ODD FACTS ABOUT NEW DB2 for z/OS SQL

NEW WAYS OF THINKING ABOUT OLD THINGS

+

STATIC/DYNAMIC SQL CHANGES, PREDICATE APPLICATION AND LOCKS. LATCHES, CLAIMS, & DRAINS

Bonnie K. Baker

Bonnie Baker Corporation
PO Box 18123
Tampa, FL 33679-8123 USA

1-813-477-4885

bkbaker@bonniebaker.com
http://www.bonniebaker.com

January 25, 2012

- **WHAT DOES THE OPTIMIZER CONSIDER?**

  SQL

  SORT SYNTAX & SORT AVOIDANCE

  BI-DIRECTIONAL INDEXES

  IMPACT OF PI AND NPSI VERSUS DPSI FOR SORT AVOIDANCE

  REQUIRED SORT AVOIDANCE? - DYNAMIC SCROLL CURSOR

- **WHAT DOES THE OPTIMIZER CONSIDER? continued**

  SQL

  PREDICATE TYPES (STAGE 1 OR STAGE 2)

  PREDICATE PUSHDOWN

  COLUMN = :HV1 + :HV2

  COLUMN COMPARED TO LONGER LITERAL OR HOST VARIABLE

  COLUMN COMPARED TO DIFFERENT DATA TYPE
  IMPLICIT CASTING VS. EXPLICIT CASTING

- **WHAT DOES THE OPTIMIZER CONSIDER?**

  CATALOG STATISTICAL INFORMATION

  CLUSTERRATIO

  WITH PARTITIONING/CLUSTER INFO IN INDEX DDL

  WITH PARTITIONING INFO IN TABLE DDL &
  CLUSTER INFO IN INDEX DDL

- **INDEX-ONLY ACCESS FOR VARCHAR DATA**

  SELECT COLB FROM BIGTABLE WHERE COLA = :HVA

  INDEX ON COLA, COLB

  COLA IS CHAR(30)

  COLB IS VARCHAR(40)

  COLB WAS “PADDED” IN INDEX AND IN SORT RECORD

  NOW COLB CAN BE UNPADDED IN INDEX

  BUT IS STILL PADDED IN SORT RECORD

EXAMPLE OF PROBLEM:

```
SELECT LNAME FROM T1
  WHERE EMPID = :HVEMPID
```

INDEX on EMPID, LNAME - - created just for Index-Only

Scene 1: LNAME DEFINED AS VARCHAR(40)
LNAME = BAKER
2-BYTE INDICATOR ON TABLE CONTAINS (5)

What are possible scenarios?

VARCHAR vs. CHAR

- **BUT WHAT IF POPULATED INCORRECTLY**

  AT LOAD

  BY APPLICATION PROGRAM

  LNAME LENGTH IS ALWAYS 40?!!!!!

  NO ADVANTAGE TO UNPADDED INDEXES
  NO SPACE SAVINGS IN TABLE
  NO SPACE SAVINGS IN INDEX

SQL to HELP

TO HELP ESTIMATE SAVINGS
TO DETERMINE CODING/LOADING PRACTICES

1) Select MIN(LENGTH(LNAME)), MAX(LENGTH(LNAME))
into :HVMIN :HVMININD, :HMAX :HVMAXIND -- WHY INDICATOR?
from MYTABLE

ARE THEY THE SAME?

2) Select LENGTH(LNAME), COUNT(*)
from MYTABLE
Group by LENGTH(LNAME)

ANSWER:	**2**	**10**
	3	**15**
	4	**85**
	5	**106**
	6	**110**
	7	**98**
	.	**….**
	40	**4,673,218**

HOW MANY LNAMEs ARE THERE AT EACH POSSIBLE LENGTH?

VARCHAR vs. CHAR

- **SQL UPDATE TO TRIM TRAILING SPACES AND RESET LENGTH**

```
UPDATE table
   SET VARCHAR_COL = RTRIM(VARCHAR_COL)
   WHERE …please use limiting predicates for reasonable
                 UOWs with COMMITS between
```

- **V7 UNLOAD**

  CAN TRIM TRAILING BLANKS

- **V8/9/10 UNLOAD AND LOAD UTILITIES**

  CAN TRIM TRAILING BLANKS

USEFUL IF CONVERTING

CHAR TO VARCHAR
VARCHAR TO LONGER VARCHAR
INCORRECTLY POPULATED TO CORRECTLY POPULATED

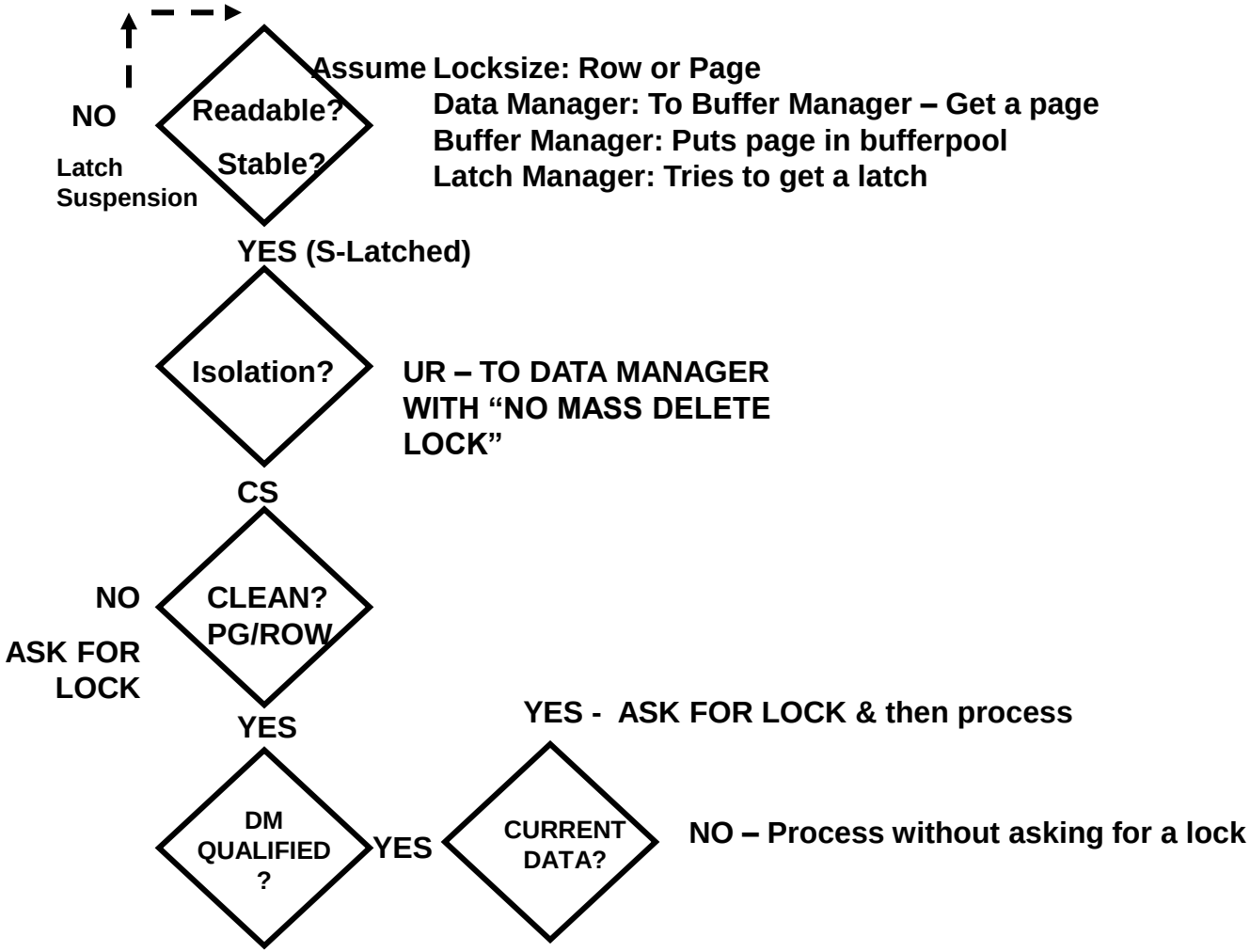
- **SINGLETON (SEARCHED) SELECT LOCKING CHANGES**

  CURRENTDATA NOW APPLIES TO SEARCHED SELECTS

  MORE LOCK AVOIDANCE

LOCK AVOIDANCE

For UNAmbiguous (V3) & Ambiguous (V4) Read-Only SQL Cursors and Singleton Selects (V8)



ODD FACTS

- **WHAT IF NO CLUSTER INDEX?**

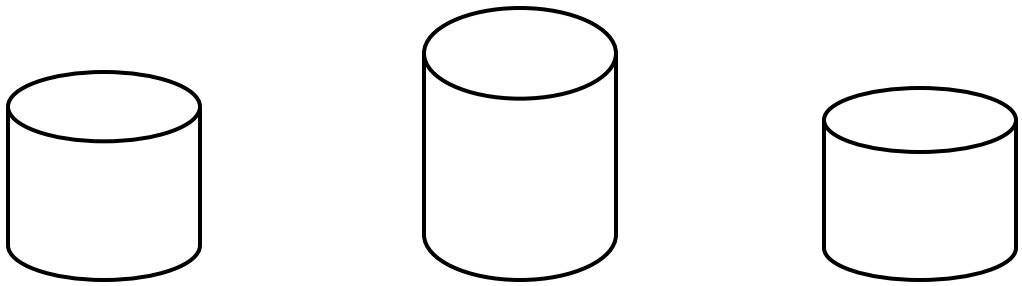
  INSERTS? V7 AND PRIOR = FIRST INDEX CREATED

  REORGS? V7 AND PRIOR = NO CLUSTERING ENFORCED

  NOW? TAIL WAGGED THE DOG!

  BUT NEW FEATURE OF REORG IS VERY USEFUL!

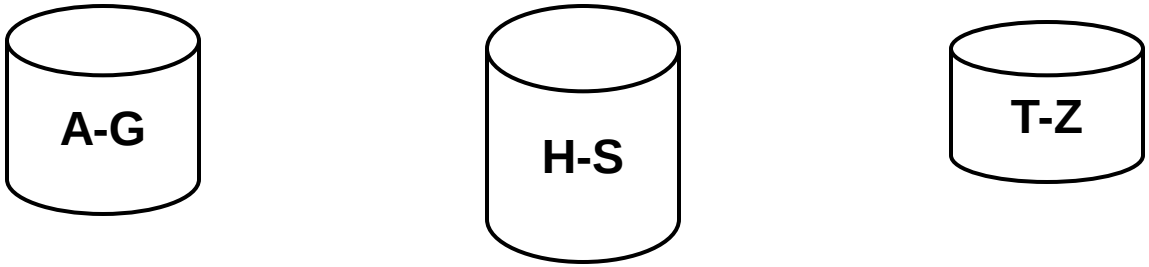
PARTITIONING METHODS
&
NEW INDEX TYPES



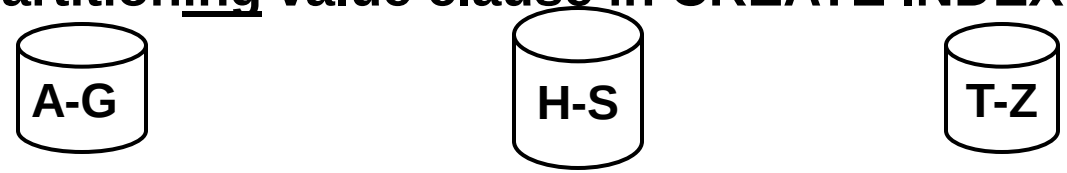
INDEX PARTITIONING

- PI = PARTITIONING + CLUSTER + ONLY 1 ALLOWED + CAN BE UNIQUE OR NOT
- NPI = NON-PARTITIONING + CAN BE MORE THAN ONE + CAN BE UNIQUE OR NOT

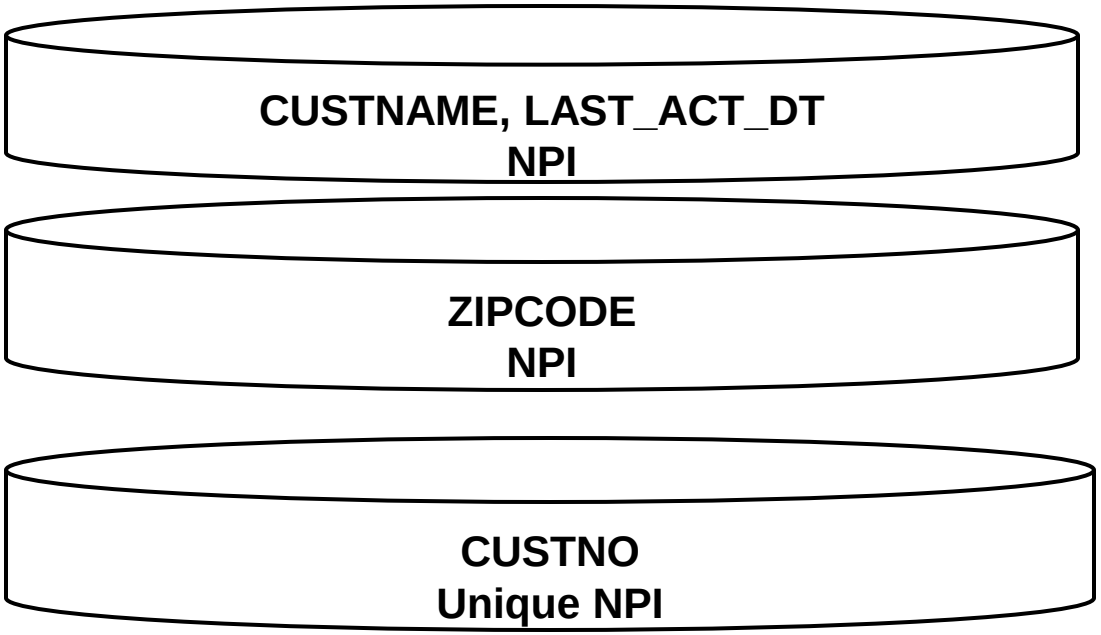
```
CREATE TABLESPACE TSCUSTMASTER
        NUMPARTS 3
        PART 1
                PRIQTY….
        PART 2
                PRIQTY…
        PART 3...

CREATE TABLE CUSTMASTER… in TSCUSTMASTER

CREATE UNIQUE INDEX PARTINDEX ON CUSTMASTER  --- WHY UNIQUE?
    (CUSTNAME, CUSTNO)
        CLUSTER
        PART 1
                PRIQTY…
                VALUE ('G999999999…', 9999999999)
        PART 2
                PRIQTY…
                VALUE ('S999999999…', 9999999999)
        PART 3...

CREATE INDEX INDEX2 ON CUSTMASTER  ------ AN NPI
    (CUSTNAME, LAST_ACT_DT)
     PRIQTY…
```



V7 partitioning value clause in CREATE INDEX CLUSTER



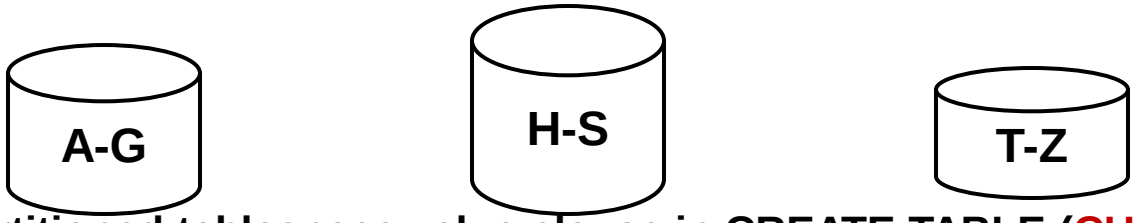
CUSTNAME,CUSTNO = partitioning & cluster



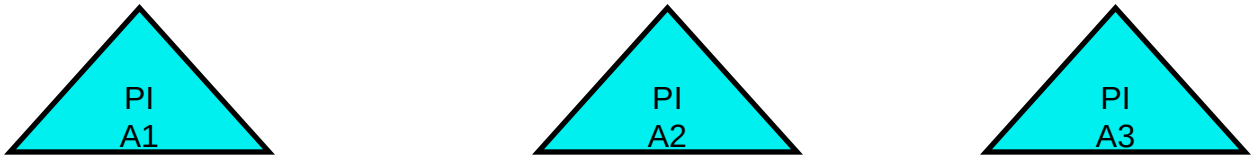
V8 INDEX TYPES FOR PARTITIONED TABLESPACES

- PI = PARTITIONING = New Way = CAN BE MORE THAN ONE PARTITIONING INDEX
- Any index aligned with (BEGINNING WITH) partitioned column data
- AND any index can be the CLUSTER index

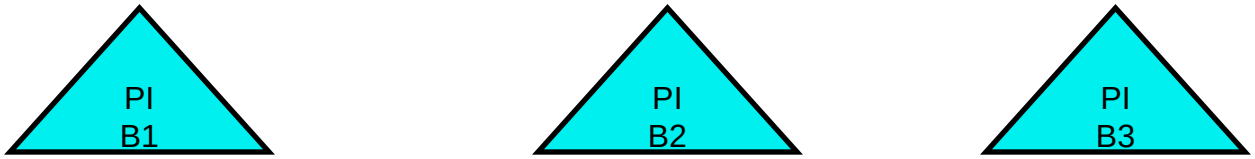
```
CREATE TABLESPACE  TSCUSTMASTER
                   NUMPARTS 3                          ---NOTE - LOCKPART YES REQUIRED IN V8
                   PART 1
                              PRIQTY….
                   PART 2
                              PRIQTY…

        CREATE TABLE CUSTMASTER…   in TSCUSTMASTER
                   (CUSTNAME …
                              PARTITIONING CLAUSE HERE!

        CREATE UNIQUE INDEX INDEX1 ON CUSTMASTER       --- PARTITIONING INDEX = PI
           (CUSTNAME, CUSTNO)                          ---  A UNIQUE INDEX
                   PART 1
                              PRIQTY…
                   PART 2
                              PRIQTY…

        CREATE INDEX INDEX2 ON CUSTMASTER              --- 2ND PARTITIONING INDEX = PI
           (CUSTNAME, LAST_ACT_DT)                     --- A NON-UNIQUE INDEX
                   PART 1
                              PRIQTY…
                   PART 2
                              PRIQTY…
```



V8 **partitioned tablespace value clause in CREATE TABLE (CUSTNAME)**



Index A = CUSTNAME,CUSTNO = UNIQUE PARTITIONING INDEX



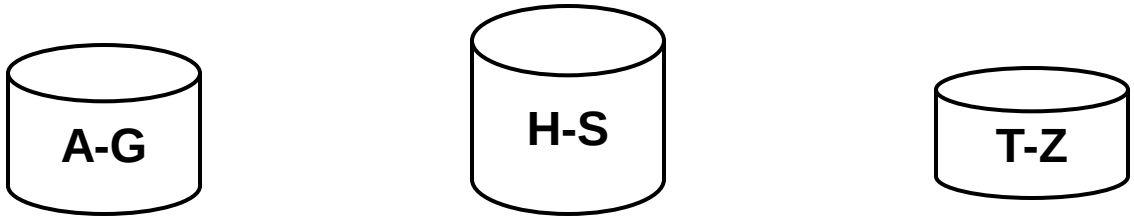
Index B = CUSTNAME,LAST_ACT_DT = NON-UNIQUE PARTITIONING INDEX

V8 INDEX TYPES FOR PARTITIONED TABLESPACES cont.

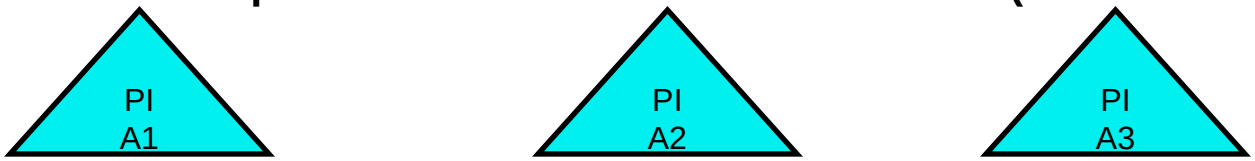
DPSI = DATA PARTITIONED SECONDARY INDEX
CAN BE MORE THAN ONE
CANNOT BE PHYSICALLY UNIQUE (V9 CHANGE)

NPSI = NON-PARTITIONED SECONDARY INDEX
SAME AS OLD NPI
CAN BE MORE THAN ONE
UNIQUE IS OK

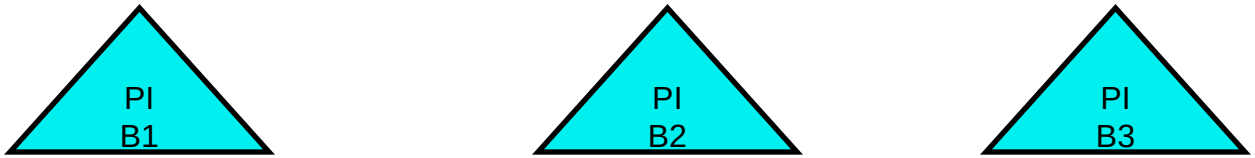
```
CREATE INDEX INDEX3 ON CUSTMASTER      --- DATA-PARTITIONED
       (ZIPCODE)                       --- SECONDARY INDEX
                PART 1                 ---  DPSI
                   PRIQTY...           ---  V9 CHANGE: CAN BE UNIQUE IF

                PART 2                 ....  ON ZIPCODE, CUSTNAME, CUSTNO
                   PRIQTY...
                PART 3...

CREATE UNIQUE INDEX INDEX4 ON CUSTMASTER    --- NON-PARTITIONED
       (CUSTNO)                             --- SECONDARY INDEX
                CLUSTER                     ---  NPSI
                PRIQTY...                   --- THE UNIQUE INDEX
```



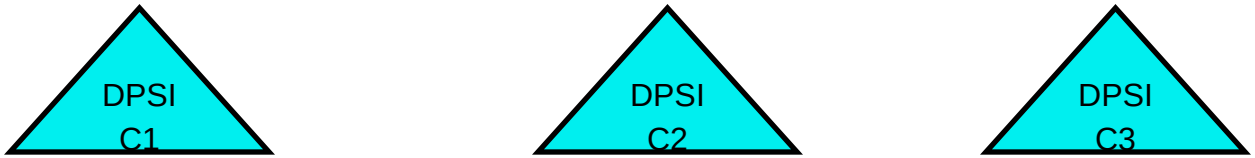
partitioned tablespace value clause in CREATE TABLE (CUSTNAME)



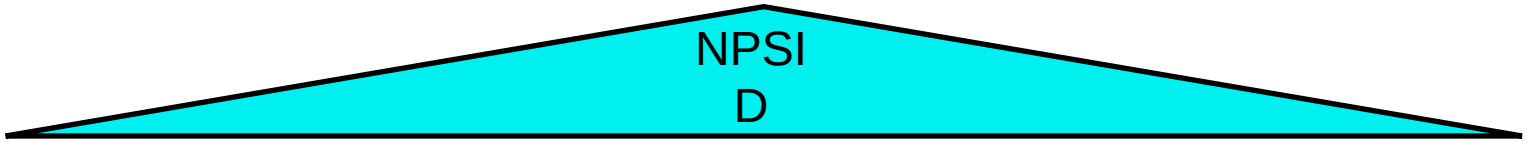
Index A = CUSTNAME,CUSTNO = PARTITIONING INDEX (UNIQUE)



Index B = CUSTNAME,LAST_ACT_DT = PARTITIONING INDEX(NON-UNIQUE)



Index C = ZIPCODE = DATA PARTITIONED SECONDARY INDEX



Index D = CUSTNO = UNIQUE, CLUSTER, NON-PARTITIONED SECONDARY INDEX

PIs, DPSIs, & NPSIs

- **PARTITIONING INDEXES**

  MAY BE UNIQUE OR NOT

  MAY BE MORE THAN ONE

  IF TS PARTITIONED ON A, B

  INDEX ON ABCDE
  INDEX ON ABQRS

TABLE-CONTROLLED PARTITIONING

```
CREATE TABLE CUSTOMER

(CUSTNO          INTEGER,
 CUSTNAME        VARCHAR(30),
 LAST_ACT_DT     DATE,
 ZIPCODE         SMALLINT)

PARTITION BY (CUSTNAME  ASC )
( PARTITION  1  ENDING AT (G9999999999…),
  PARTITION  2  ENDING AT (S9999999999… ),
  PARTITION  3  ENDING AT (Z9999999999… )
);
```

Last value is ENFORCED!

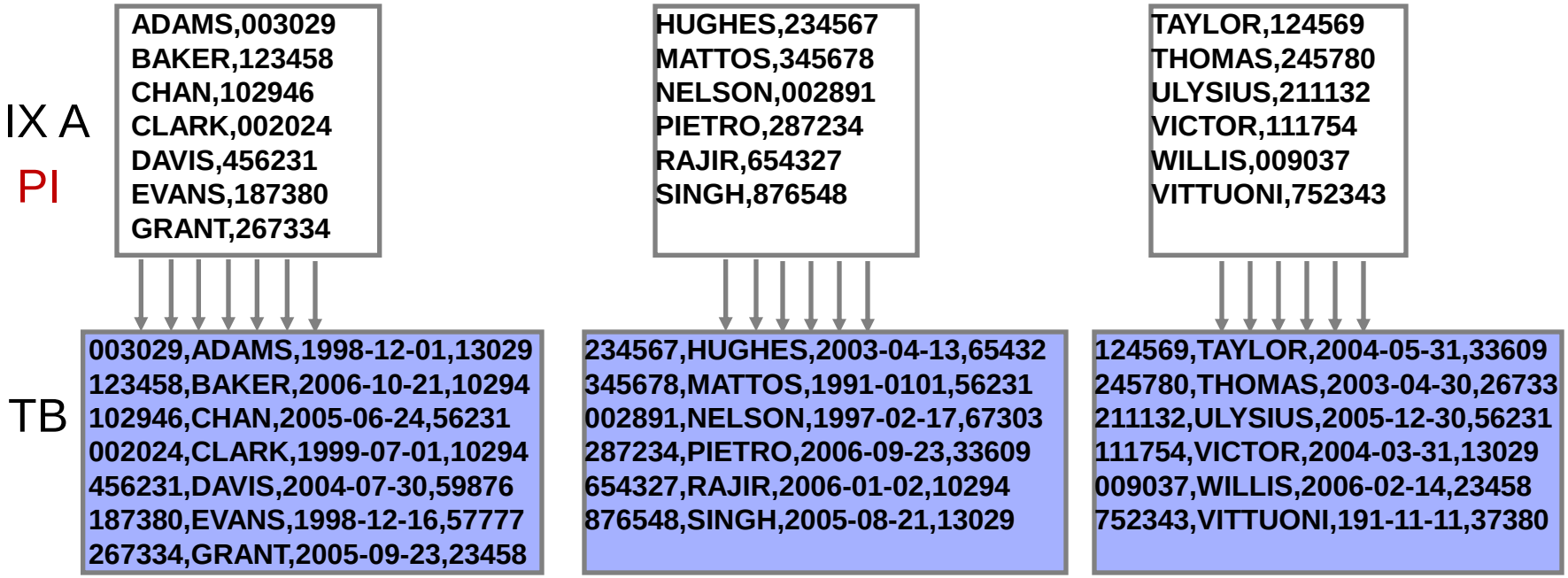
No indexes are required for partitioning!!

Partitioned Tablespace

PartitionING indexes

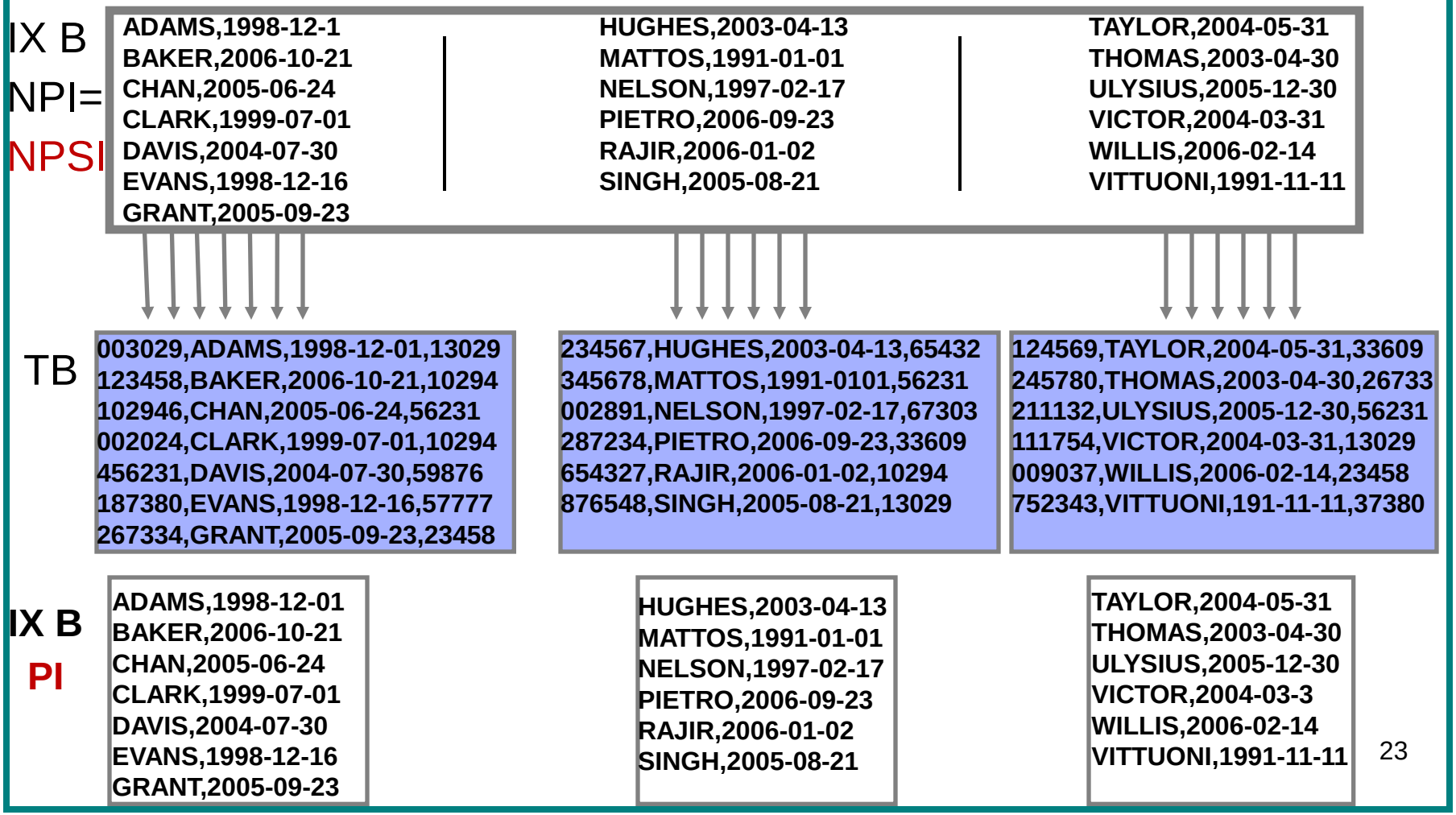
A partitionING index has the same leftmost columns, in the same collating sequence, as the columns which partition the table

PartitionING Index A (A1, A2, A3) on (CUSTNAME, CUSTNO)

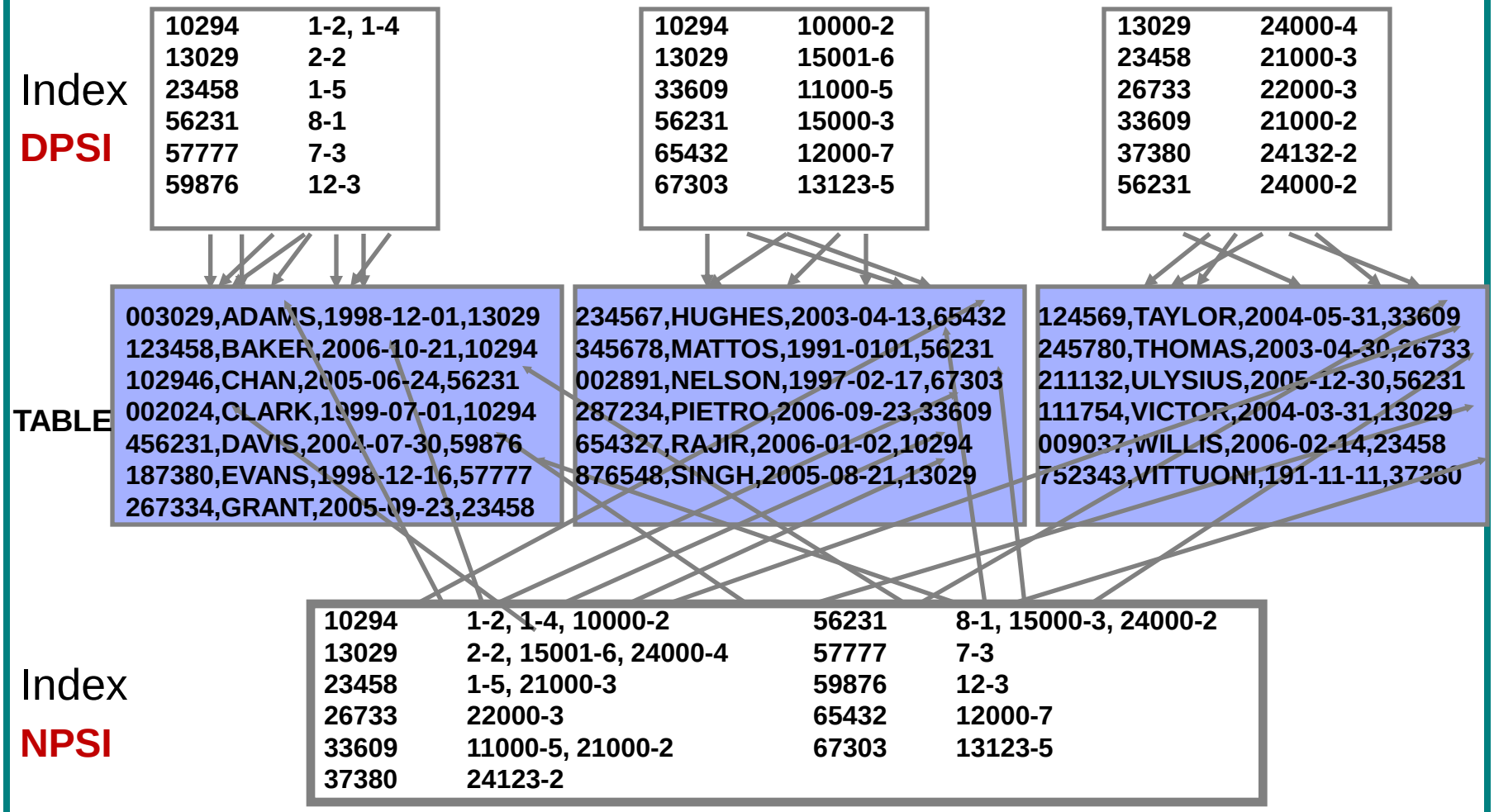


PartitionING indexes

Potential PartitionING Index B (could be B1, B2, B3) on (CUSTNAME, LAST_ACT_DT)



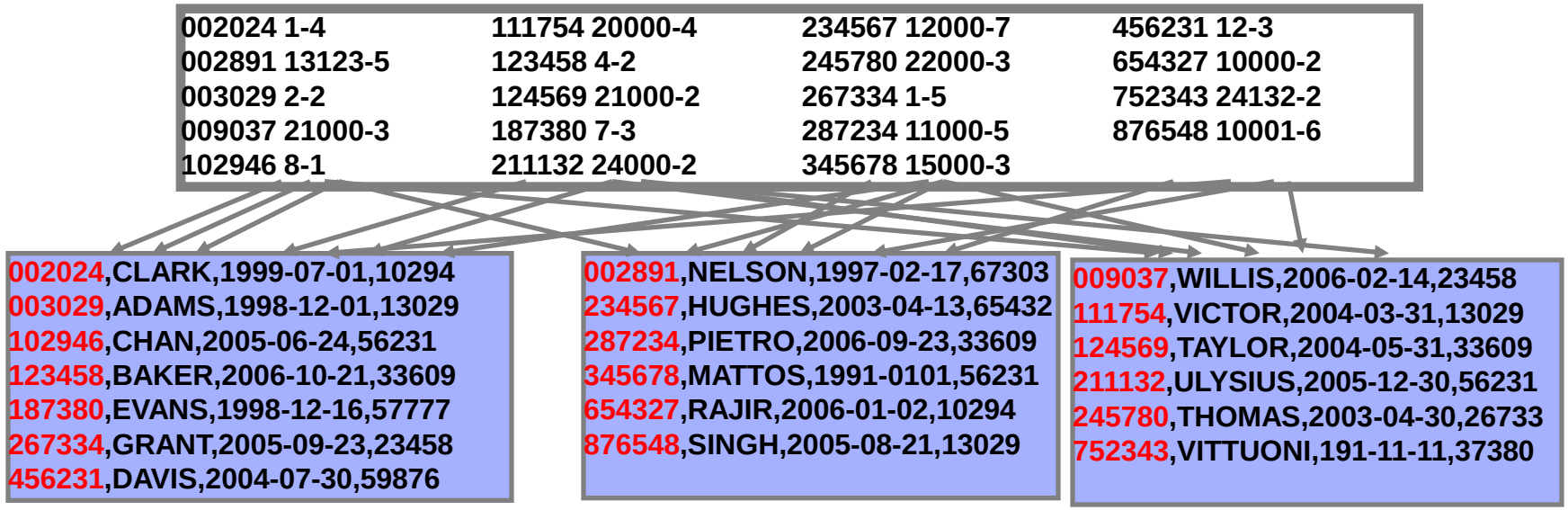
Secondary Index on ZIPCODE as a non-CLUSTER DPSI



Secondary Index on ZIPCODE AS NPI = NPSI

Non-Partitioned Secondary Index

NPSI D on (CUSTNO) CLUSTER & UNIQUE



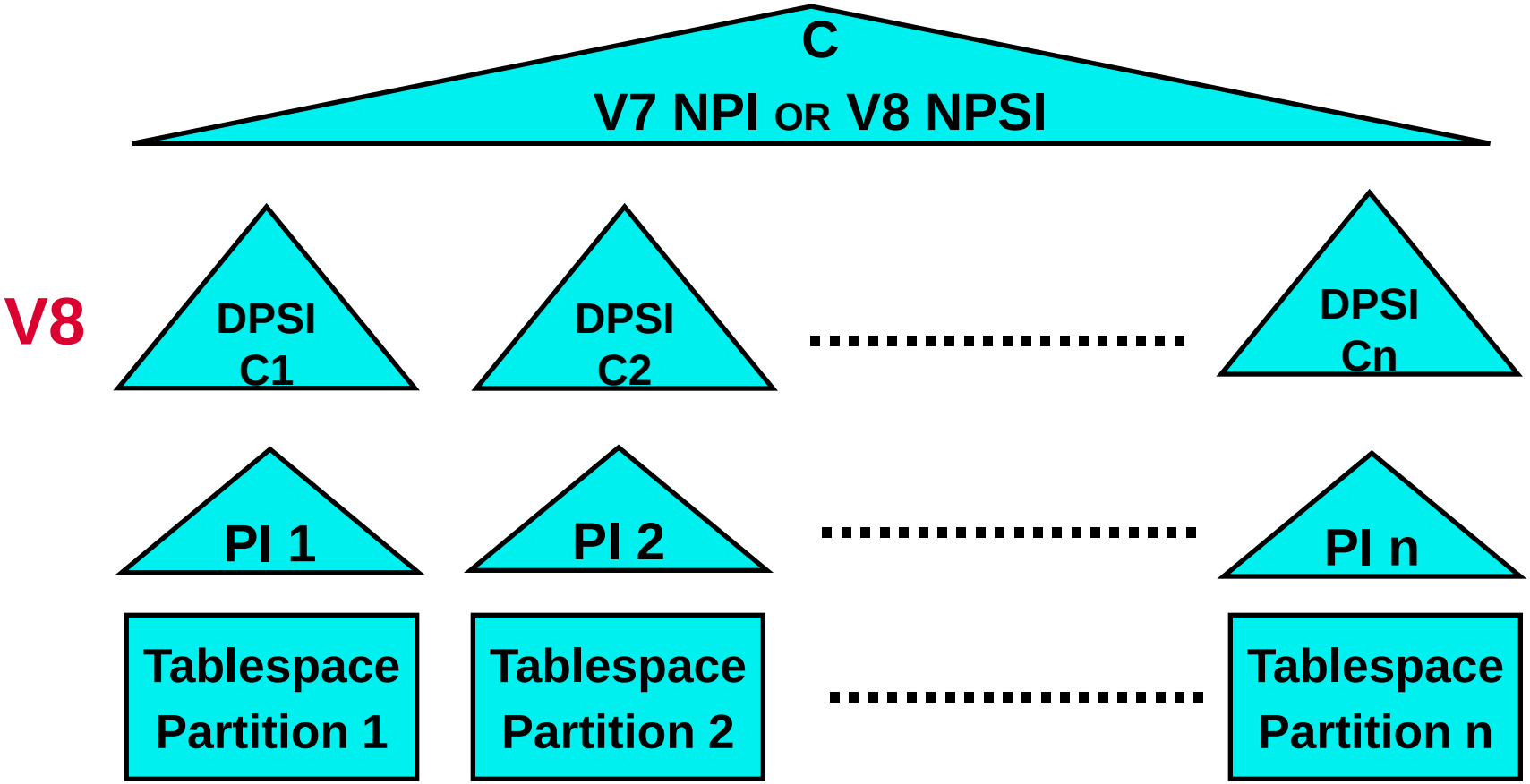
INDEX D IS UNIQUE AND CANNOT BE A DPSI IN V8

It can only be a PI or an NPSI.

In V9 a UNIQUE index can be a DPSI if it is a **SUPERset** of the partitioning COLUMN(s).

NOTE: After a REORG the table data is now in order by CUSTNO, not CUSTNAME within each partition

Parallel Partition Load / Reorg / Rebuild with DPSI versus (NPI(V7) and NPSI(V8))



PIs, DPSIs, & NPSIs

DPSIs

- **FOUR HUGE DISADVANTAGES**

1) IF INDEX IS NEEDED TO AVOID SORT

2) IF INDEX IS NEEDED TO MAINTAIN POSITION OF A DYNAMIC SCROLL CURSOR

3) IF PROGRAM ACCESSES ALL PARTITIONS

WHERE B = :HVB

UP TO 4096 INDEX TREES TO PROBE
USE NPSIs – SAME AS OLD NPIs

4) DPSIs lose the advantage of INDEX LOOKASIDE and SEQUENTIAL DETECTION

A FEW OTHER THINGS cont.

ORDER BY ON SEARCHED SINGLETON SELECTS

REMEMBER LOCK AVOIDANCE FOR SINGLETON SELECTS?

```
SELECT… INTO …
FROM…WHERE LASTNAME LIKE :HV
ORDER BY ZIPCODE
FETCH FIRST ROW ONLY
```

UPDATE OF PARTITION INDEX DATA WITHOUT DRAIN

V7 vs. V8 TECHNIQUE

Base Table

DB2 Table

Accessed by many

V7 STATIC SCROLL CURSORS

Base Table & Result Table are kept in sync

Result Table

DB2 Declared Temp Table

1) Own changes if DECLARE sensitive

&

2) other's changes if FETCH sensitive

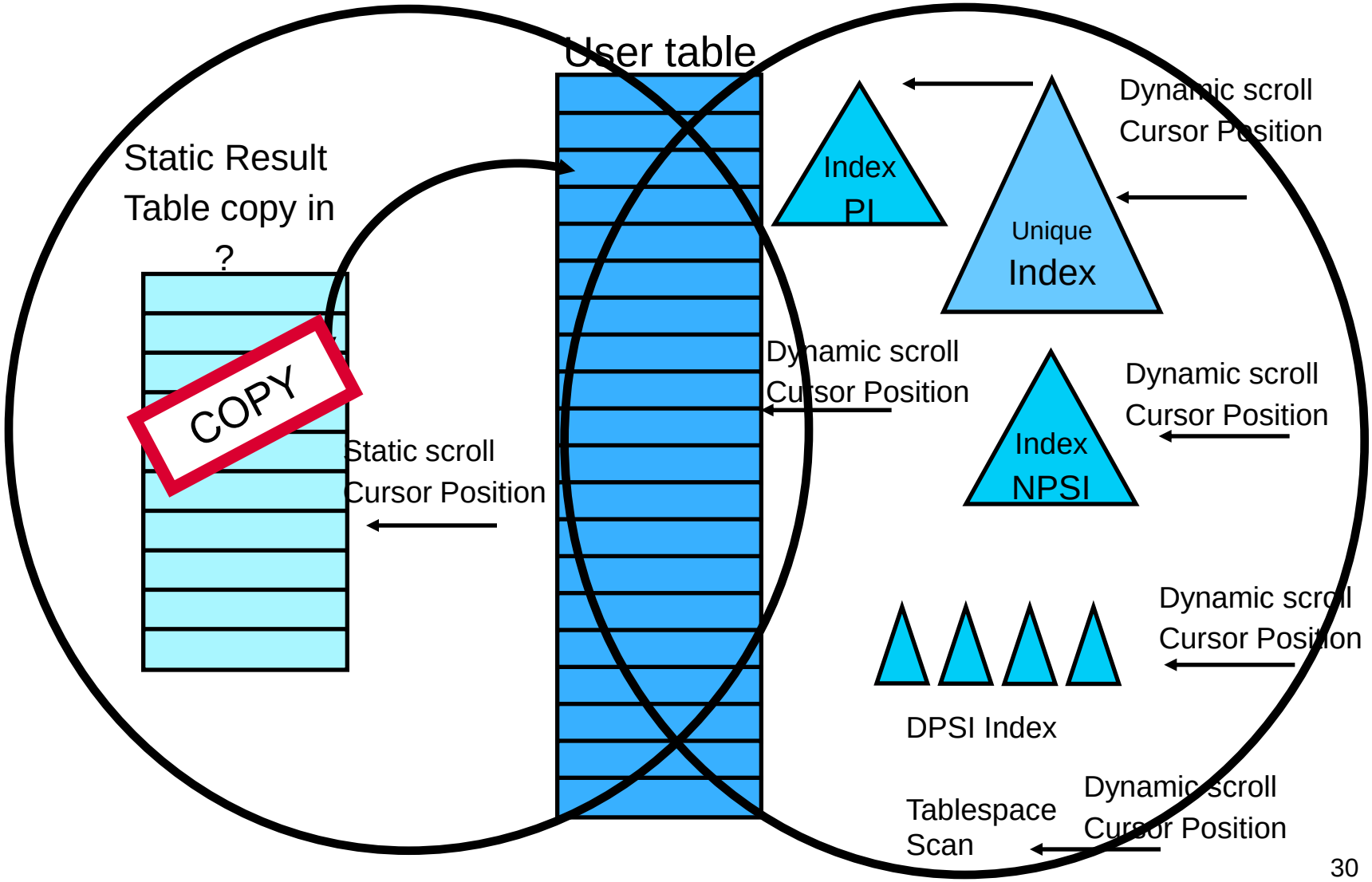
Exclusive access by agent

Fixed number of rows with delete holes and update holes if sensitive

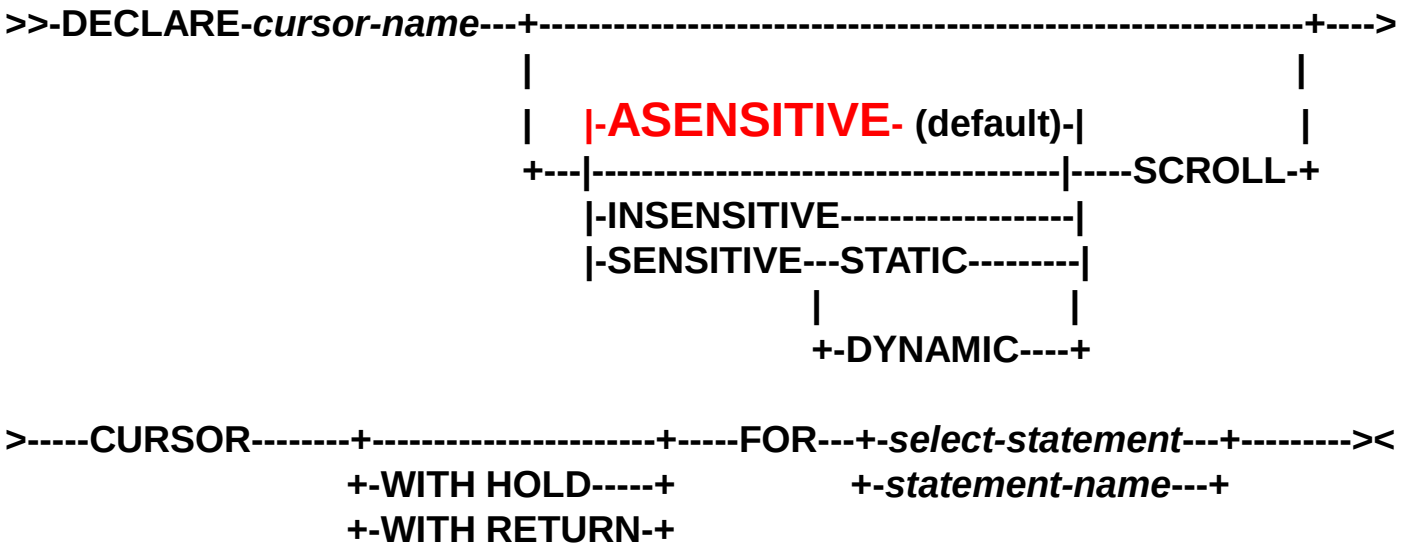
Goes away at Close Cursor

In V7 & V8 requires TEMP database and predefined table spaces

STATIC SCROLL VS. DYNAMIC SCROLL



DECLARE CURSOR SYNTAX – V8 UPDATE



ASENSITIVE

- **Highest sensitivity based off SELECT statement**

  A IS FOR **"AMIBIGUOUS"**

- **IF NO *SCROLL* keyword in CURSOR declaration**

  The CURSOR we have been using for years

  ```
  DECLARE C1 CURSOR FOR
  SELECT * FROM EMP
  ```

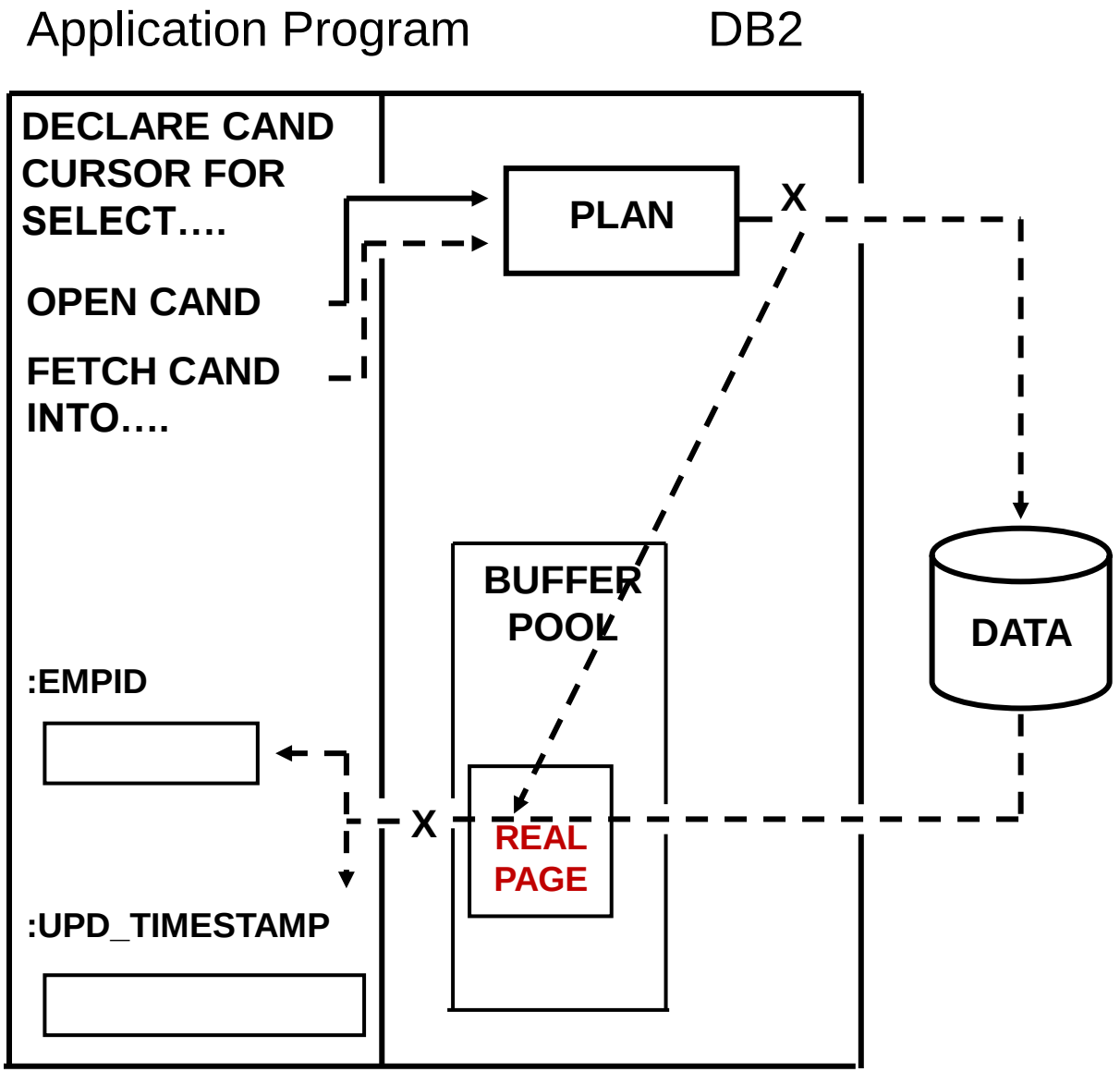
  will result in "SENSITIVE DYNAMIC" (NO MATERIALIZED DATA)

  ```
  DECLARE C1 CURSOR FOR
  SELECT * FROM EMP
  GROUP BY WORKDEPT
  ORDER BY EMPNO
  ```

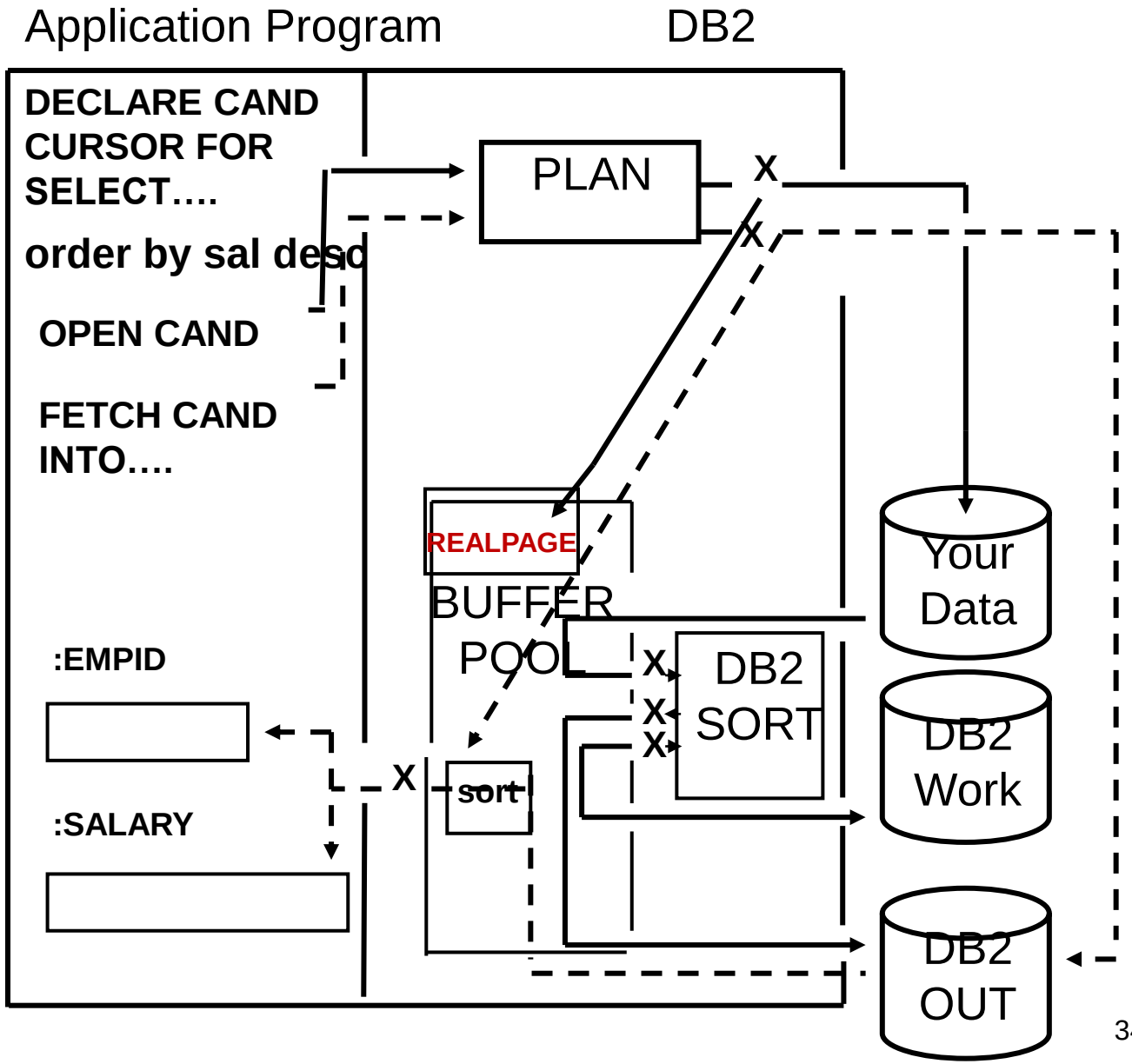
  will result in "INSENSITIVE STATIC" (SORTOUT FILE AT FETCH)

UNDERSTANDING WHAT HAPPENS AT OPEN CURSOR IS KEY TO UNDERSTANDING ASENSITIVE CURSORS

SENSITIVE


Application Program
DB2
DECLARE CAND
CURSOR FOR
SELECT….
OPEN CAND
FETCH CAND
INTO….
PLAN
X
BUFFER
POOL
DATA
:EMPID
X
REAL
PAGE
:UPD_TIMESTAMP

INSENSITIVE



OTHER IMPLICATIONS of DYNAMIC SCROLL CURSORS

- **Predicate evaluation**
  Occurs at every FETCH (in all directions)

- **If Cursor is NOT defined "FOR UPDATE OF…" BUT you are using UPDATE(DELETE) WHERE CURRENT OF…**

  CS & CURRENTDATA(NO) = predicates are reevaluated at UPDATE/DELETE to make sure that row still qualifies BUT other columns (including those selected) are NOT reevaluated

OTHER IMPLICATIONS of DYNAMIC SCROLL CURSORS

- If Cursor is coded with "FOR UPDATE OF…" followed by UPDATE(DELETE) WHERE CURRENT OF

  No need for DB2 to reevaluate predicates; Pessimistic technique to ensure that row will not change or be deleted, a U-lock will be held between the FETCH and UPDATE/DELETE WHERE CURRENT OF…

- Serialization
  - No "built-in optimistic locking" with DYNAMIC SCROLL CURSORS
    Only SENSITIVE STATIC cursors have "built-in optimistic locking" where both the WHERE clause predicates as well as all columns SELECTed are reevaluated at UPDATE/DELETE WHERE CURRENT OF…
  - Standard locking semantics, including lock avoidance options, apply to current ROW(SET)

INTERSECTION WITH OTHER NEW V8 FUNCTIONS

- **Data Partitioning Secondary Indexes (DPSIs)**

  Dynamic scrolling is supported using a DPSI

  If selected as the access path by the optimizer
  DPSI can be used in two ways
  Nuance – SORT syntax or no SORT syntax?

  Externally, DPSI is like any other index

  However, DB2 has specific code to accommodate SORT AVOIDANCE & SCROLLing with DPSIs

BASIC DECISIONS FOR APPLICATION DESIGN

- What "result" type?
  - STATIC - fixed answer set in result table

    Sensitive Static allows UPDATE WHERE CURRENT OF CURSOR (if ambiguous) and provides "built-in optimistic locking"

  - DYNAMIC – no result set

    Answer changes as rows are inserted or deleted

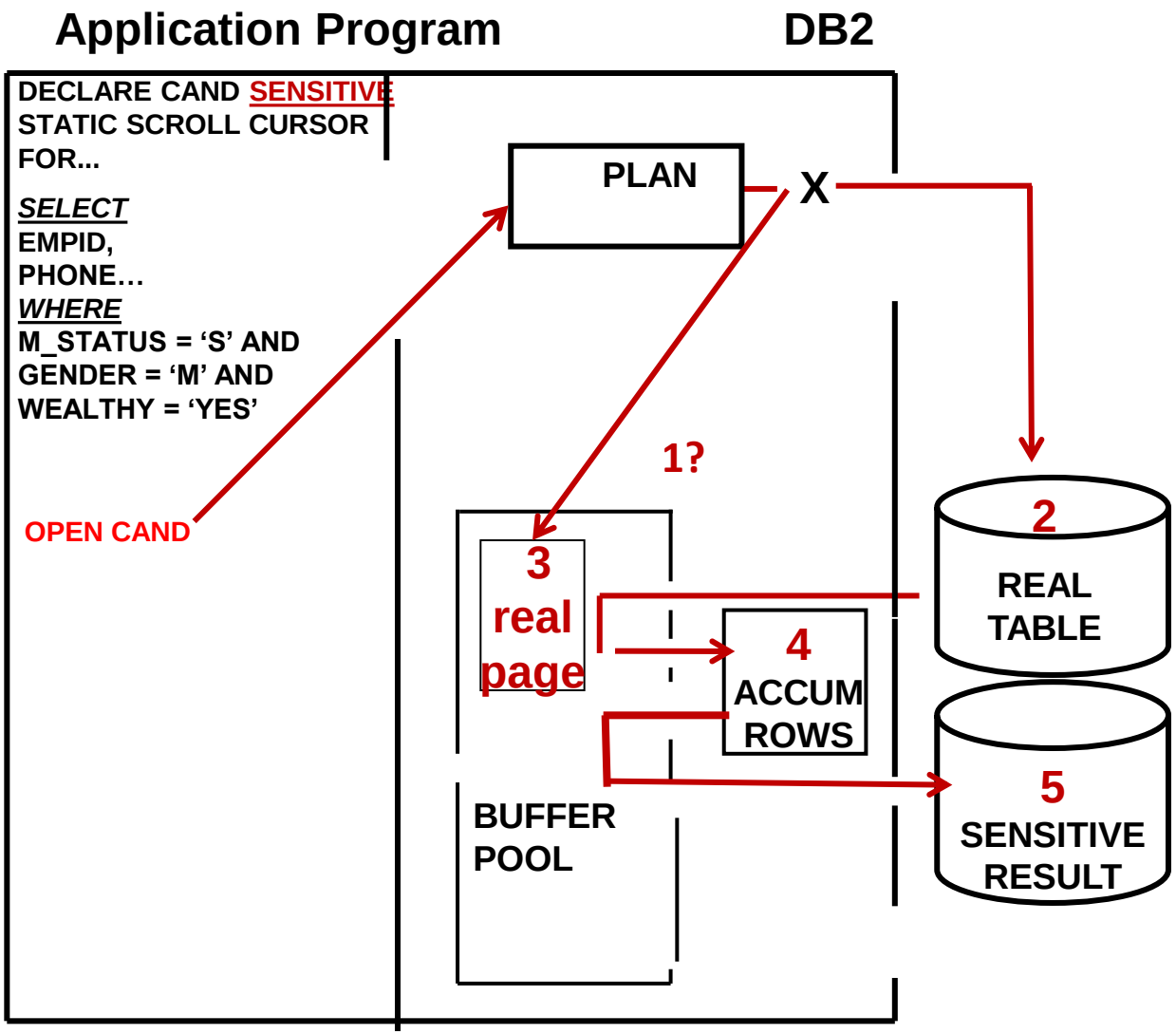
    BIND SUCCESS? Access Path dependent SYNTAX CHECK

    If SORT SYNTAX index must be available to avoid sort

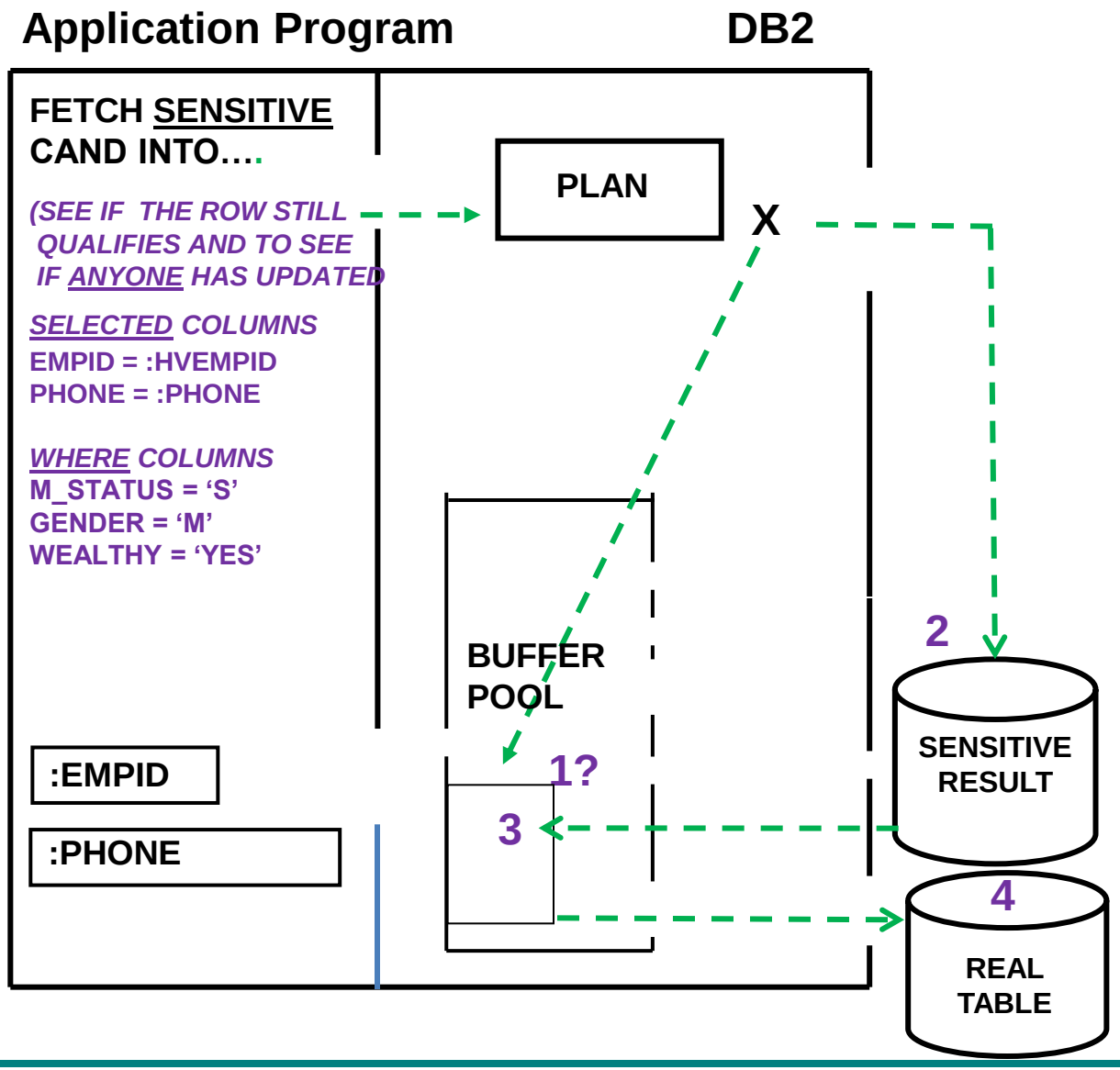
    No joins, functions, complex SQL

    No built-in optimistic locking (only WHERE PREDs are reapplied)

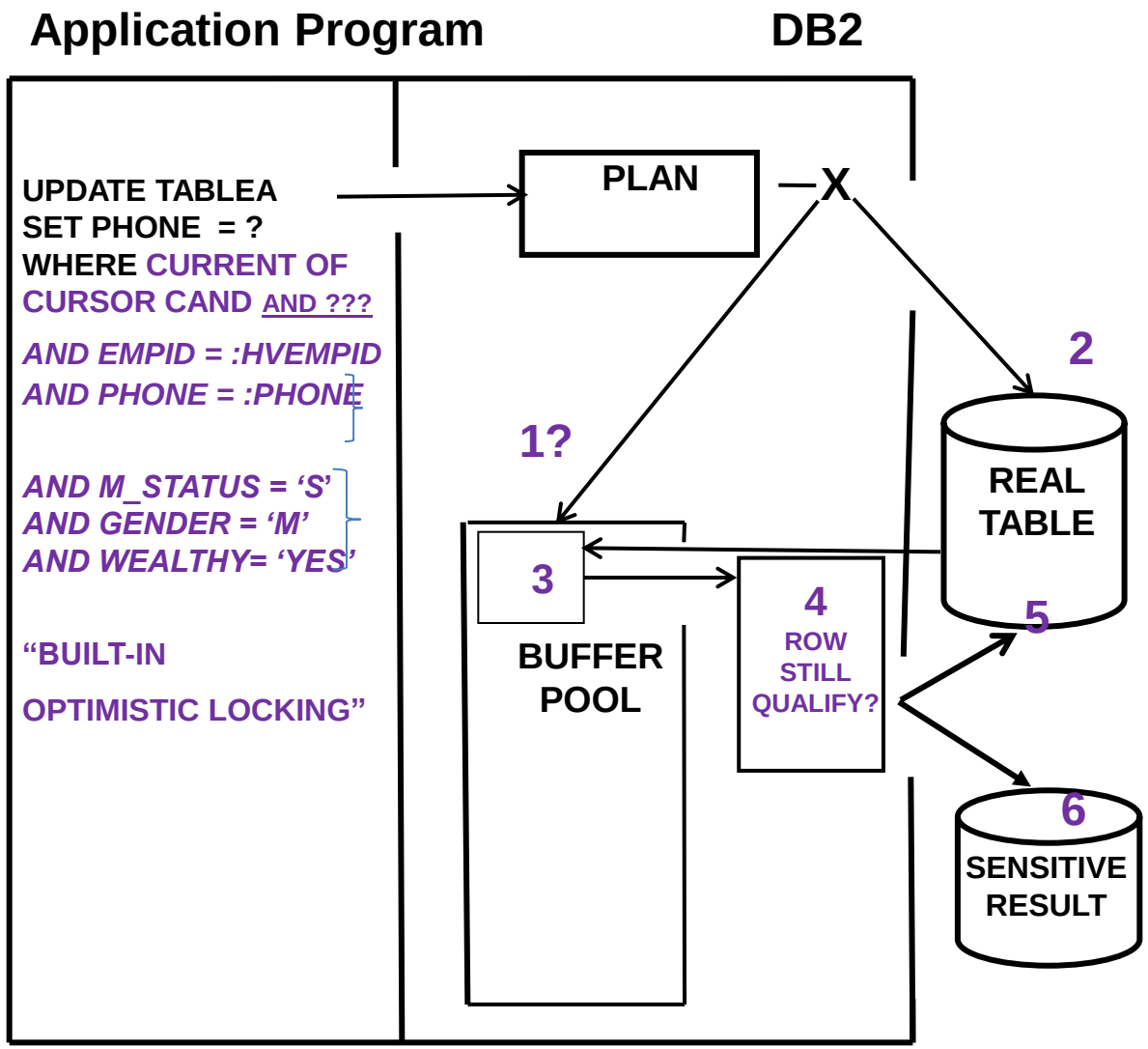
OPEN WITH STATIC SCROLL CURSORS



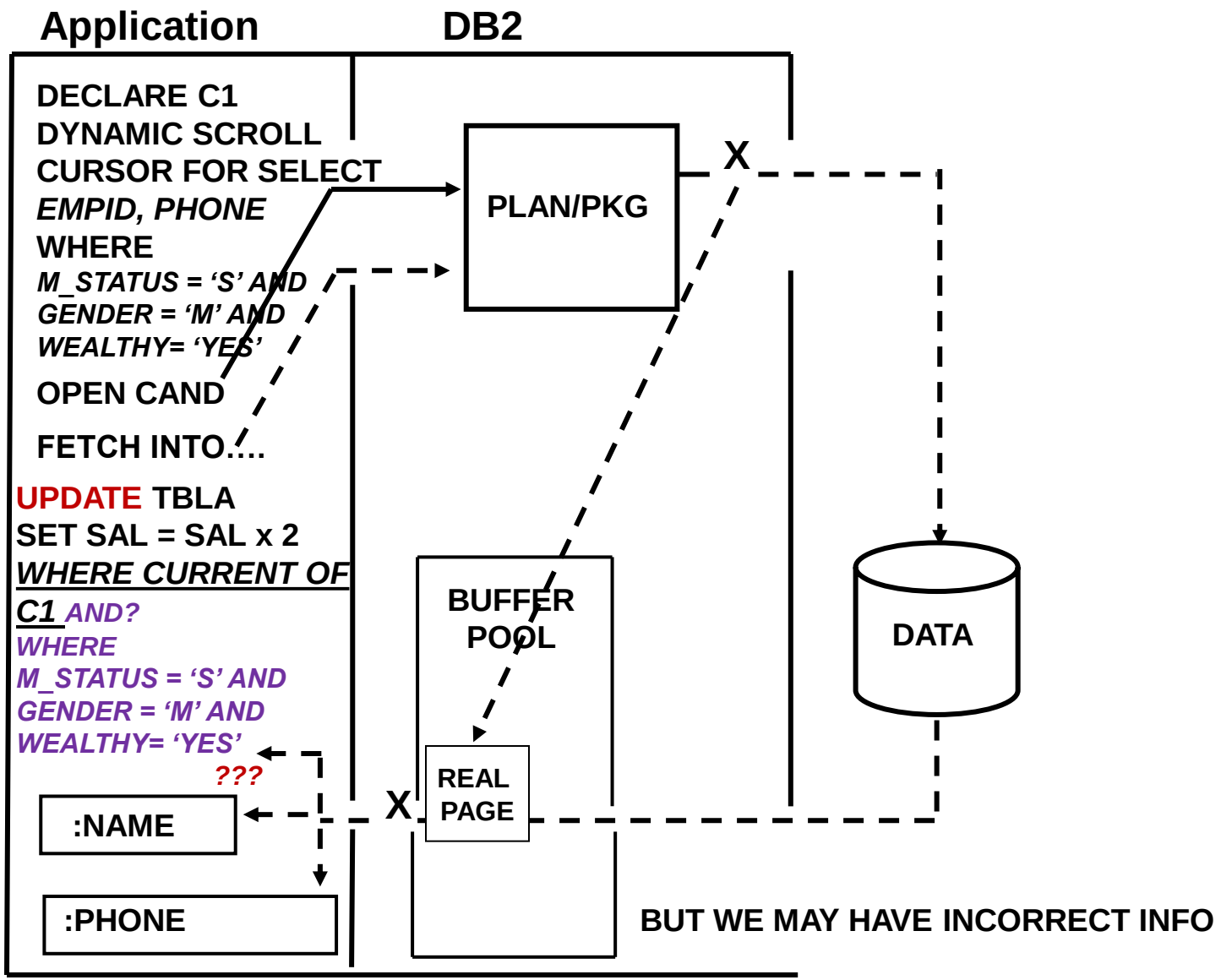
FETCH WITH STATIC SCROLL CURSORS



UPDATE WITH SENSITIVE STATIC CURSOR



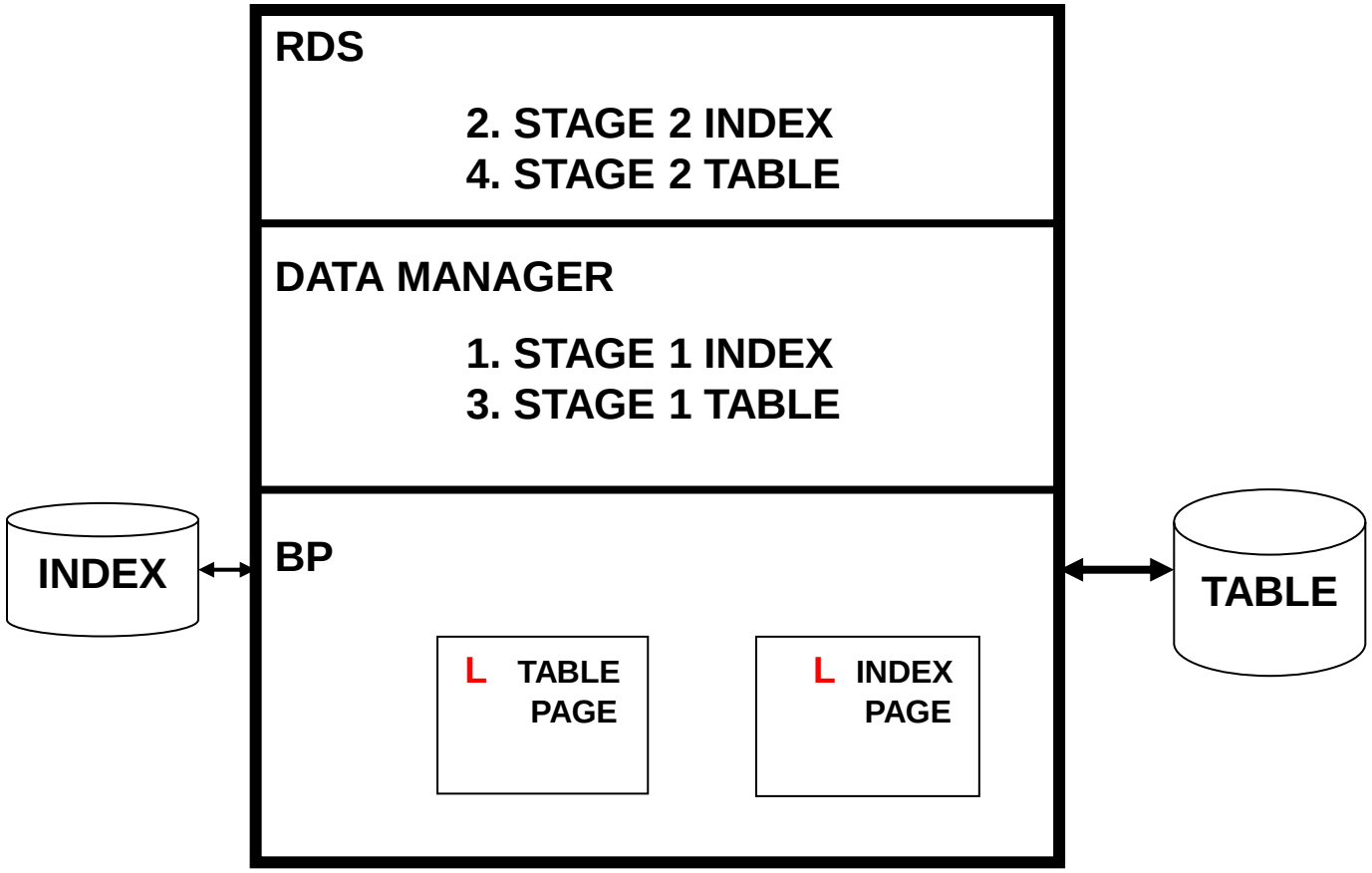
UPDATE WITH DYNAMIC SCROLL CURSOR



PREDICATE EVALUATION

IN WHAT ORDER ARE PREDICATES APPLIED?

V9 vs. V10



PREDICATE EVALUATION

IN WHAT ORDER ARE PREDICATES APPLIED?

INDEX ON B, D, A, E, F

V10

A =

B =

D >

E + F =

G =

H >

I =

J BETWEEN

K IN (…)

L LIKE

M + N =

V9

A =

B =

D >

E + F =

G =

H >

I =

J BETWEEN

K IN (…)

L LIKE

M + N

LATCHES, LOCKS, CLAIMS, & DRAINS

LATCHES: WHAT ARE THEY AND WHY DO WE NEED THEM?

LOCKS: THE NEGATIVE AND THE POSITIVE

AND THEN WE HAVE

CLAIMS: WHEN AND WHY?

DRAINS: WHAT COMPONENT USES THESE?