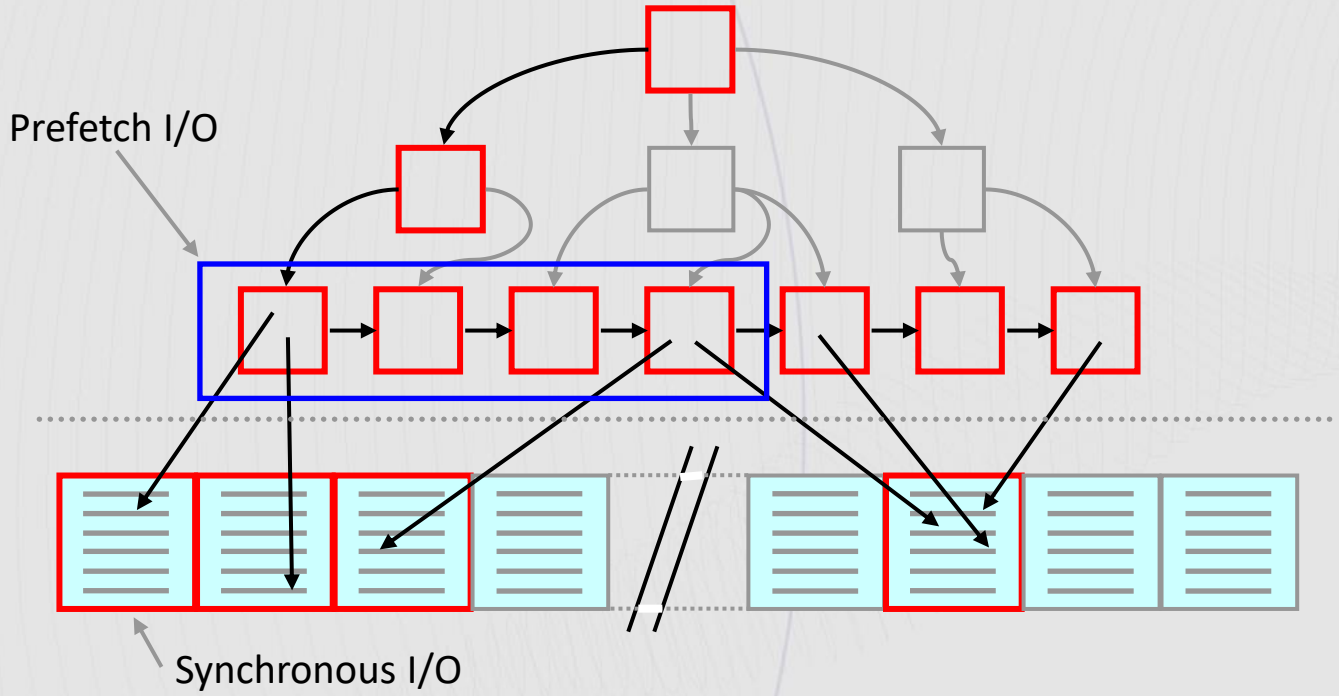
FILTER_FACTOR:

- Number between 0 and 1
- #rows_out = #rows_in * filter_factor
- The smaller the better

STAGE:

- Index access
  - MATCHING
  - SCREENING
- After data page access
  - STAGE1
  - STAGE2

Access Path: Non-Matching Index Scan



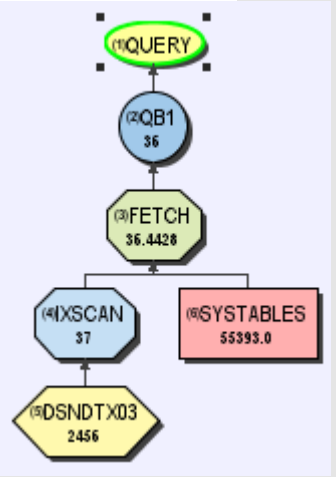
- Used when index is expected to provide good filtering but Db2 can not use matching predicates to limit search to certain leaf pages
  - Leaf pages typically read using sequential prefetch
  - Entire set of leaf pages must be read
  - If not index-only, fraction of the data pages read using synchronous I/O

Access Path: Non-Matching Index Scan

- How is this recorded in PLAN_TABLE
  - first row **Accesstype = 'I, Matchcols=0**
  - Acessname contains index used

Query:

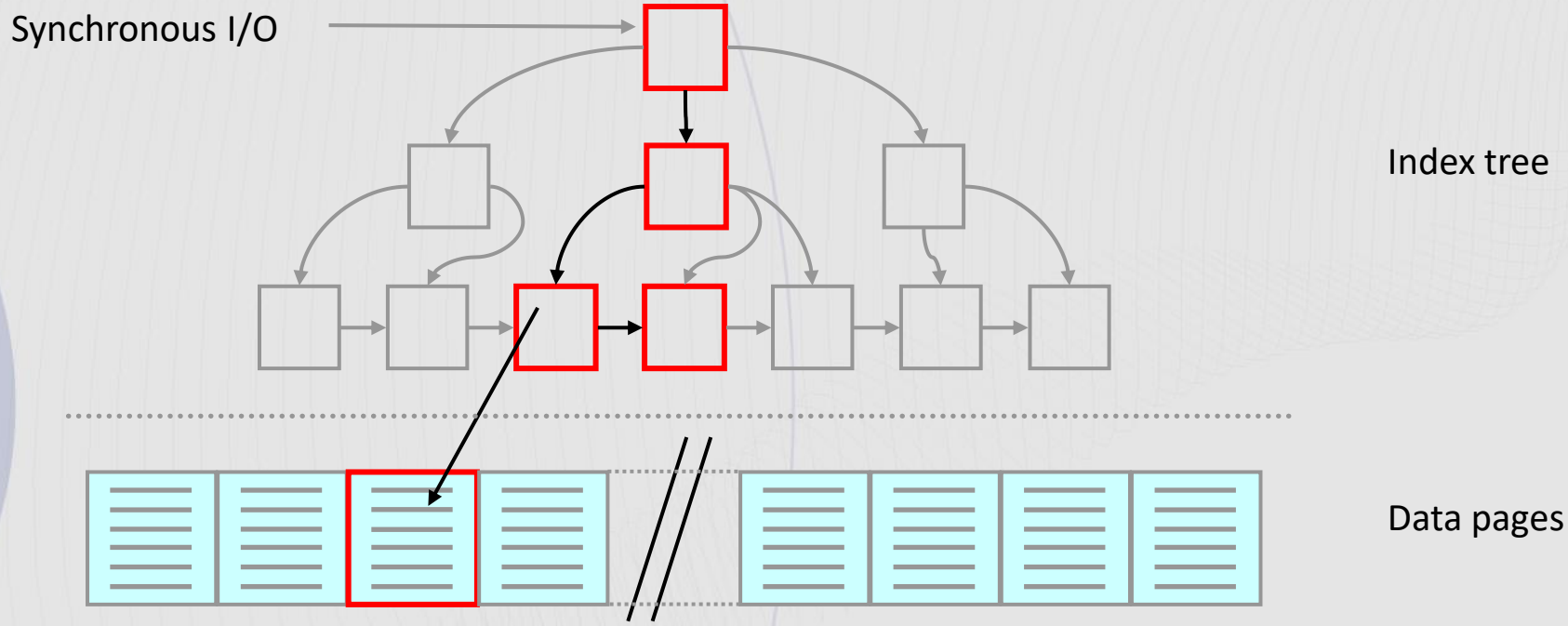
```
EXPLAIN ALL SET QUERYNO = 4 FOR
SELECT *  FROM SYSIBM.SYSTABLES
WHERE TBNAME = 'ABC'
WITH UR;
```



Plan_table contents (important attributes only):

QNO	CREATOR	TNAME	ACCESS TYPE	MATCH COLS	ACCESS CREATOR	ACCESSNAME	PREFETCH
4	SYSIBM	SYSTABLES	I	0	SYSIBM	DSNDTX03	

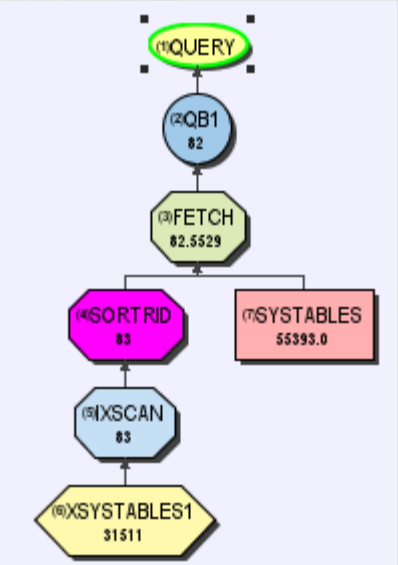
Access Path: Matching Index Scan



- Db2 uses matching predicates to find first leaf page that may contain qualifying entries (synchronous I/O)
  - Additional ‘screening’ predicates will be applied to further limit number of entries that are passed to subsequent processing stages
  - Only a fraction of leaf pages (and non-leaf pages) expected to be processed
  - Only a fraction of data pages (if any) expected to be processed

Access Path: matching Index scan

- How is this recorded in PLAN_TABLE
  - **Accesstype = 'I', Matchcols > 0**
  - Tablename not blank, Accessname contains name of the index u



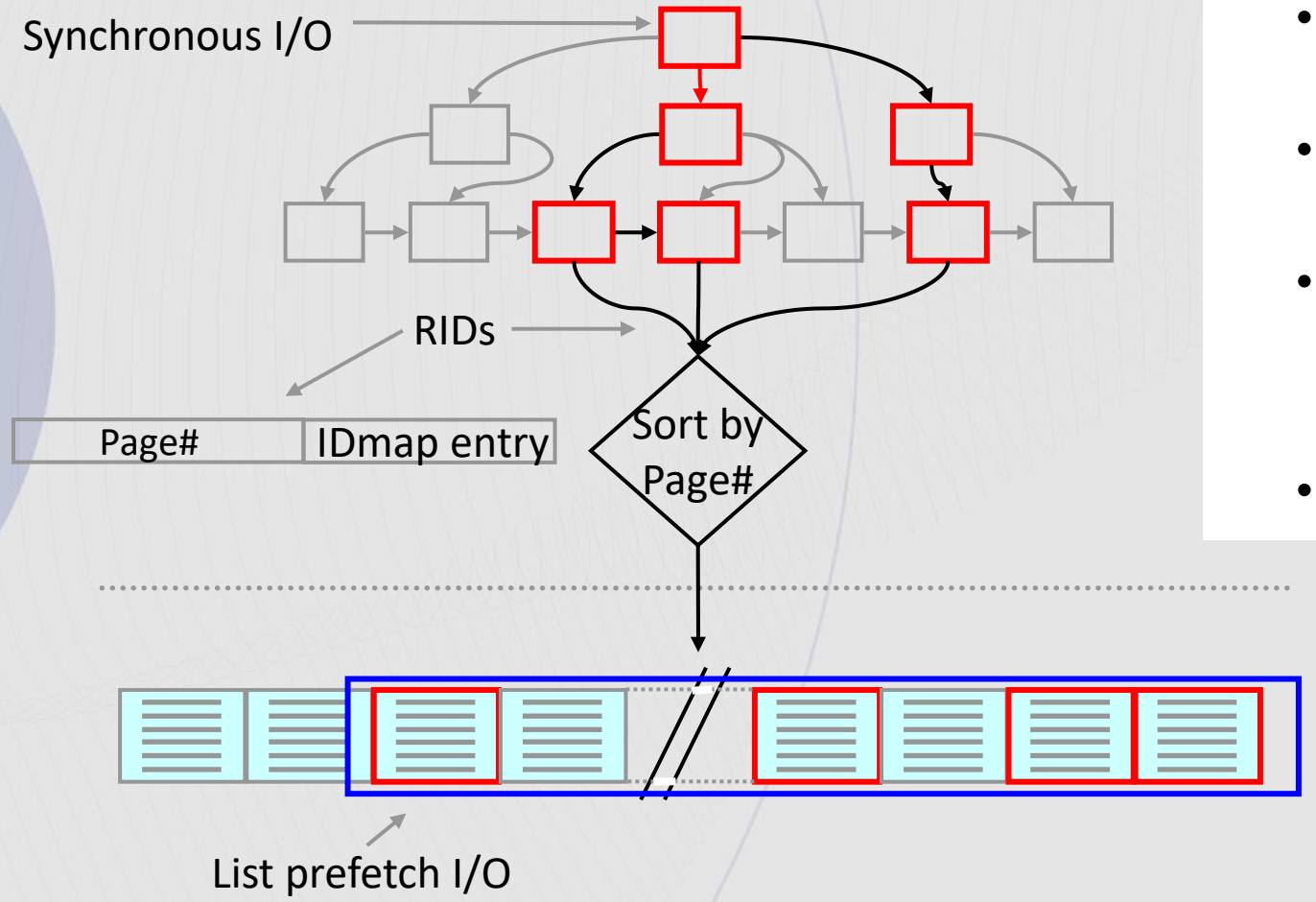
Query:

```
EXPLAIN ALL SET QUERYNO = 2 FOR
SELECT * FROM SYSIBM.SYSTABLES
WHERE DBNAME = 'D'!!USER
  AND TYPE   = 'T'
  AND TBNAME LIKE '%ABC%'
WITH UR;
```

Plan_table contents (important attributes only):

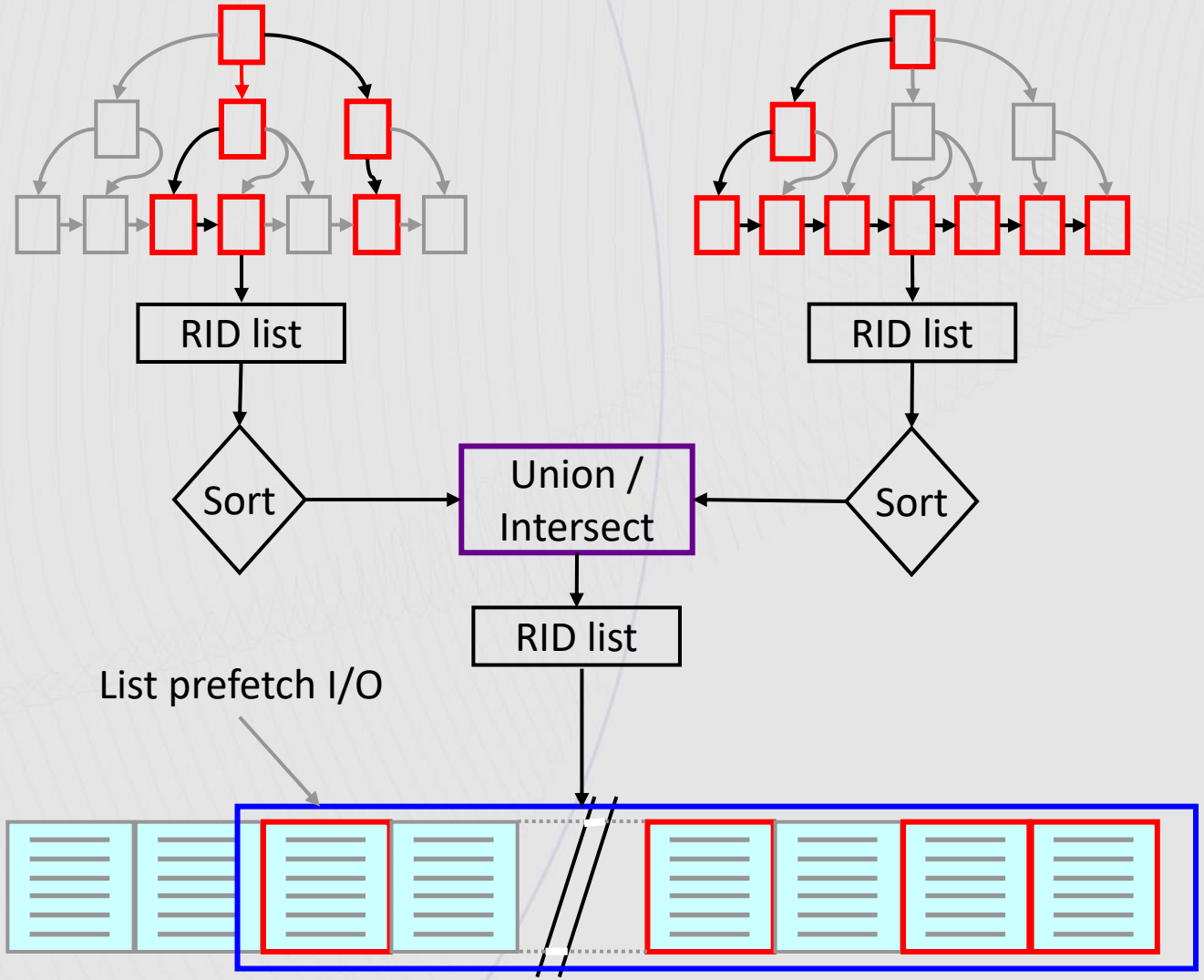
QNO	CREATOR	TNAME	ACCESSTYPE	MATCH COLS	ACCESS CREATOR	ACCESSNAME	PREFETCH
2	SYSIBM	SYSTABLES	I	1	DB2SYS	XSYSTABLES1	L

Access Path Optimization: List Prefetch



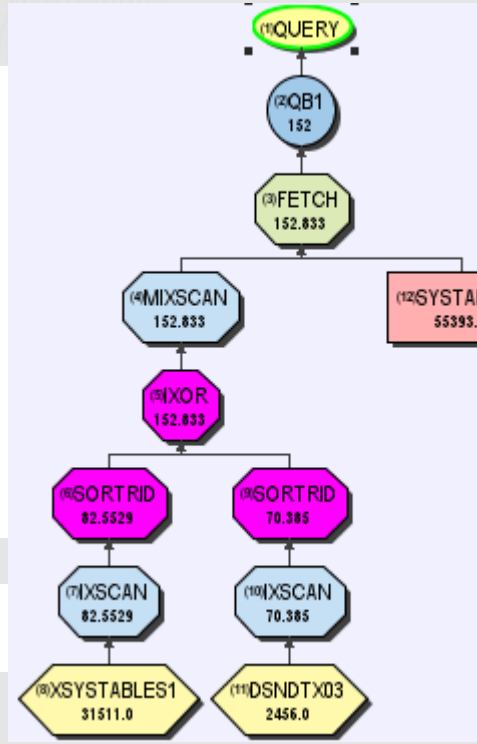
- **List prefetch is used when:**
  - A significant number of data pages needs to be accessed
  - The index used is not clustered (clusterratio < 80%)
  - Multiple non-consecutive pages will be combined in a single I/O request
    - Random I/O is avoided
  - PREFETCH = 'L'

Compound Access Path: Multiple IX Access



Compound Access Path: Multiple IX Access

- How is this recorded in PLAN_TABLE
  - first row **Accesstype = 'M', Prefetch='L'**
  - subsequent rows show index access (in MIXOPSEQ) order
    - **Accesstype = 'MX'**
  - or operations on the rid list:
    - **Accesstype = 'MU'** for union, **'MI'** for intersection



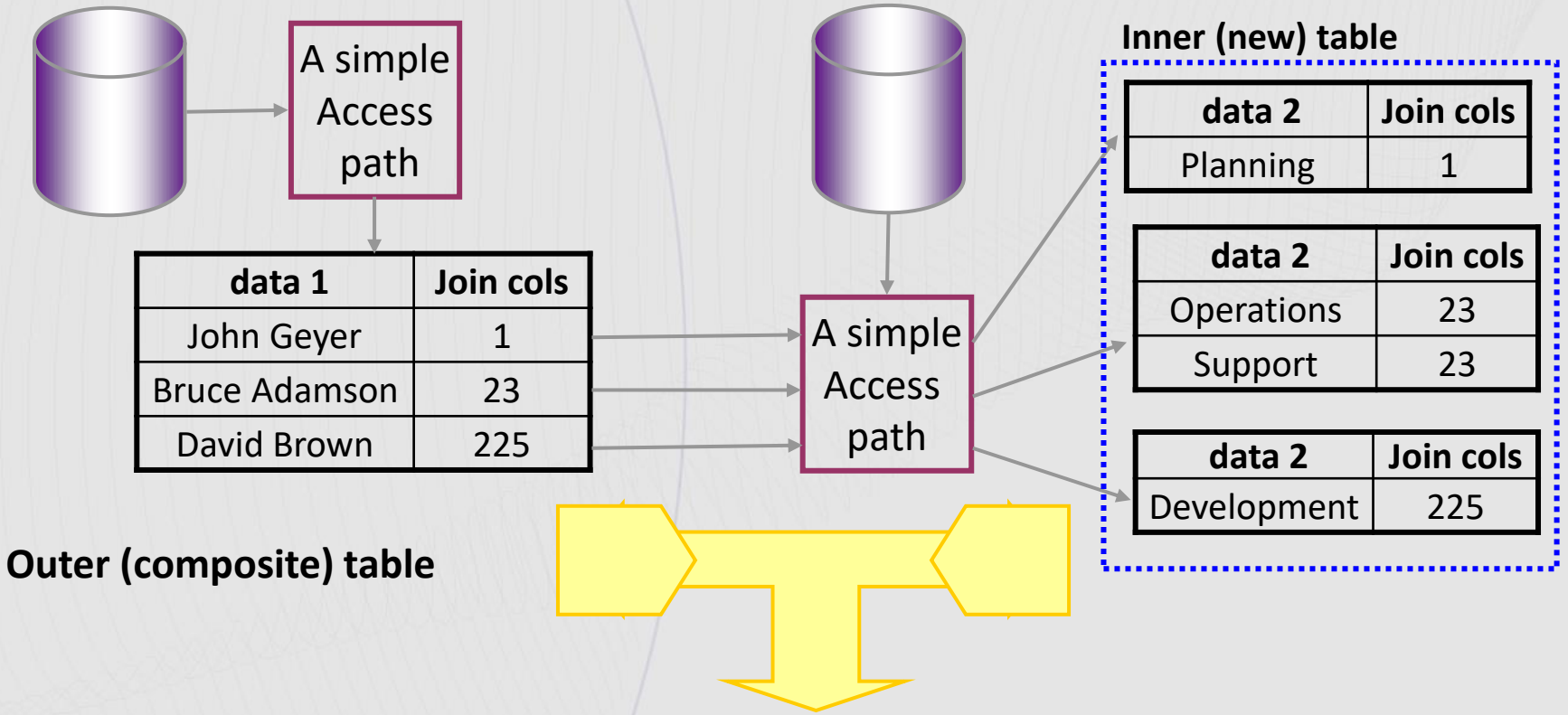
Query:

```
EXPLAIN ALL SET QUERYNO = 3 FOR
SELECT * FROM SYSIBM.SYSTABLES
WHERE DBNAME     = 'D'!!USER
   OR  TBCREATOR = USER
WITH UR;
```

Plan_table contents (important attributes only):

QNO	CREATOR	TNAME	ACCESS TYPE	ACCESS CREATOR	ACCESSNAME	INDEXONLY	PREFETCH
3	SYSIBM	SYSTABLES	M			N	L
3	SYSIBM	SYSTABLES	MX	SYSIBM	DSNDTX03	Y	
3	SYSIBM	SYSTABLES	MX	DB2SYS	XSYSTABLES1	Y	
3	SYSIBM	SYSTABLES	MU			N	

Compound access paths: Nested loop join



data 1	Join cols
John Geyer	1
Bruce Adamson	23
David Brown	225

data 2	Join cols
Planning	1

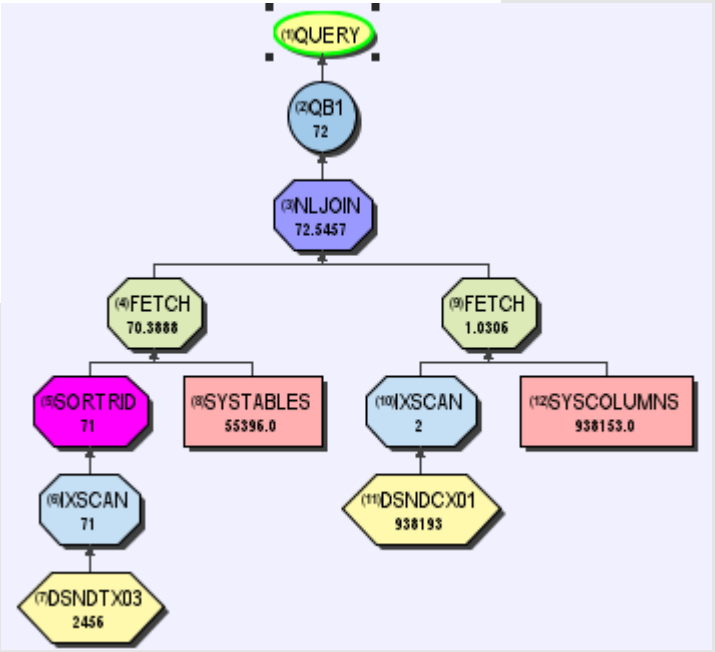
data 2	Join cols
Operations	23
Support	23

data 2	Join cols
Development	225

data 1	data 2	Join cols
John Geyer	Planning	1
Bruce Adamson	Operations	23
Bruce Adamson	Support	23
David Brown	Development	225

Compound access paths: Nested loop join

- How is this recorded in plan_table:
  - The individual simple access paths for outer and inner as usual
  - First outer table accessed: **Method = 0**
  - Inner table : **Method = 1**



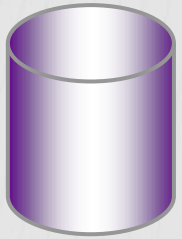
Query:

```
EXPLAIN ALL SET QUERYNO = 9 FOR
SELECT *
FROM SYSIBM.SYSTABLES   A
   ,   SYSIBM.SYSCOLUMNS B
WHERE  A.TBCREATOR = B.TBCREATOR    AND
       A.TBNAME    = B.TBNAME       AND
       A.TBCREATOR = USER
WITH UR;
```

Plan_table contents (important attributes only):

QNO	METHOD	CREATOR	TNAME	ACCESS TYPE	MATCH COLS	ACCESS CREATOR	ACCESSNAME	PREFETCH
9	0	SYSIBM	SYSTABLES	I	1	SYSIBM	DSNDTX03	L
9	1	SYSIBM	SYSCOLUMN	I	2	SYSIBM	DSNDCX01	

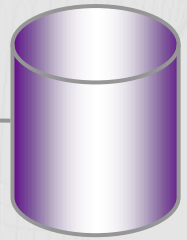
Compound access paths: Merge Scan Join



A simple Access path

Outer and inner table sorted in join column order

A simple Access path



data 1	Join cols
John Geyer	1
Bruce Adamson	23
David Brown	225

Outer (composite) table

data 2	Join cols
operation	1
write specs	2
testing	23
write code	23
control	225

Inner (new) table

Composite/result table

data 1	data 2	Join cols
John Geyer	operation	1
Bruce Adamson	testing	23
Bruce Adamson	write code	23
David Brown	control	225

Compound access paths: Merge Scan Join

- How is this recorded in plan_table:
  - The individual simple access paths for outer and inner as usual
  - First outer table accessed: **Method = 0**
  - Inner table : **Method = 2**

Query:

```
SELECT DISTINCT A.NAME, B.NAME
  FROM SYSIBM.SYSCOLUMNS A
      ,SYSIBM.SYSCOLUMNS B
WHERE (A.TBCREATOR = B.TBCREATOR  OR 0=1)
  AND A.TBNAME     = B.TBNAME
  ORDER BY A.NAME
```

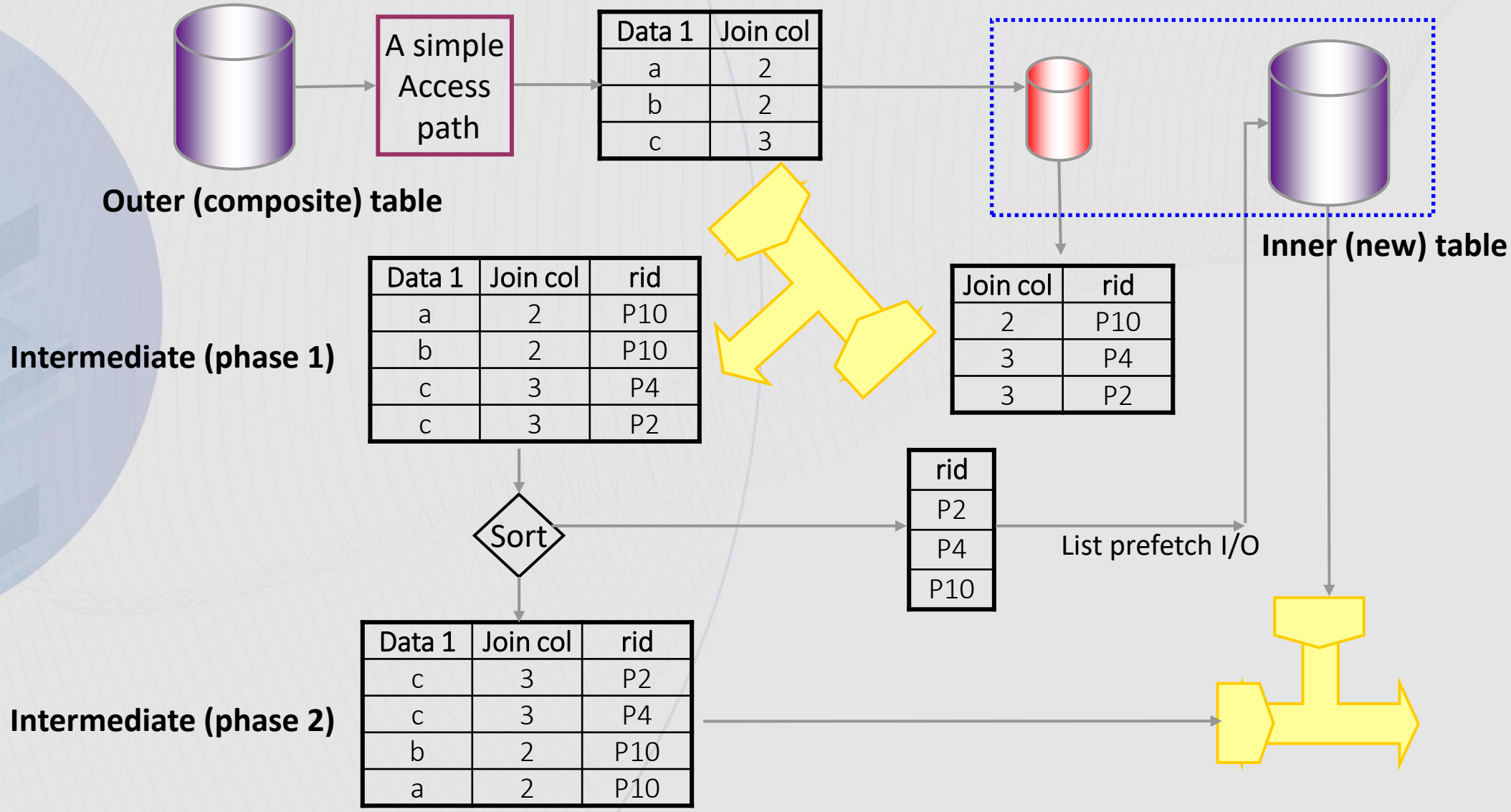
And what about sorts ?

- Composite (outer)
- New (inner)
- **Sort final result set of this query block**
- Method = **3**

Plan_table contents (important attributes only):

QNO	METHOD	CREATOR	TNAME	ACCESS TYPE	MATCH COLS	ACCESSNAME	Sort New UJOG	Sort Comp UJOG
8	0	SYSIBM	SYSCOLUMNS	I	0	DSNDCX01	....	....
8	2	SYSIBM	SYSCOLUMNS	I	0	DSNDCX01	.Y..	.Y..
8	3				0		....	Y.Y.

Compound access paths: Hybrid Join



Data 1	Join col
a	2
b	2
c	3

Join col	rid
2	P10
3	P4
3	P2

Data 1	Join col	rid
a	2	P10
b	2	P10
c	3	P4
c	3	P2

rid
P2
P4
P10

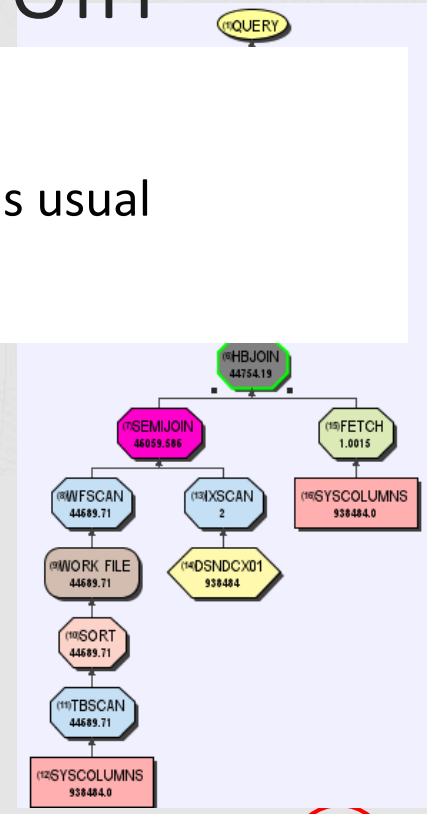
Data 1	Join col	rid
c	3	P2
c	3	P4
b	2	P10
a	2	P10

Compound access paths: Hybrid Join

- How is this recorded in plan_table:
  - The individual simple access paths for outer and inner as usual
  - Inner table : **Method = 4**

Query:

```
SELECT A.NAME, B.LENGTH , B.NAME, B.LENGTH
FROM SYSIBM.SYSCOLUMNS A
   , SYSIBM.SYSCOLUMNS B
WHERE A.TBCREATOR = B.TBCREATOR
  AND A.TBNAME    = B.TBNAME
  AND A.COLTYPE   = 'CHAR'
  AND B.COLTYPE   = 'SMALLINT'
  AND A.NAME LIKE '%ID%'
ORDER BY 3,1
```



Plan_table contents (important attributs only):

QNO	METHOD	CREATOR	TNAME	ACCESS TYPE	MATCH COLS	ACCESSNAME	Sort New UJOG	Sort Comp UJOG	PREF
7	0	SYSIBM	SYSCOLUMN	R	0		NNNN	NNNN	S
7	4	SYSIBM	SYSCOLUMN	I	2	DSNDCX01	NNNN	NYNN	L
7	3				0		NNNN	NNYN	

Final remarks

Best practices:

- The best ……. is the one never executed
  - Be wary of “STAGE3” predicates
  - Minimize your “potential sync I/O”, aka GETPAGE
- Prevent unnecessary SORTS
  - Remove ORDER BY when application is not dependent on a specific row order
  - Use DISTINCT carefully, better investigate completeness of join criteria's
  - Use UNION ALL instead of UNION when you can guarantee disjoint parts
- Appropriate Statistics
  - Optimizer decisions base on available statistics
  - When bind is done on empty tables which will contain a significant amount of rows at runtime, theaccess plan may be totally wrong
  - Evaluate when to REBIND

Best practices (cont.):

- Appropriate Statistics (cont)
  - Up-to-date statistics does not mean execute RUNSTATS indiscriminately on a frequent base. Collecting statistics is not for free.
  - Let a housekeeping process when to execute RUNSTATS, triggered by thresholds based on real time statistics.
  - New statistics after adding one million rows to a 1000 million row table will usually not affect optimizer behavior. However…….the same amount of new rows inserted in a 1000 row table…
- Adding indexes
  - In this presentation "stage1" and "indexable" are pointed out. So you may have the idea that an index to support each and every SQL statement is a good idea.
  - Changing data results in updating the index-tree as well
  - Each additional index is a trade-off between good insert/update/delete and select performance

Questions