Esoteric functions in Db2 for z/OS

Roy Boxwell, Software Engineering GmbH

Agenda

- What does “esoteric functions” mean?
- FIT/FTB
- Spatial Indexes
- Regular Expressions
- Clone tables
- Scrollable Cursors

Agenda

- What does “esoteric functions” mean?
- FIT/FTB
- Spatial Indexes
- Regular Expressions
- Clone tables
- Scrollable Cursors

What does “esoteric functions” mean?

What I mean is to describe some, but not all, Db2 functions that are, in my humble opinion:

- Odd
- Strange
- Not well understood
- Not used
- Downright weird

So, please join me on a voyage of discovery…

Agenda

- What does “esoteric functions” mean?
- FIT/FTB
- Spatial Indexes
- Regular Expressions
- Clone tables
- Scrollable Cursors

FIT/FTB

Is not this:

FIT/FTB

OK, I will be honest this is *not* a function, but it sure is badly understood and not just by customers!

The idea behind Fast Index Traversal or Fast Traversal Block was to cache small, unique keyed index data in memory so that the non-leaf pages must not be trawled through. The design was tightly bundled to z hardware as the point of control was a 256 byte “line” of RAM in the L2 cache of the processor.

The use of these “lines” meant incredibly fast look up and usage as long as the data fits into the storage line.

FIT/FTB

But why?

What was so wrong with the good old B-Tree index system?

FIT/FTB

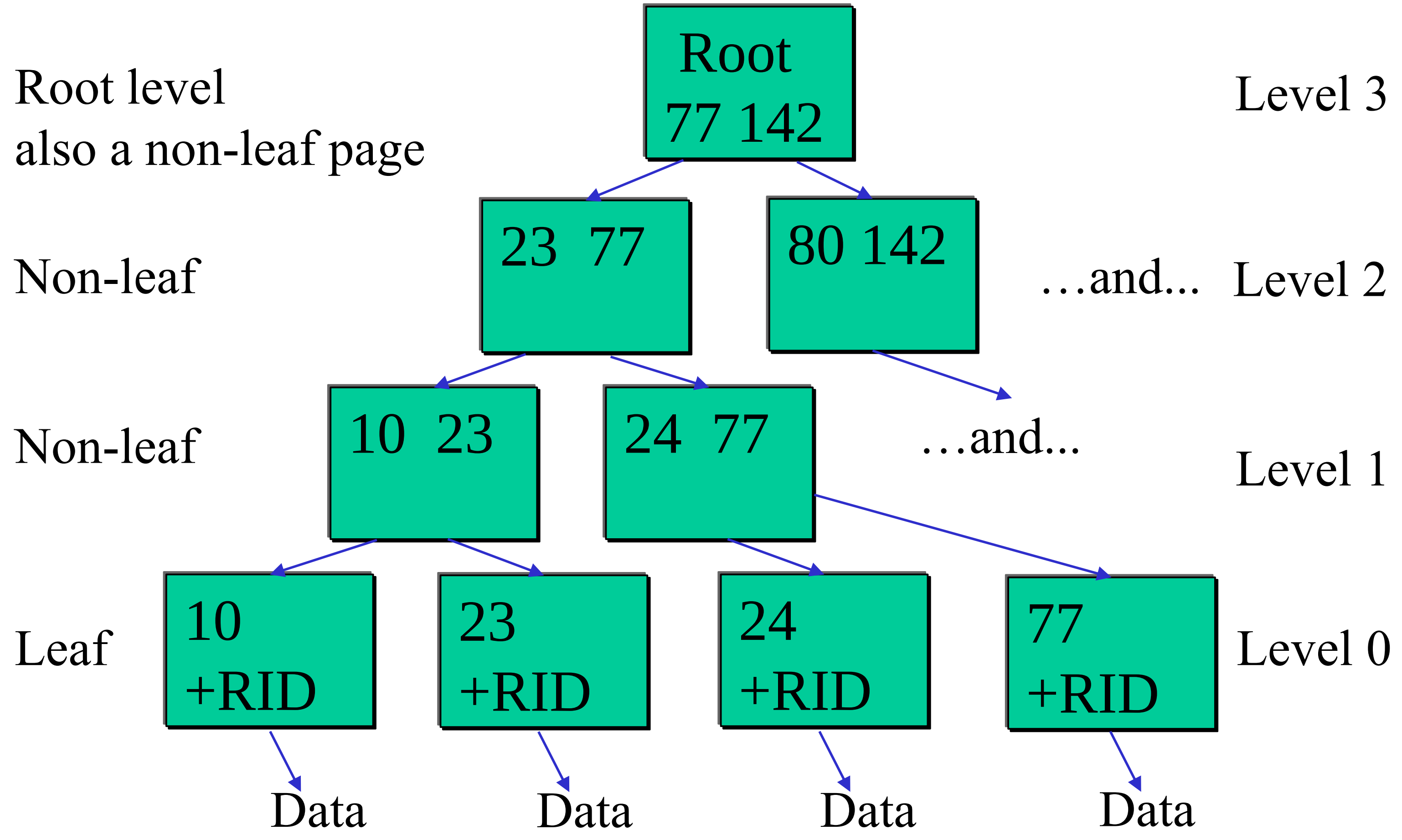
But why?

What was so wrong with the good old B-Tree index system?

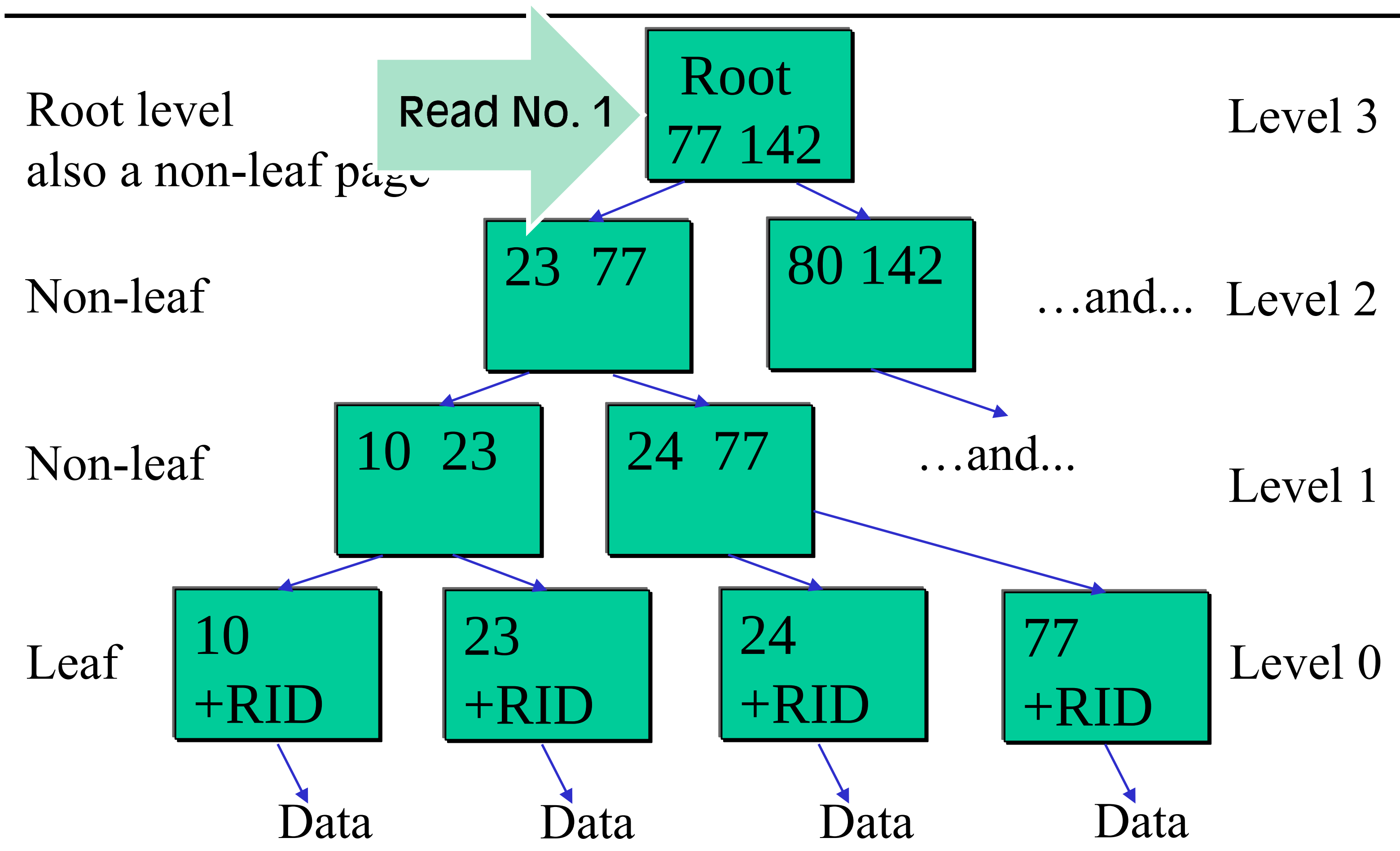
Too many reads!

Let’s have a quick look at a normal Db2 for z/OS b-tree index hierarchy and how it is processed.

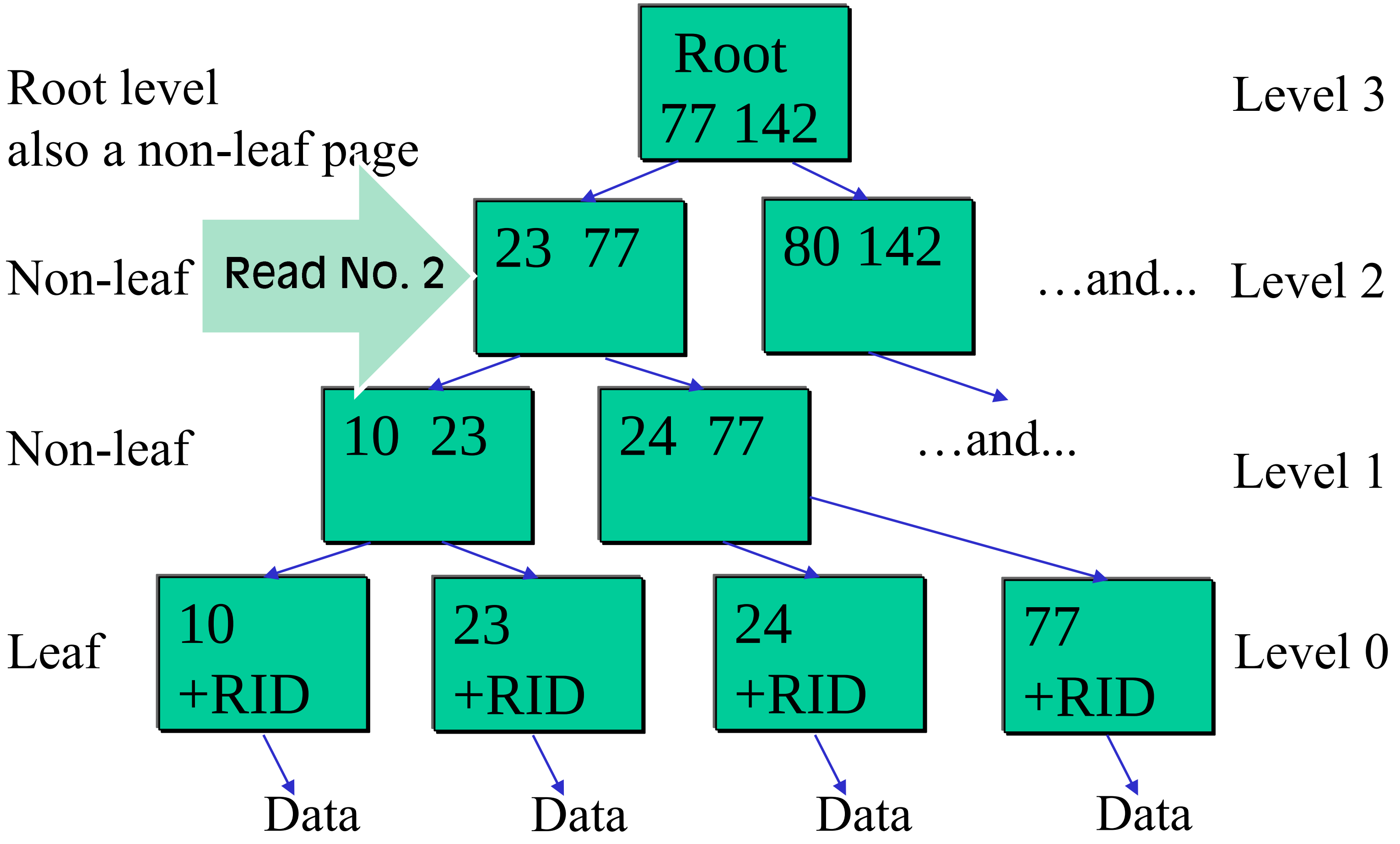
FIT/FTB


Root level
also a non-leaf page
Root
77 142
Level 3
Non-leaf
23 77
80 142
…and...
Level 2
Non-leaf
10 23
24 77
…and...
Level 1
Leaf
10
+RID
23
+RID
24
+RID
77
+RID
Level 0
Data
Data
Data
Data

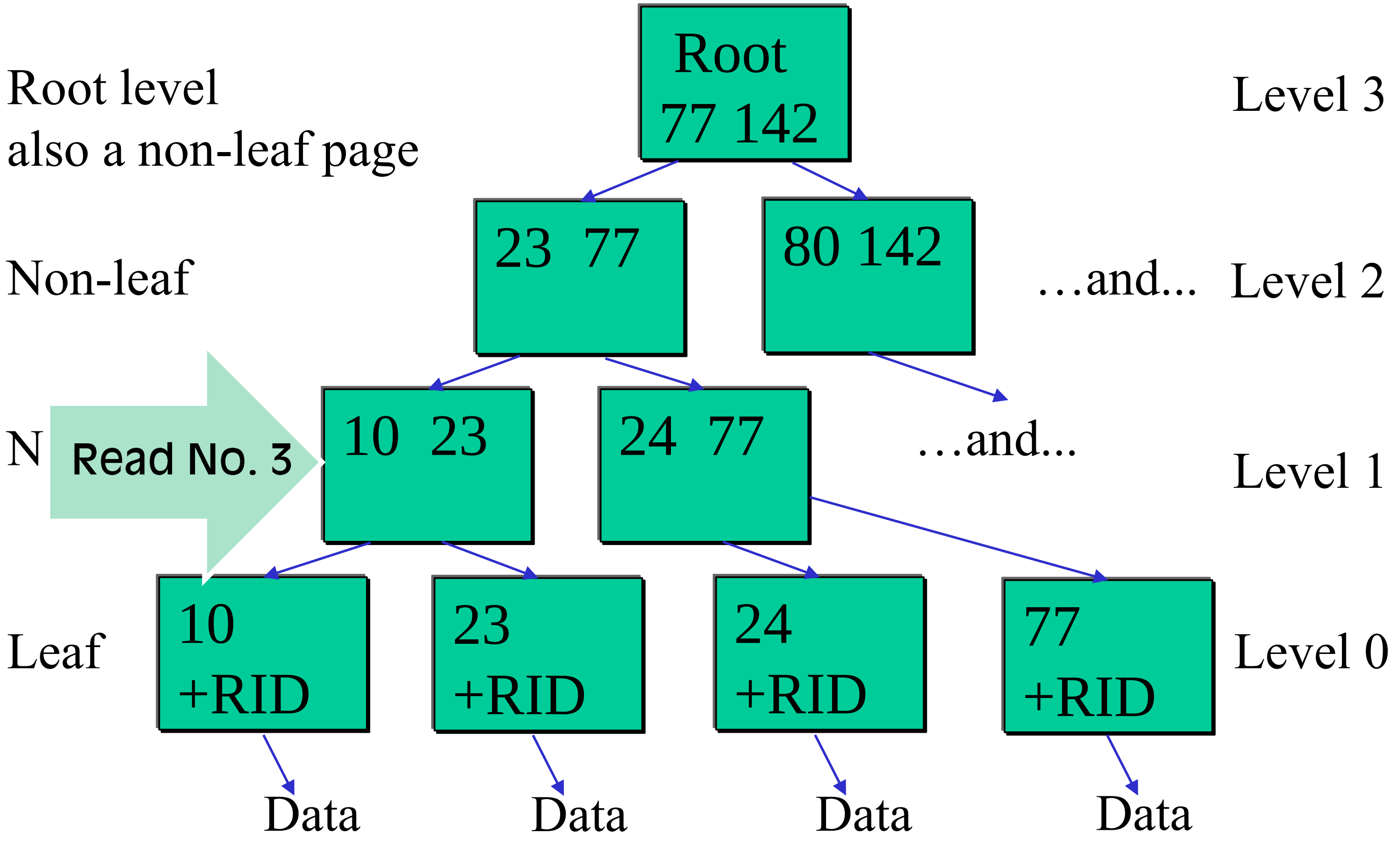
FIT/FTB


Root level
also a non-leaf page
Read No. 1
Root
77 142
Level 3
Non-leaf
23 77
80 142
…and…
Level 2
Non-leaf
10 23
24 77
…and…
Level 1
Leaf
10
+RID
23
+RID
24
+RID
77
+RID
Level 0
Data
Data
Data
Data

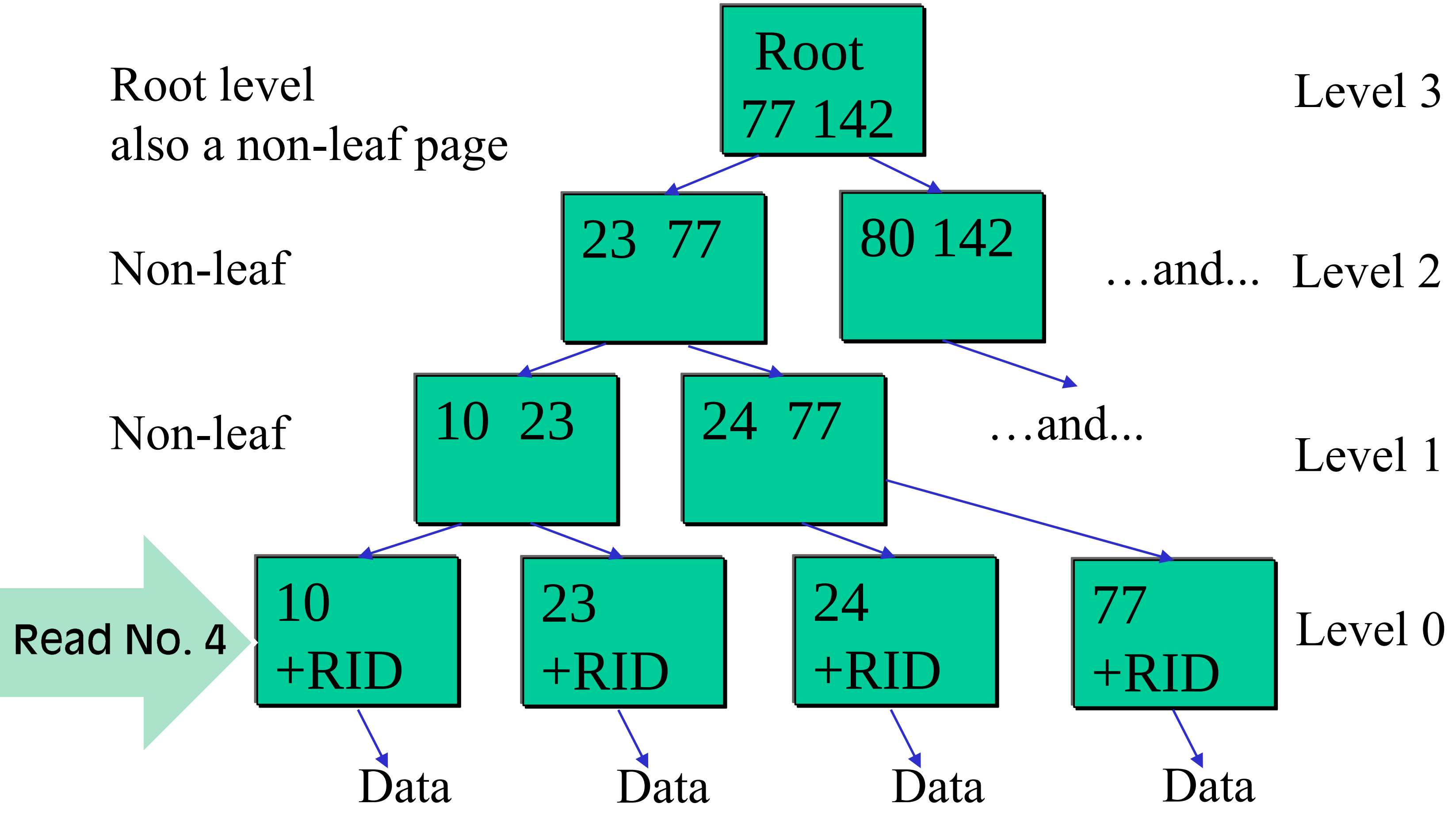
FIT/FTB


Root level
also a non-leaf page
Root
77 142
Level 3
Non-leaf
Read No. 2
23 77
80 142
…and…
Level 2
Non-leaf
10 23
24 77
…and…
Level 1
Leaf
10 +RID
23 +RID
24 +RID
77 +RID
Level 0
Data
Data
Data
Data

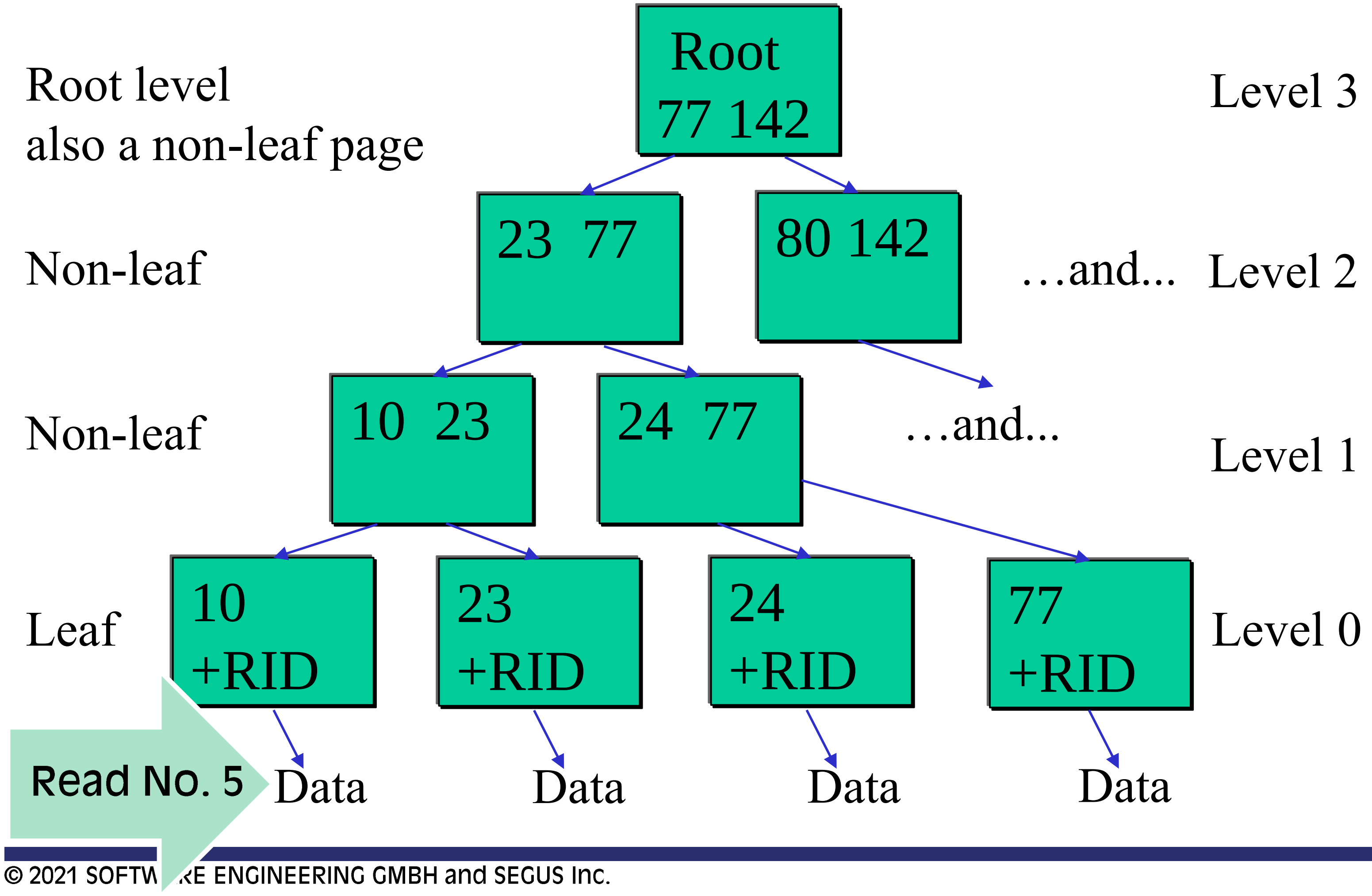
FIT/FTB


Root level
also a non-leaf page
Root
77 142
Level 3
Non-leaf
23 77
80 142
…and…
Level 2
N
Read No. 3
10 23
24 77
…and…
Level 1
Leaf
10
+RID
23
+RID
24
+RID
77
+RID
Level 0
Data
Data
Data
Data

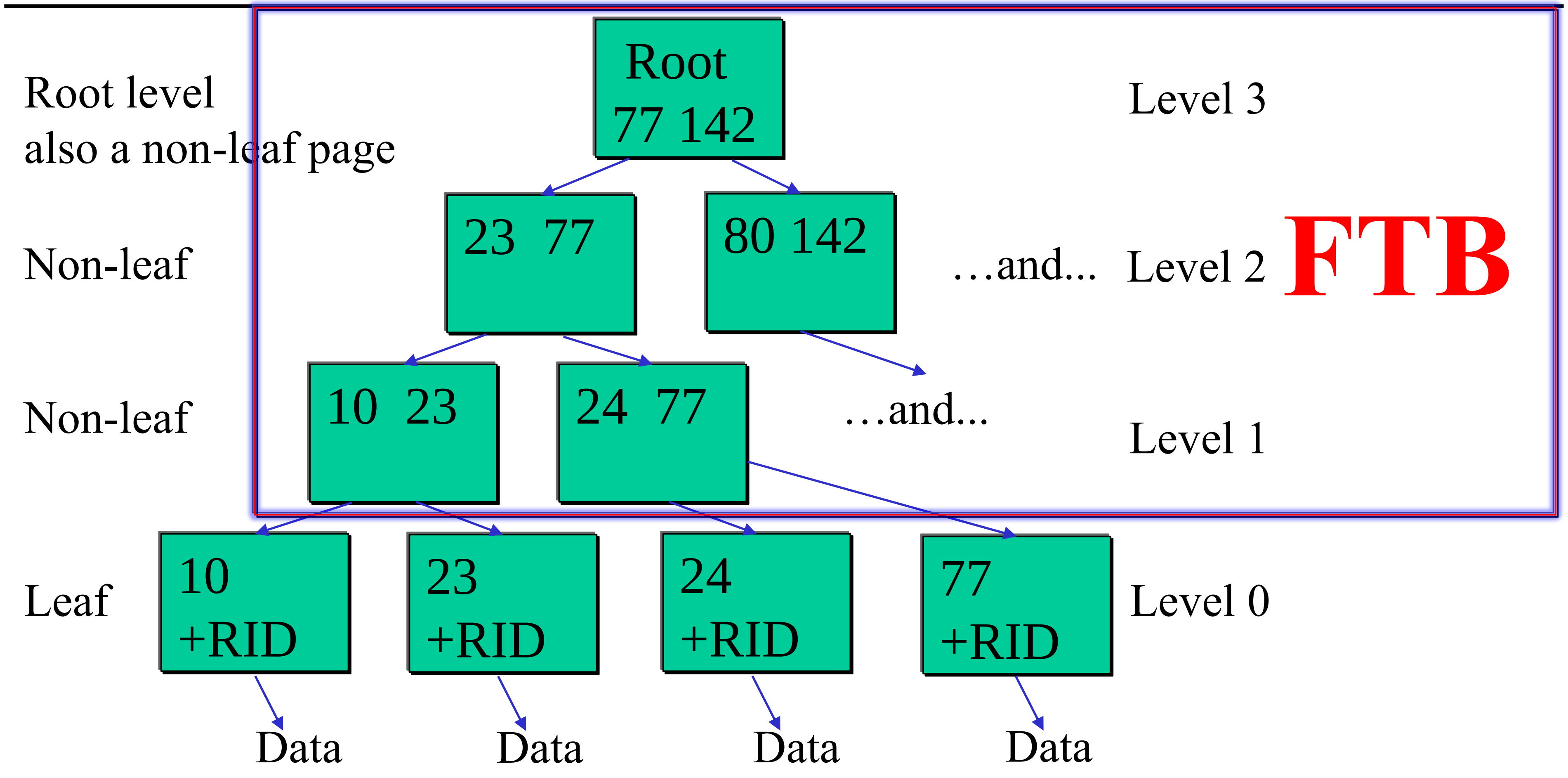
FIT/FTB


Root level
also a non-leaf page
Root
77 142
Level 3
Non-leaf
23 77
80 142
…and…
Level 2
Non-leaf
10 23
24 77
…and…
Level 1
Read No. 4
10
+RID
23
+RID
24
+RID
77
+RID
Level 0
Data
Data
Data
Data

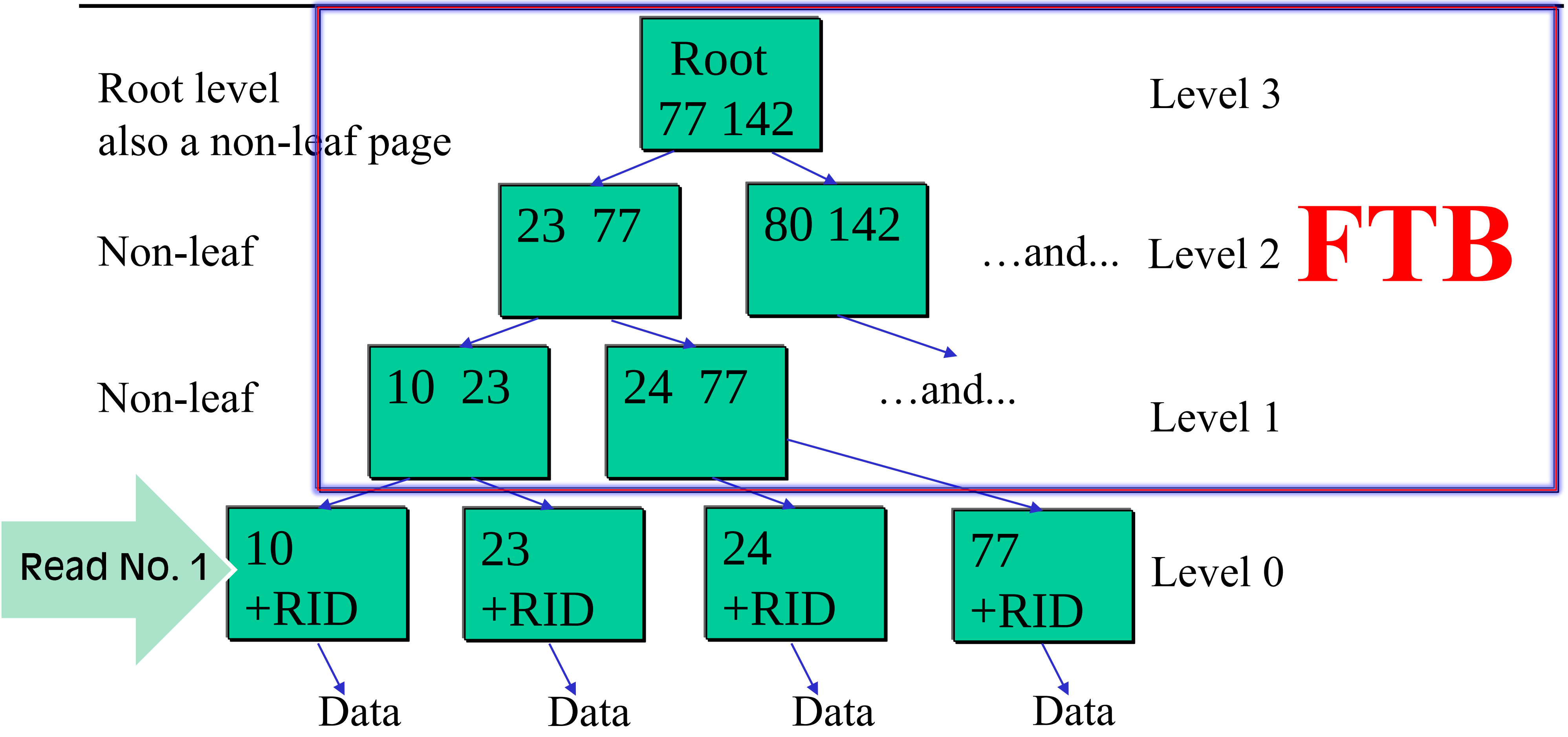
FIT/FTB


Root level
also a non-leaf page
Root
77 142
Level 3
Non-leaf
23 77
80 142
…and...
Level 2
Non-leaf
10 23
24 77
…and...
Level 1
Leaf
10 +RID
23 +RID
24 +RID
77 +RID
Level 0
Read No. 5
Data
Data
Data
Data

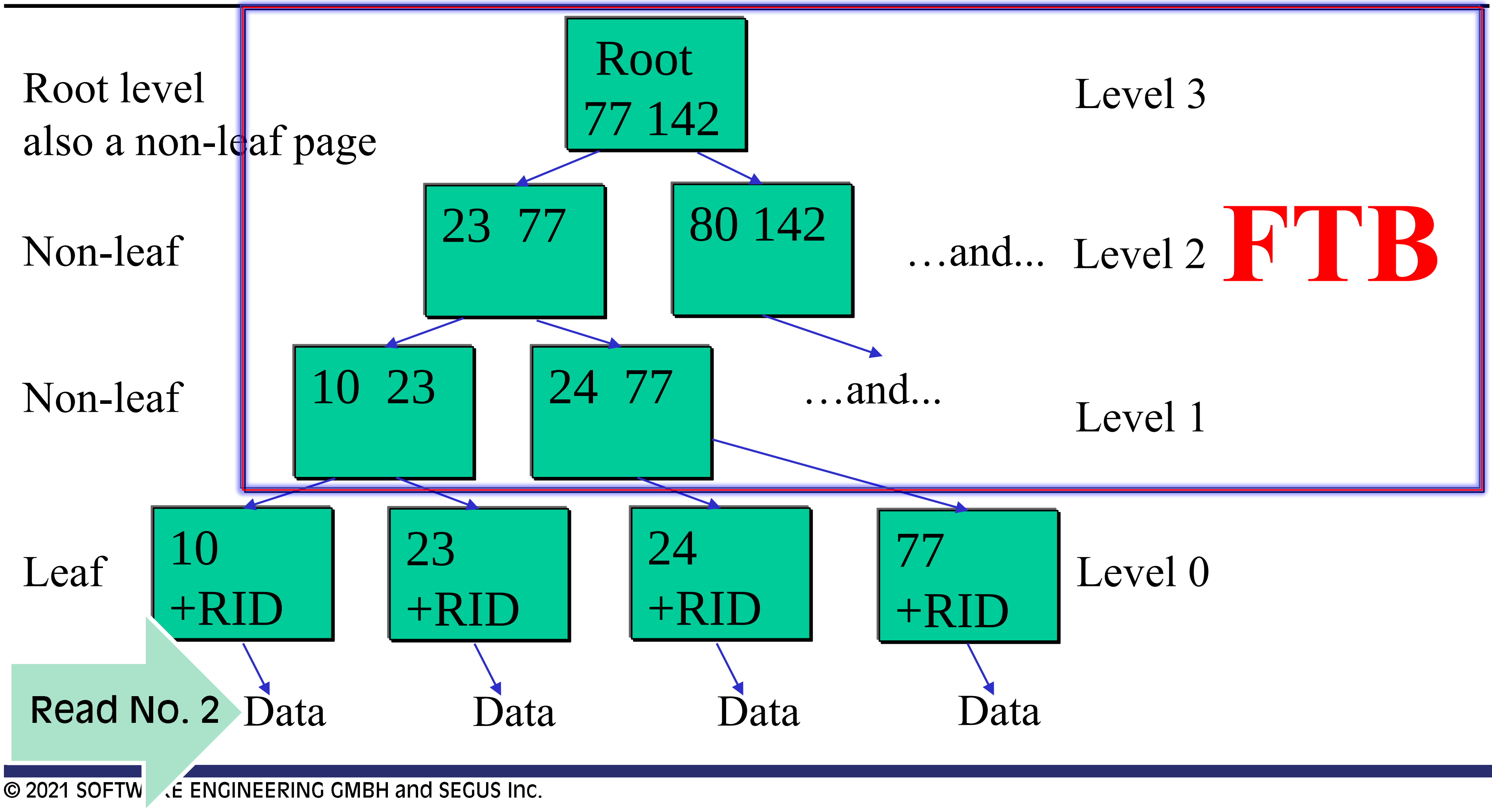
FIT/FTB


Root level
also a non-leaf page
Non-leaf
Non-leaf
Leaf
Root
77 142
23 77
80 142
10 23
24 77
10
+RID
23
+RID
24
+RID
77
+RID
…and…
…and…
Data
Data
Data
Data
Level 3
Level 2
Level 1
Level 0
FTB

FIT/FTB


Root level
also a non-leaf page
Non-leaf
Non-leaf
Read No. 1
Root
77 142
23 77
80 142
…and…
10 23
24 77
…and…
10
+RID
23
+RID
24
+RID
77
+RID
Data
Data
Data
Data
Level 3
Level 2
Level 1
Level 0
FTB

FIT/FTB


Root level
also a non-leaf page
Non-leaf
Non-leaf
Leaf
Root
77 142
23 77
80 142
…and…
10 23
24 77
…and…
10
+RID
23
+RID
24
+RID
77
+RID
Data
Data
Data
Data
Level 3
Level 2
Level 1
Level 0
FTB
Read No. 2

FIT/FTB

You can see that an FTB has *always* just two reads regardless of the depth of the index – this is very good!

It requires just one mutex (which is actually just one assembler instruction compare-and-set) and must *not* latch all of the intervening non-leaf pages – this is very good!

So what's wrong?

FIT/FTB

Well, they were pretty limited in availability:

Unique only, maximum length 64 Bytes
No versioning
No column with type TIMESTAMP with TIMEZONE
Not larger than 2,000,000 leaf pages
No more than 10,000 FTBs (One FTB per partition)

Anything else –> No FTB

Pretty restrictive huh?

FIT/FTB

If you got past the door and were at least a candidate for FTB, how and when did you become one?

A new daemon is now running checking all the candidate indexes every two minutes.

There is an internal counter and it is adjusted in various ways depending on what has happened to the index since the last check...

FIT/FTB

Index usage:

Any random index traversal, index only or index plus data?

- Super! +

Used for lookaside?

- Not good! –

Used for sequential access?

- Not so good! –

Caused an index split?

- Very bad! (Halves the internal counter!) – –

Every two minutes the daemon compares these results with an internal threshold and then the index is either FTB or not...

FIT/FTB

Set up and control:

The physical size of the FTBs is controlled by ZPARM:

INDEX_MEMORY_CONTROL

With valid values AUTO, DISABLE, or 10 – 200,000 MBs. Default is AUTO and then the size is capped at 20% of all bufferpools or 10MB whichever is the highest.

-DISPLAY STATS(INDEXMEMORYUSAGE) or its baby sibling (IMU)

This shows you, in message DSNT783I, which indexes are being processed and how much memory they are taking up.

FIT/FTB

Set up and control:

With APAR PH34859 (PTF UI75254 Closed 2021-05-05) a new command was added:

-DISPLAY STATS(INDEXTRAVERSECOUNT) or its baby sibling (ITC)

This shows you a list of traverse counts, in descending order, in message DSNT830I all filterable on DBNAME, SPACENAM and PART.

FIT/FTB

Set up and control:

Two new IFCIDs were also created:

IFCID 389 is part of statistics trace class 8. It records all indexes that use fast index traversal in the system.

IFCID 477 is part of performance trace class 4 and records the allocation and deallocation activities of FTBs for fast index traversal.

FIT/FTB

Set up and control:

IFCID 2 (statistics record) got several new fields:

- **Current threshold for FTB creation.**
- **Number of FTB candidates.**
- **Total size allocated for all FTBs.**
- **Number of FTBs currently allocated.**
- **Number of objects that meet the criteria for FTB creation.**

FIT/FTB

Set up and control:

For the fine control, what I call micro-management, there is a new catalog table SYSIBM.SYSINDEXCONTROL where you can limit which index is available for FTB and exactly when.

I would never recommend using this table, if at all possible, with the one exception of excluding certain indexes, for whatever reason, from FTB processing completely.

FIT/FTB

Set up and control:

Remember from now on:

If you ALTER ADD a column to an index, you want to *check* beforehand if you are going to put it over 64 bytes. It might well be being used as an FTB and saving tons of CPU and I/O, and your ALTER will destroy this!

FIT/FTB

Changes in FL508:

Docu update:

> **„Columns in the INCLUDE list do not count toward the size limit“**

This means that the INCLUDED columns are naturally *not* used for FTB.

FIT/FTB

Changes in FL508:

Docu update:

> **„Columns in the INCLUDE list do not count toward the size limit"**

This means that the INCLUDED columns are naturally *not* used for FTB.

Duplicate index support!

FIT/FTB

Changes in FL508:

Docu update:

„Columns in the INCLUDE list do not count toward the size limit“

This means that the INCLUDED columns are naturally *not* used for FTB.

Duplicate index support!

Length limited to 56 bytes or less…

FIT/FTB

Changes in FL508:

New ZPARM:

FTB_NON_UNIQUE_INDEX

Controls whether or not to even consider duplicate indexes for FTB processing. Default is NO.

FIT/FTB

Queries:

Here’s a couple of queries to review all candidate indexes you have and to see how close to the limits for unique and non-unique you are getting!

FIT/FTB

```
WITH INPUT (NLEVELS, LENGTH, TABLE_NAME, INDEX_NAME) AS
(SELECT B.NLEVELS
      , SUM(CASE D.COLTYPE
            WHEN 'DECIMAL'  THEN
                            SMALLINT(CEILING((D.LENGTH + 1 ) / 2 ))
            WHEN 'GRAPHIC'  THEN D.LENGTH * 2
            WHEN 'VARG'     THEN D.LENGTH * 2
            WHEN 'LONGVARG' THEN D.LENGTH * 2
            ELSE D.LENGTH
            END)
      + SUM(CASE B.PADDED
            WHEN 'Y' THEN 0
            ELSE
                CASE D.COLTYPE
                WHEN 'VARG'     THEN 2
                WHEN 'LONGVARG' THEN 2
                WHEN 'VARCHAR'  THEN 2
                WHEN 'LONGVAR'  THEN 2
                WHEN 'VARBIN'   THEN 2
                WHEN 'DECFLOAT' THEN 2
                ELSE 0
                END
            END)
```

FIT/FTB

```
+ SUM(CASE D.NULLS
            WHEN 'Y' THEN 1
            ELSE 0
            END) AS LENGTH
       , STRIP(D.TBCREATOR) CONCAT '.' CONCAT STRIP(D.TBNAME)
       , STRIP(B.CREATOR)   CONCAT '.' CONCAT STRIP(B.NAME)
 FROM SYSIBM.SYSINDEXES B
     ,SYSIBM.SYSKEYS    C
     ,SYSIBM.SYSCOLUMNS D
WHERE B.UNIQUERULE NOT IN ('D','N')         -- NOT DUPLICATE
  AND D.COLTYPE        <> 'TIMESTZ'         -- NOT TIMEZONE
  AND B.DBID            > 6                 -- NOT DIR/CAT
  AND B.OLDEST_VERSION  = B.CURRENT_VERSION -- NOT VERSIONED
  AND C.ORDERING       <> ' '               -- NO INCLUDE/IOE
  AND B.TBNAME          = D.TBNAME
  AND B.TBCREATOR       = D.TBCREATOR
  AND B.NAME            = C.IXNAME
  AND B.CREATOR         = C.IXCREATOR
  AND C.COLNAME         = D.NAME
GROUP BY D.TBCREATOR, D.TBNAME, B.CREATOR, B.NAME, B.NLEVELS)
SELECT NLEVELS, LENGTH , INDEX_NAME
FROM INPUT
WHERE LENGTH <= 64
```

FIT/FTB

```
-- ORDER BY NLEVELS DESC, LENGTH DESC -- IF STATISTICS ARE GOOD
ORDER BY LENGTH DESC, INDEX_NAME
FOR FETCH ONLY
WITH UR
;
```

The commented out ORDER BY makes sense if the statistics are all up to date and then shows you the “best” candidates first as the higher the number of levels the better the savings when using FTBs.

FIT/FTB

```
WITH INPUT (NLEVELS, LENGTH, TABLE_NAME, INDEX_NAME) AS
(SELECT B.NLEVELS
      , SUM(CASE D.COLTYPE
            WHEN 'DECIMAL'  THEN
                             SMALLINT(CEILING((D.LENGTH + 1 ) / 2 ))
            WHEN 'GRAPHIC'  THEN D.LENGTH * 2
            WHEN 'VARG'     THEN D.LENGTH * 2
            WHEN 'LONGVARG' THEN D.LENGTH * 2
            ELSE D.LENGTH
            END)
      + SUM(CASE B.PADDED
            WHEN 'Y' THEN 0
            ELSE
                CASE D.COLTYPE
                WHEN 'VARG'     THEN 2
                WHEN 'LONGVARG' THEN 2
                WHEN 'VARCHAR'  THEN 2
                WHEN 'LONGVAR'  THEN 2
                WHEN 'VARBIN'   THEN 2
                WHEN 'DECFLOAT' THEN 2
                ELSE 0
                END
            END)
```

FIT/FTB

```
+ SUM(CASE D.NULLS
            WHEN 'Y' THEN 1
            ELSE 0
            END) AS LENGTH
      , STRIP(D.TBCREATOR) CONCAT '.' CONCAT STRIP(D.TBNAME)
      , STRIP(B.CREATOR)   CONCAT '.' CONCAT STRIP(B.NAME)
FROM SYSIBM.SYSINDEXES B
    ,SYSIBM.SYSKEYS    C
    ,SYSIBM.SYSCOLUMNS D
WHERE B.UNIQUERULE     IN ('D','N')         -- DUPLICATE
  AND D.COLTYPE        <> 'TIMESTZ'         -- NOT TIMEZONE
  AND B.DBID            > 6                 -- NOT DIR/CAT
  AND B.OLDEST_VERSION  = B.CURRENT_VERSION -- NOT VERSIONED
  AND C.ORDERING       <> ' '               -- NO INCLUDE/IOE
  AND B.TBNAME          = D.TBNAME
  AND B.TBCREATOR       = D.TBCREATOR
  AND B.NAME            = C.IXNAME
  AND B.CREATOR         = C.IXCREATOR
  AND C.COLNAME         = D.NAME
GROUP BY D.TBCREATOR, D.TBNAME, B.CREATOR, B.NAME, B.NLEVELS)
SELECT NLEVELS, LENGTH, INDEX_NAME
FROM INPUT
WHERE LENGTH <= 56
```

FIT/FTB

```
-- ORDER BY NLEVELS DESC, NLENGTH DESC -- IF STATISTICS ARE GOOD
ORDER BY LENGTH DESC, INDEX_NAME
FOR FETCH ONLY
WITH UR
;
```

The commented out ORDER BY makes sense if the statistics are all up to date and then shows you the “best” candidates first as the higher the number of levels the better the savings when using FTBs.

FIT/FTB

APAR list as of 2021-11-09:

```
APAR    Closed     PTF     Description
PH28182 2020-09-25 UI71784 INDEX LOOK ASIDE SUPPORT WHEN INDEX FAST TRAVERSE BLOCK(FTB) IS IN USE
PH29102 2020-10-27 UI72276 ABEND04E DSNKTRAV ERQUAL505B RC00C90101 FTB TRAVERSAL
PH29336 2020-09-22 UI71351 IRLM CORRECT RESULTANT HELD STATE FOR FTB PLOCKS WHEN PLOCK EXIT WOULD
                              HAVE EXITTED WITH ERROR.
PH29676 2020-10-16 UI72118 ABEND04E RC00C90101 AT DSNKTRAV 5058 DURING INSERT VIA FTB
PH30978 2021-06-01 UI75643 SUBSYSTEM PARAMETER TO ENABLE INDEX IN-MEMORY OPTIMIZATION (FTB) FOR
                              NON-UNIQUE INDEXES
PH34468 2021-04-20 UI75007 ABEND04E RC00C90101 AT DSNKTRAV ERQUAL5021 VIA FTB TRAVERSAL
PH34859 2021-05-05 UI75254 DB2 12 FOR Z/OS NEW FUNCTION FOR FTB (FAST TRAVERSE BLOCKS)
PH35596 2021-04-07 UI74814 INSERT SPLITTING PAGE INTO FTB LEAF NODE GOT DSNKFTIN:5002 ABEND
                              BECAUSE OLD PAGE THAT CAUSE THE PAGE SPLIT WAT MISSING IN FTB.
PH36406 2021-05-07 UI75288 INSERT KEY INTO FTB PROCESS DETECTING INCONSISTENT STRUCTURE
                              MODIFICATION NUMBER THEN GOT DSNKFTIN:5043 ABEND
PH36434 2021-05-13 UI75392 DB2 12 FOR Z/OS INTERNAL SERVICEABILITY UPDATE (Improve Create / Free
                              FTB log recs)
```

FIT/FTB

APAR list as of 2021-11-09:

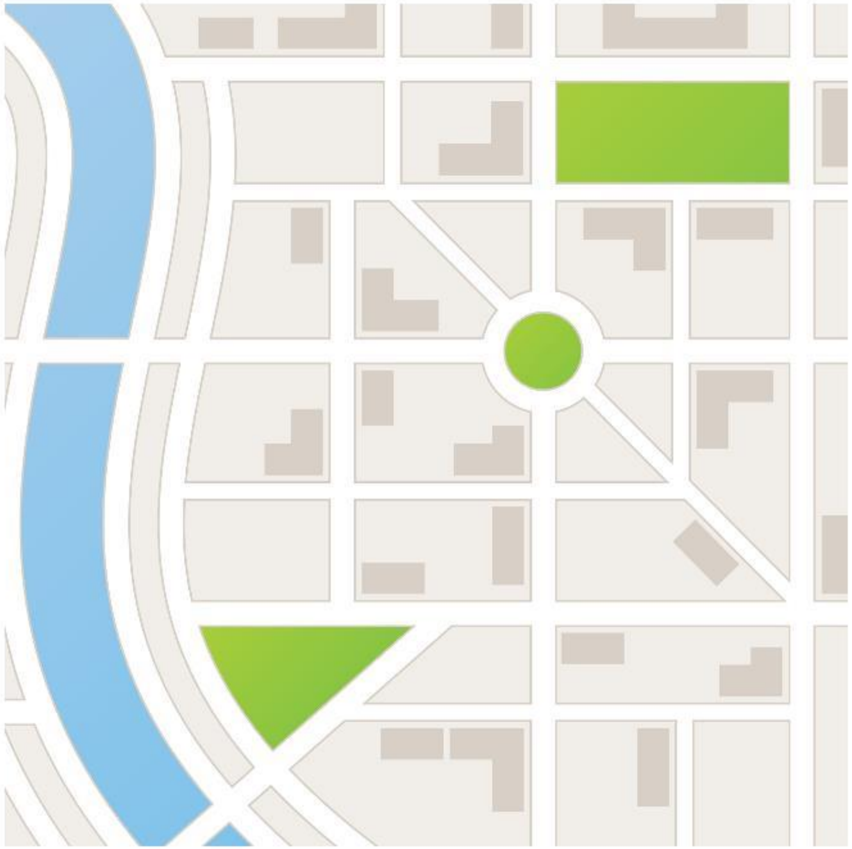
```
APAR    Closed     PTF     Description
PH36531 2021-05-13 UI75391 ABEND04E RC00C90101 AT DSNKINSN ERQUAL5009 AND DSNKFTIN ERQUAL5066 FOR
                             FTB INSERT PLOCK FAILURE
PH36978 2021-06-18 UI75978 FTB MESSAGE MSGDSNT351I ISSUED INCORRECTLY
PH38212 2021-07-07 UI76239 ABEND04E RC00C90101 AT DSNKFTBU ERQUAL5061 AND DSNK1CNE ERQUAL5006
                             DURING FTB CREATION
PH39105 2021-10-18 UI77687 DB2 12 FTB INDEXTRAVERSECOUNT = 4294967295 FOR OBJECTS NOT ENABLED FOR
                             FTB
PH40269 2021-09-16 UI77189 ABEND04E RC00E72068 AT DSNXSRME OFFSET01024 DUE TO A TIMING WINDOW
                             WHEN USING INDEX FAST TRAVERSE BLOCK (FTB)
PH40273 2021-11-09 UI78000 IMPROVE PERFORMANCE OF FTB STORAGE POOL ADMF INDEX MANAGER CL20
PH40539 2021-10-07 UI77500 FTB DEADLOCK OCCURS WITH SYSTEM ITASK – CORRID=014.IFTOMK01
```

PH40273 introduces a new 20 minute cycle when Db2 contracts the ADMF INDEX MANAGER CL20 (Hiperspace) storage pool due to serious fragmentation issues.

Agenda

- What does “esoteric functions” mean?
- FIT/FTB
- Spatial Indexes
- Regular Expressions
- Clone tables
- Scrollable Cursors

Spatial Indexes



Spatial Indexes

These are very interesting beasts and are really great if you wish to work with areas of data or locations.

You cannot simply create a “Spatial Index,” you must call a stored procedure to do it for you.

It might sound obvious but you need spatial data before you can create a spatial index!

Spatial Indexes

To get spatial data you can use one of the many DB2GSE stored procs like this:

```
INSERT INTO USA.ZIPCODES
  SELECT DB2GSE.ST_POINT('POINT (LONGITUDE LATITUDE)' , 1003)
        ,LONGITUDE
        .
        .
```

It uses as input the POINT of the co-ordinates, in this case Longitude and Latitude and the all important SRS_ID (Spatial Reference System), in my case 1003 as it was based on an old file containing ZIPCODEs using the WGS84 system.

Spatial Indexes

You should have a list of SRS's like this:

```
SELECT * FROM DB2GSE.ST_SPATIAL_REFERENCE_SYSTEMS;

---------+---------+---------+-------------+-
SRS_NAME                                SRS_ID
---------+---------+---------+-------------+-
DEFAULT_SRS                                  0
NAD83_SRS_1                                  1
NAD27_SRS_1002                            1002
WGS84_SRS_1003                            1003
DE_HDN_SRS_1004                           1004
```

Spatial Indexes

You should have a list of Units of Measure starting like this:

```
SELECT UNIT_NAME FROM DB2GSE.ST_UNITS_OF_MEASURE ;
---------+---------+---------
UNIT_NAME
---------+---------+---------
METRE
FOOT
US SURVEY FOOT
CLARKE'S FOOT
FATHOM
NAUTICAL MILE
GERMAN LEGAL METRE
US SURVEY CHAIN
US SURVEY LINK
US SURVEY MILE
KILOMETRE
CLARKE'S YARD
CLARKE'S CHAIN
CLARKE'S LINK
BRITISH YARD (SEARS 1922)
```

Spatial Indexes

An insert using the POINT spatial proc looks like:

```
INSERT INTO USA.ZIPCODES
VALUES (DB2GSE.ST_POINT('POINT (-71,013202 43,005895)' , 1003)
    , -71,013202 , 43,005895 ,'NH', -5 , 1 , 210 ,'Portsmouth’)
;
```

These inserts are pretty expensive in cpu, by the way!

Spatial Indexes

How to select spatial data using the ST_WITHIN and ST_BUFFER spatial procs:

```
SELECT A.CITY, COUNT(*)
FROM USA.ZIPCODES A
WHERE DB2GSE.ST_WITHIN(A.LOCATION,DB2GSE.ST_BUFFER (
        (SELECT B.LOCATION
         FROM USA.ZIPCODES B
         WHERE B.ZIP = 85009)      -- phoenix
 , 15 , 'STATUTE MILE')) = 1
GROUP BY A.CITY
;
```

This returns all cities, with how many zipcodes, within 15 statute miles of Phoenix.

Spatial Indexes

There are about 70 different spatial functions by the way.

In this SQL I just used two:

- ST_BUFFER has three parameters. The first is a "geometry" or LOCATION type field, then a distance, and finally the units. What it does is create a geometric space that is centered on the input geometry and is then the number of units around it (A radius in this case as we have a point as the input geometry). Thus, we have a "space" of 15 statute miles in radius centered on Phoenix AZ (Well actually on the location of zip code 85009 but that is near enough for me!).
- ST_WITHIN has two parameters which are both "geometries" and if one is within the other it returns a 1 else a 0 thus enabling the simple SQL I wrote.

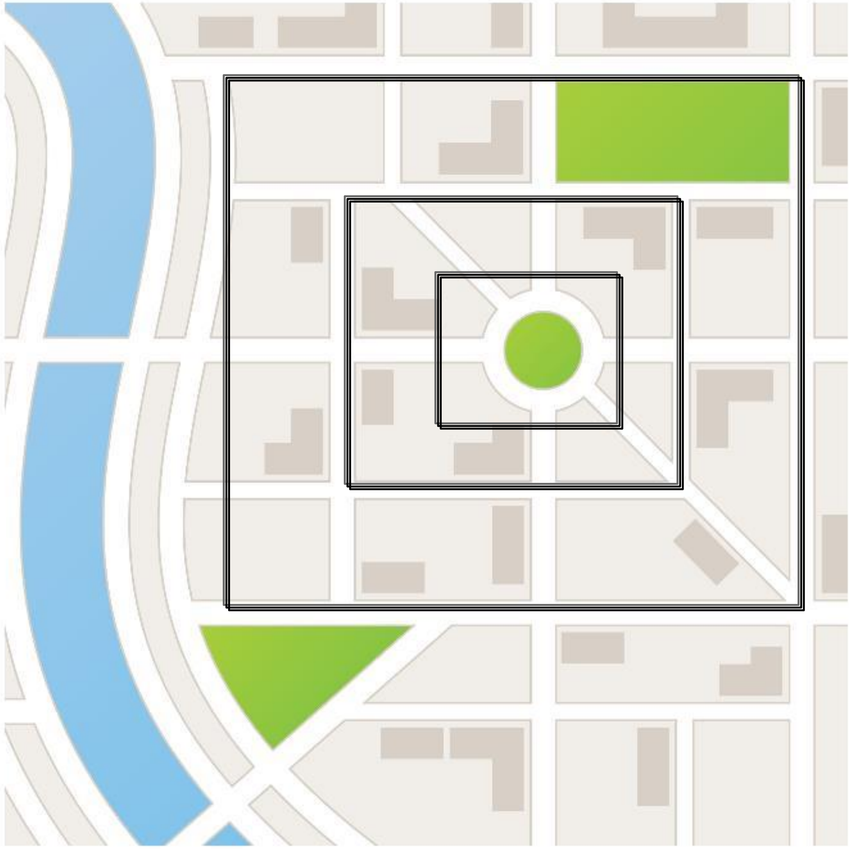
Spatial Indexes

Ok, we now have Spatial data but of course it runs *very* slowly, so now we, finally, get around to creating a Spatial Index!

In order to create a SPATIAL INDEX you must use a stored procedure with a bunch of parameters:

sysproc.ST_create_index (*table_schema/NULL*, *table_name* , *column_name* , *index_schema/NULL*, *index_name* , *other_index_options/NULL*, *grid_size1* , *grid_size2* , *grid_size3* , *msg_code* , *msg_text*)

Spatial Indexes



Grid sizes

Spatial Indexes

Spatial indexes generate a spatial grid index using the minimum bounding rectangle (MBR) of a geometry.

A spatial grid index divides a region into logical square grids with a fixed size that you specify when you create the index. The spatial index is constructed on a spatial column by making one or more entries for the intersections of each geometry's MBR with the grid cells. An index entry consists of the grid cell identifier, the geometry MBR, and the internal identifier of the row that contains the geometry.

You can define up to three spatial index levels (grid levels). Using several grid levels is beneficial because it allows you to optimize the index for different sizes of spatial data.

Spatial Indexes

In the command are the interesting “grid_size” columns. This is one area where you can really tweak performance.

The number of Grid levels, maximum three, this should be the best fit to the different sizes of grids that will be contained within the data. If all you cells are the same size use one, if all different three!

Per Grid you then define the cell size and this is very important as it determines the granularity of the resulting grid. The best value is the lowest number that suits your map grid “finest fit”. The larger the value the smaller the index.

Spatial Indexes

If all three levels are used and the index intersection is bigger than the third then the index fails over into a system defined overflow index.

This overflow index should be avoided for maximum performance.

For example, if multiple grid levels exist, the indexing algorithm attempts to use the lowest grid level possible to provide the finest resolution for the indexed data. When a geometry intersects more than four grid cells at a given level, it is promoted to the next higher level, (provided that there is another level). Therefore, a spatial index that has the three grid levels of 10.0, 100.0, and 1000.0 will first intersect each geometry with the level 10.0 grid. If a geometry intersects with more than four grid cells of size 10.0, it is promoted and intersected with the level 100.0 grid. If more than four intersections result at the 100.0 level, the geometry is promoted to the 1000.0 level. If more than 10 intersections result at the 1000.0 level, the geometry is indexed in the overflow level.

Spatial Indexes

sysproc.ST_create_index (*table_schema/NULL*, *table_name* , *column_name* , *index_schema/NULL*, *index_name* , *other_index_options/NULL*, *grid_size1* , *grid_size2* , *grid_size3* , *msg_code* , *msg_text*)

Setting the grid_size2 to a non-zero value gives you a two level grid and then setting the grid_size3 to a non-zero value gives you a three level grid.

Spatial Indexes

Using the command processor DSN5SCLP the syntax looks like:

```
DSN5SCLP /create_idx ZA00QA1A +
 -tableschema USA -tablename ZIPCODES -columnname LOCATION +
 -indexschema USA -indexname LOC_IX +
 -otherIdxOpts "FREEPAGE 0" +
 -gridSize1 1.0 -gridSize2 2.0 -gridSize3 3.0
```

When it executes it just tells you this:

```
********************************* TOP OF DATA *****
GSE0000I  The operation was completed successfully.
******************************** BOTTOM OF DATA ***
```

Spatial Indexes

When do Spatial index's get used by the optimizer?

Only when the SQL contains one of the functions in the following list and the value on the right hand side of the predicate is 1 an index can be used:

- ST_Contains
- ST_Crosses
- ST_Distance (Predicate must be less than the left hand side)
- EnvelopesIntersect
- ST_Equals
- ST_Intersects
- ST_Overlaps
- ST_Touches
- ST_Within

Spatial Indexes

Problems?

Number one is <still> no LOAD is allowed…
Number two is no SHRLEVEL CHANGE on REBUILD INDEX

Spatial Indexes

Benefits?

Lots! Here's the benchmark data of 100 runs of the Phoenix SQL earlier:

Without Spatial Index		**With Spatial Index**	
Elapsed	**CPU**	**Elapsed**	**CPU**
202 seconds	186 seconds	25 seconds	23 seconds

So they have their uses!

Agenda

- What does “esoteric functions” mean?
- FIT/FTB
- Spatial Indexes
- Regular Expressions
- Clone tables
- Scrollable Cursors

Regular expressions

/r/e/g/e/x

Regular expressions

<u>**Yes!**</u>

They are available, and not just on computers you can lift!

Now it is ugly, very ugly, in fact it is the ugliest thing I have ever seen and I am not a very toxic person normally…

Let us begin with a blog entry for LUW:

> Well… on the “boxes you can lift” they have had regex for a long time, all built into the Db2 Engine. Fred Sobotka’s article “Advanced Pattern Matching with Regular Expressions in DB2 11.1 for LUW” in the IDUG Blog shows lots of really cool ways of using REGEXP_LIKE and its brethren and is well worth a read.

Regular expressions

Here is a regex that lists out all tables that start with between two and five characters ranging from B to Z and then ends with just two numerics:

```
SELECT NAME, creator from SYSIBM.SYSTABLES
WHERE
XMLEXISTS('$newXDoc[fn:matches(., "^[B-Z]{2,5}[0-9]{2}$")]' PASSING
      XMLQUERY('<doc>{$xmltbname}</doc>' PASSING NAME as "xmltbname")
      as "newXDoc")
order by 1
fetch first 10 rows only
;
---------+---------+---------+---------+---------+---------+---------
NAME
---------+---------+---------+---------+---------+---------+---------
CERNT01
CERNT02
CERNT04
```

It will not win a beauty contest any day soon but, hey, it works!

Regular expressions

Ok, to understand what on earth that sql does I will now strip it down into its component parts:

```
SELECT NAME, creator from SYSIBM.SYSTABLES
WHERE
XMLEXISTS('$newXDoc[fn:matches(., "^[B-Z]{2,5}[0-9]{2}$")]' PASSING
      XMLQUERY('<doc>{$xmltbname}</doc>' PASSING NAME as "xmltbname")
      as "newXDoc")
order by 1
fetch first 10 rows only
;
```

A normal “select” statement right down to the WHERE

Regular expressions

Ok, to understand what on earth that sql does I will now strip it down into its component parts:

```
SELECT NAME, creator from SYSIBM.SYSTABLES
WHERE
XMLEXISTS('$newXDoc[fn:matches(., "^[B-Z]{2,5}[0-9]{2}$")]' PASSING
      XMLQUERY('<doc>{$xmltbname}</doc>' PASSING NAME as "xmltbname")
      as "newXDoc")
order by 1
fetch first 10 rows only
;
```

This part is where the magic is actually happening and must be viewed as being two separate statements!

Regular expressions

Ok, to understand what on earth that sql does I will now strip it down into its component parts:

```
SELECT NAME, creator from SYSIBM.SYSTABLES
WHERE
XMLEXISTS('$newXDoc[fn:matches(., "^[B-Z]{2,5}[0-9]{2}$")]' PASSING
      XMLQUERY('<doc>{$xmltbname}</doc>' PASSING NAME as "xmltbname")
      as "newXDoc")
order by 1
fetch first 10 rows only
;
```

The inner part “translates” the column NAME into an XML construct called xmltbname ready for the outer part that does the regex.

Regular expressions

Ok, to understand what on earth that sql does I will now strip it down into its component parts:

```
SELECT NAME, creator from SYSIBM.SYSTABLES
WHERE
XMLEXISTS('$newXDoc[fn:matches(., "^[B-Z]{2,5}[0-9]{2}$")]' PASSING
      XMLQUERY('<doc>{$xmltbname}</doc>' PASSING NAME as "xmltbname")
      as "newXDoc")
order by 1
fetch first 10 rows only
;
```

The outer part uses the fn:matches function which does the regex using the PASSING XMLQUERY output xmltbname "cast" as newXDoc as input. Because fn:matches is a Boolean predicate there is no need for any other predicate as it returns TRUE – row is good or FALSE – row is bad.

Regular expressions

It is a bit weird and I can recommend some more reading, firstly the excellent Rex Egg regex site where you can learn all about the joys and dangers of these beasts:

http://www.rexegg.com/

And the IBM XQuery docu that describes how the fn:matches works in detail:

https://www.ibm.com/docs/en/db2-for-zos/12?topic=expressions-regular

Both are worth a read - be especially careful about “explosive quantifiers”!

Regular expressions

CPU?

Yep, you guessed it. There is no such thing as a free lunch! The use of this method is *not* cpu light. It should only ever be used if normal LIKE, REPLACE or TRANSLATE cannot easily get the job done and if you end up coding a regex like:

```
^(?=(?!(.)\1)([^\DO:105-93+30])(?-1)(?<!\d(?<=(?![5-90-3])\d))).[^\WHY?]$
```

Then do not be surprised if your colleagues all start to hate you!

Regular expressions

Agenda

- What does “esoteric functions” mean?
- FIT/FTB
- Spatial Indexes
- Regular Expressions
- Clone tables
- Scrollable Cursors

Clone tables

Clone tables

CLONE tables arrived in DB2 9 and have not really been well used.

The idea behind them was pretty good:

- Load/Insert data over days/weeks into a table while the primary table is “in use”
- At some point in time decide to “swap” the data around really quickly
- Users now use the “new” data without realizing ***any*** change!

Sounds great doesn’t it?

Clone tables

Problems started appearing right from the get go…

- The EXCHANGE of the data is really fast *but* it requires a DBD x-lock – Ouch!
- The RTS data is separate and accurate but the Catalog Statistics are ***not*** - Pow!
- Doing any DDL Change to the Table required a ***drop*** of the CLONE – Kapow!
- Finally, ONLINE LOAD (SHRLEVEL CHANGE aka Mass INSERT) basically killed off the requirement for CLONES completely – flat line tone…

There is still a niche requirement out there and they have *not* yet been deprecated so lets dive on down into all the juicy details!

Clone tables

So what is a CLONE Table?

It is basically a duplicate table that lives in the „same“ tablespace but with a different INSTANCE.

This is the first place where people make mistakes. You read a lot about renaming the VSAM LDS.

That ***never*** happens with CLONEs. The „trick“ that IBM uses is the INSTANCE column in the SYSTABLESPACE (hence the x-lock I just mentioned).

Clone tables

Now to create a CLONE you do not use CREATE of course…

Clone tables

Now to create a **CLONE** you do not use **CREATE** of course…

Naturally, you use **ALTER**:

```
ALTER TABLE BOXWELL.TEST_BASE
  ADD CLONE RINGO.AARDVARK
;
```

This then creates a whole new set of **VSAM LDS**'s all with an **INSTANCE** value of "2" near the end of the VSAM DSN.

Clone tables

In ISPF 3.4 you would see this:

```
DB2DC1.DSNDBD.TESTDB.TESTRBAS.I0001.A001
DB2DC1.DSNDBD.TESTDB.TESTRBAS.I0002.A001
DB2DC1.DSNDBD.TESTDB.TESTTS.I0001.A001
DB2DC1.DSNDBD.TESTDB.TESTTS.I0002.A001
DB2DC1.DSNDBD.TESTDB.TEST1BZC.I0001.A001
DB2DC1.DSNDBD.TESTDB.TEST1BZC.I0002.A001
```

In SYSTABLESPACE you would see INSTANCE = 1 and column CLONE = Y telling you that this tablespace is in a clone relationship.

In the RTS you get two rows also using INSTANCE = 1 and 2.

Clone tables

Doing the swap is not another ALTER:

```
EXCHANGE DATA BETWEEN TABLE BOXWELL.TEST_BASE
                        AND RINGO.AARDVARK
;
```

At this point, the DBD gets locked so the SYSTABLESPACE can be changed from INSTANCE value 1 to 2 and you are done!

All SQL continues to work as before but are now “seeing” the data from the “other” table. For long running background refreshes of static cross-reference system of records – Super!

Clone tables

For dynamic tables, it is a disaster as the EXCHANGE has *no* idea of any updates “in flight” or “in commit” etc.

There is no log-apply phase to back-out any changes.

This was a major problem of course!

Clone tables

Utilities are severely limited on CLONEs, you can only run MODIFY RECOVERY, COPY, REORG (without inline statistics!) and QUIESCE.

Why?

Because there is only one set of catalog statistics for them. A RUNSTATS would destroy all of the data for *both* objects and the current object access paths might all go south; further, you absolutely *must* add the keyword CLONE to the utility control cards. You *cannot* rely on LISTDEF to do this for you and this is documented:

This utility processes clone data only if the CLONE keyword is specified. The use of CLONED YES on the LISTDEF statement is not sufficient.

Clone tables

Because RUNSTATS are not allowed on the CLONE, but you would probably need a RUNSTATS, you must remember to schedule a RUNSTATS as soon as possible after the exchange of data has been done.

To get rid of these, naturally DROP is not used, just another ALTER:

```
ALTER TABLE BOXWELL.TEST_BASE
      DROP CLONE
;
```

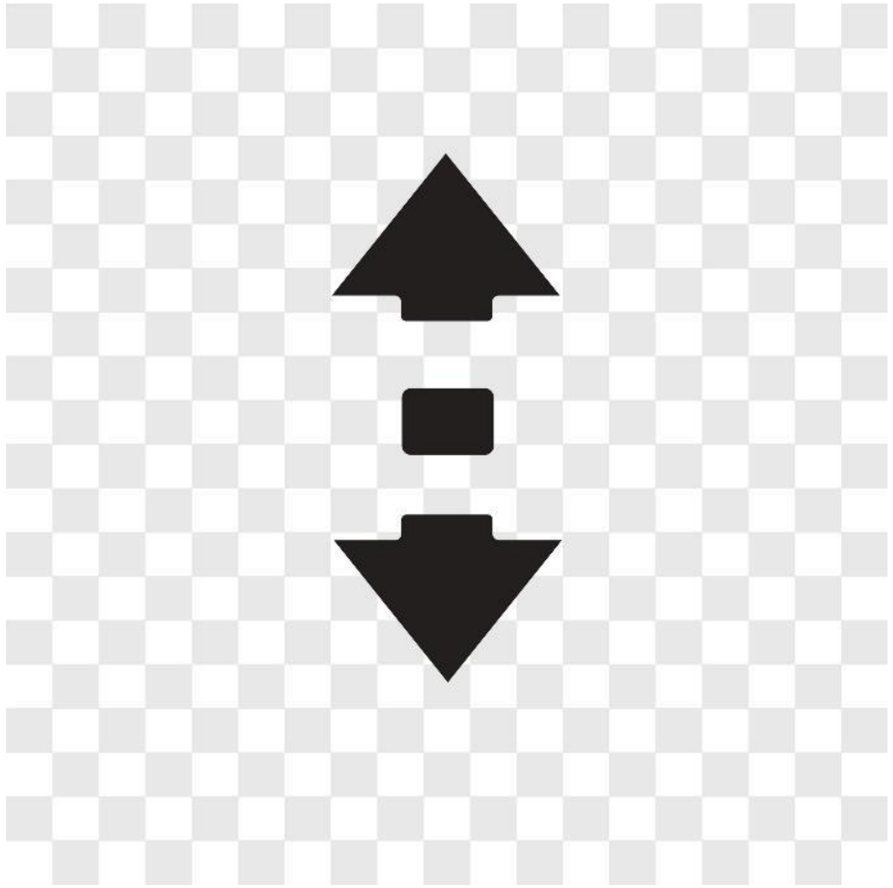
Finally, all commands got a CLONE keyword:

```
-START DATABASE(xxx) SPACENAM(yyy) CLONE
```

Agenda

- What does “esoteric functions” mean?
- FIT/FTB
- Spatial Indexes
- Regular Expressions
- Clone tables
- Scrollable Cursors

Scrollable Cursors



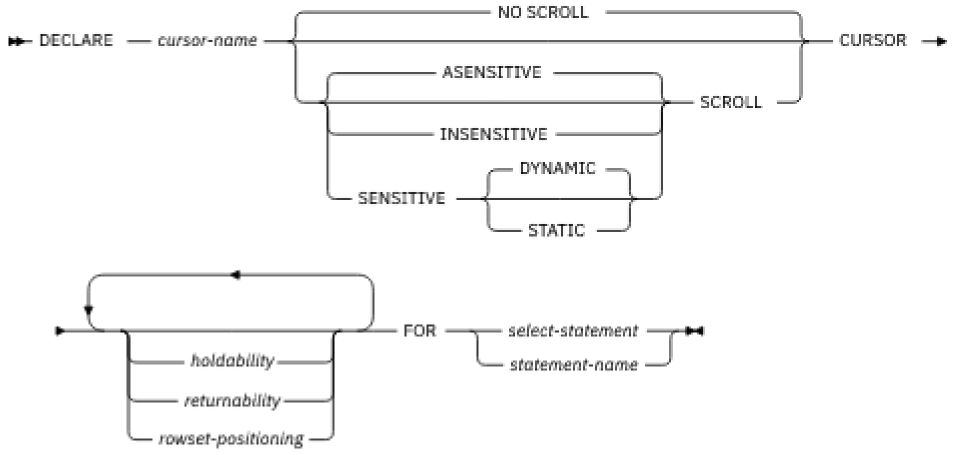
Scrollable Cursors

Quite a few people do not like these as some of them cause large amounts of CPU and I/O for materialization reasons that might/should not actually be done, but on the other hand they are great for certain processes.

As always, YMMV and the cheque is in the post…

Scrollable Cursors

Cursor definition 101:



This might not look that interesting but look how the docu reads if you start going down the road of SCROLL…

Scrollable Cursors

Cursor definition 101:

ASENSITIVE

Specifies that the cursor should be as sensitive as possible. This is the default.

A cursor that defined as ASENSITIVE will be either insensitive or sensitive dynamic; it will not be sensitive static.

The sensitivity of a cursor is a factor in the choice of access path. **Explicitly** specify the sensitivity level that you need, instead of specifying ASENSITIVE (or leaving it to be simply the default!)

Scrollable Cursors

Cursor definition 101:

<u>INSENSITIVE</u>

Specifies that the cursor does not have sensitivity to inserts, updates, or deletes that are made to the rows underlying the result table. As a result, the size of the result table, the order of the rows, and the values for each row do not change after the cursor is opened. In addition, the cursor is read-only.

<u>SENSITIVE</u>

Specifies that the cursor has sensitivity to changes that are made to the database after the result table is materialized. The cursor is always sensitive to updates and deletes that are made using the cursor (that is, positioned updates and deletes using the same cursor). When the current value of a row no longer satisfies the select-statement or statement-name, that row is no longer visible through the cursor. When a row of the result table is deleted from the underlying base table, the row is no longer visible through the cursor.

Scrollable Cursors

Cursor definition 101:

SENSITIVE DYNAMIC

Specifies that the result table of the cursor is dynamic, meaning that the size of the result table might change after the cursor is opened as rows are inserted into or deleted from the underlying table, and the order of the rows might change. Rows that are inserted, deleted, or updated by statements that are executed by the same application process as the cursor are visible to the cursor immediately. Rows that are inserted, deleted, or updated by statements that are executed by other application processes are visible only after the statements are committed. If a column for an ORDER BY clause is updated via a cursor or any means outside the process, the next FETCH statement behaves as if the updated row was deleted and re-inserted into the result table at its correct location. At the time of a positioned update, the cursor is positioned before the next row of the original location and there is no current row, making the row appear to have moved.

Scrollable Cursors

Cursor definition 101:

<u>SENSITIVE STATIC</u>

Specifies that the size of the result table and the order of the rows do not change after the cursor is opened. Rows inserted into the underlying table are not added to the result table regardless of how the rows are inserted. Rows in the result table do not move if columns in the ORDER BY clause are updated in rows that have already been materialized. Positioned updates and deletes are allowed if the result table is updatable. The SELECT statement of a cursor that is defined as SENSITIVE STATIC cannot contain an SQL data change statement.

A STATIC cursor has visibility to changes made by *this* cursor using positioned updates or deletes. Committed changes made outside this cursor are visible with the SENSITIVE option of the FETCH statement. A FETCH SENSITIVE can result in a *hole* in the result table (that is, a difference between the result table and its underlying base table). This leads to SQLWARNING +222 by FETCH.

Scrollable Cursors

That's a *lot* of text…So here are some Rules of Thumb:

- Declare scrollable cursors as SENSITIVE only if you need to see the latest data.
- If you do not need to see updates that are made by other cursors or application processes, using a cursor that you declare as INSENSITIVE requires less processing by DB2.
- If you need to see only some of the latest updates, and you do not need to see the results of insert operations, declare scrollable cursors as SENSITIVE STATIC.
- If you need to see all of the latest updates and inserts, declare scrollable cursors as SENSITIVE DYNAMIC.
- To ensure maximum concurrency when you use a scrollable cursor for positioned update and delete operations, specify ISOLATION(CS) and CURRENTDATA(NO) when you bind packages that contain updatable scrollable cursors.

Scrollable Cursors

That's a *lot* of text…So here are some Rules of Thumb:

In a work file database, create 32Kb table spaces that are large enough for processing your scrollable cursors. Db2 uses declared temporary tables for processing the following types of scrollable cursors:

- SENSITIVE STATIC SCROLL
- INSENSITIVE SCROLL
- ASENSITIVE SCROLL, if the cursor sensitivity is INSENSITIVE. (A cursor that meets the criteria for a read-only cursor has an effective sensitivity of INSENSITIVE)

Scrollable Cursors

Cursor definition 101:

DECLARE sensitivity	FETCH INSENSITIVE	FETCH SENSITIVE
INSENSITIVE	No changes to the underlying table are visible in the result table. Positioned UPDATE and DELETE statements using the cursor are not allowed.	Not valid.
SENSITIVE STATIC	Only positioned updates and deletes that are made by the cursor are visible in the result table.	All updates and deletes are visible in the result table. Inserts made by other processes are not visible in the result table.
SENSITIVE DYNAMIC	Not valid.	All committed changes are visible in the result table, including updates, deletes, inserts, and changes in the order of the rows.

Scrollable Cursors

Cursor definition 101:

Declared cursor type	Cursor is updatable or read-only?	Changes by the cursor are visible in the result table?	Changes by other cursors or processes are visible to the result table?
NO SCROLL (result table is materialized)	Read-only	Not applicable	No
NO SCROLL (result table is not materialized)	Updatable	Yes	Yes
INSENSITIVE SCROLL	Read-only	Not applicable	No
SENSITIVE STATIC SCROLL	Updatable	Yes	Depends on the explicitly or implicitly specified sensitivity in the FETCH clause
SENSITIVE DYNAMIC SCROLL	Updatable	Yes	Yes

Scrollable Cursors

Fetch definition 101:

FETCH	
Keyword in FETCH statement	Cursor position when FETCH is executed[1.a]
BEFORE	Before the first row
FIRST or ABSOLUTE +1	On the first row
LAST or ABSOLUTE -1	On the last row
AFTER	After the last row
ABSOLUTE	On an absolute row number, from before the first row forward or from after the last row backward
RELATIVE	On the row that is forward or backward a relative number of rows from the current row
CURRENT	On the current row
PRIOR or RELATIVE -1	On the previous row
NEXT	On the next row (default)

Scrollable Cursors

Pain Points :

While sensitive static scrollable cursors are open against a table, Db2 disallows reuse of space in that table space to prevent the scrollable cursor from fetching newly inserted rows that were not in the original result set.

Although this is normal, it can result in a seemingly false out-of-space indication. The problem can be more noticeable in a data sharing environment with transactions that access LOBs.

Scrollable Cursors

Pain Points :

In addition to the space reuse issue, the use of a sensitive static scrollable cursor in a data sharing environment might also result in lock contention on INSERT statements if the inserted objects are in the same buffer pool.

This situation applies regardless of whether the objects have sensitive static scrollable cursors, and regardless of whether the objects contain any LOB columns.

You can minimize this problem by isolating objects that have a large volume of insert activity so that they are in a dedicated buffer pool within the data sharing environment.

Scrollable Cursors

And a final Pain Point that is squirreled away in the docu that can easily catch you out:

Db2 does not use an expression-based index for queries that use sensitive static cursors.

Questions & Answers