Know Your Onions When it Comes to Db2 Indexes

Randy Bright
BMC Software, Inc.

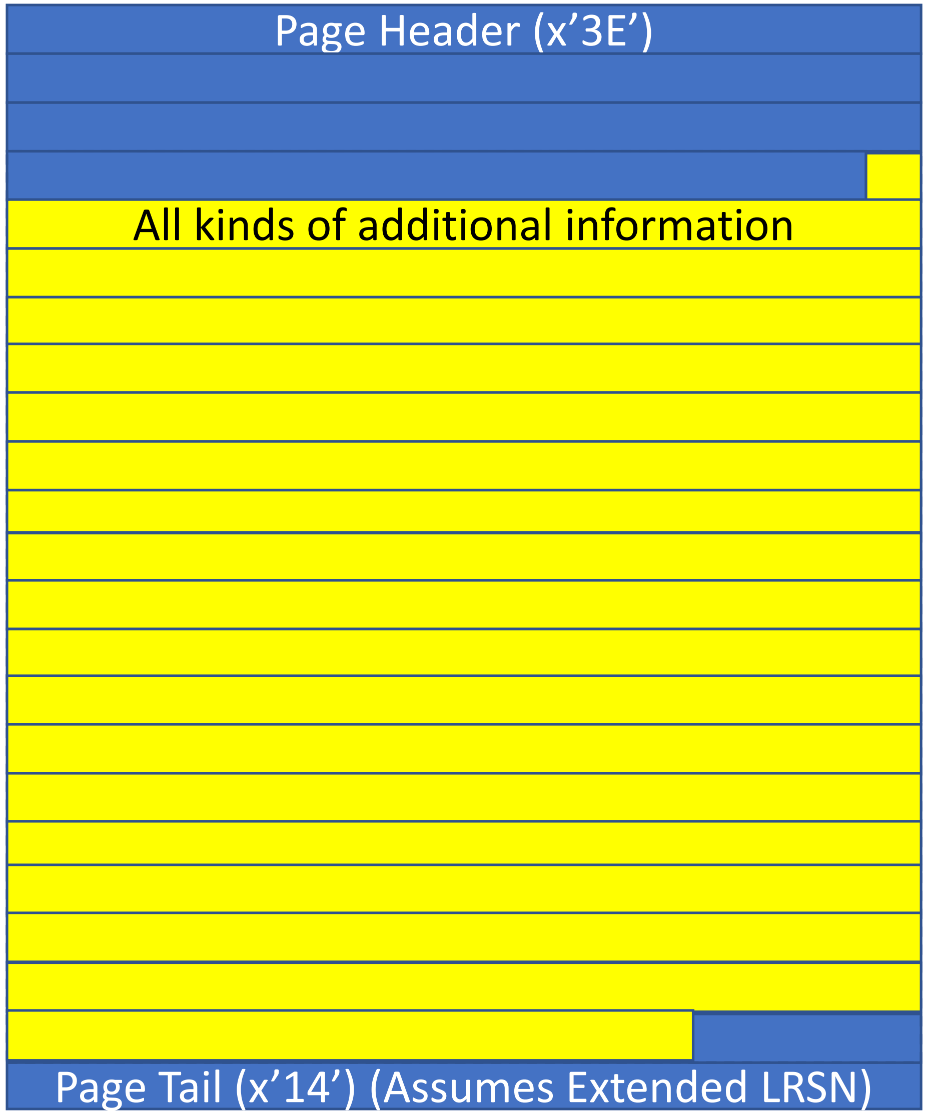
November 7, 2018
Session IJ – 09:30-10:30

Agenda

- Fundamentals
  - Header Page, Space Map, Directory Page, Non-Leaf Page, Leaf Page
- Page Header, Header Page, Space Map
  - Bits and Bytes
- Leaf Pages
  - Bits and Bytes
  - What Happens with SQL
- Non-Leaf Pages
  - Bits and Bytes
  - Their Purpose

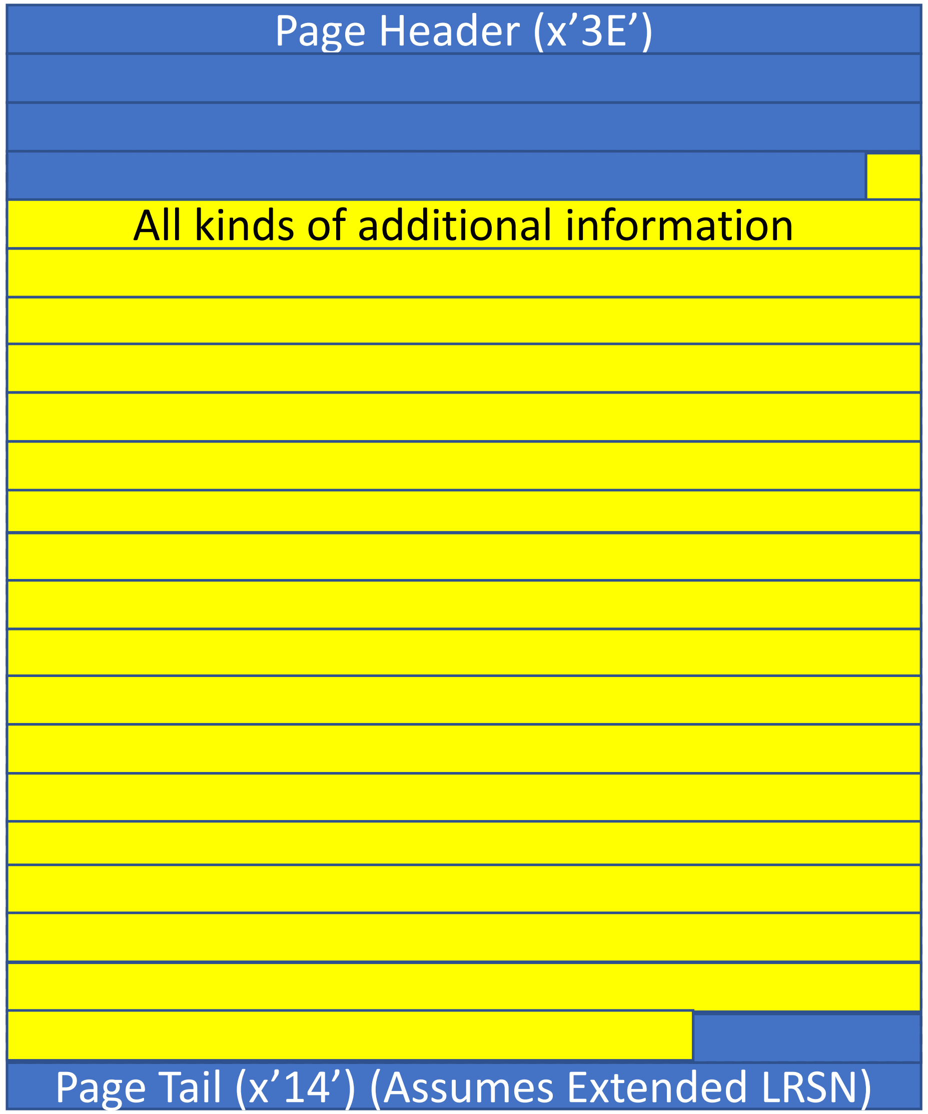
Header Pages, Page Headers, and Space Maps

The Header page


Page Header (x’3E’)
All kinds of additional information
Page Tail (x’14’) (Assumes Extended LRSN)

- There is only one Header Page.
- Header page is always page zero.
- The first x’3E’ bytes are the Header Page Page Header.
  - Remember, all pages contain a Page Header, even the Header Page.
  - Beyond that, this page has tons of information about this object.
  - I’m not going to take the time to list every bit of information in the Header Page and where it is, but it includes things like…

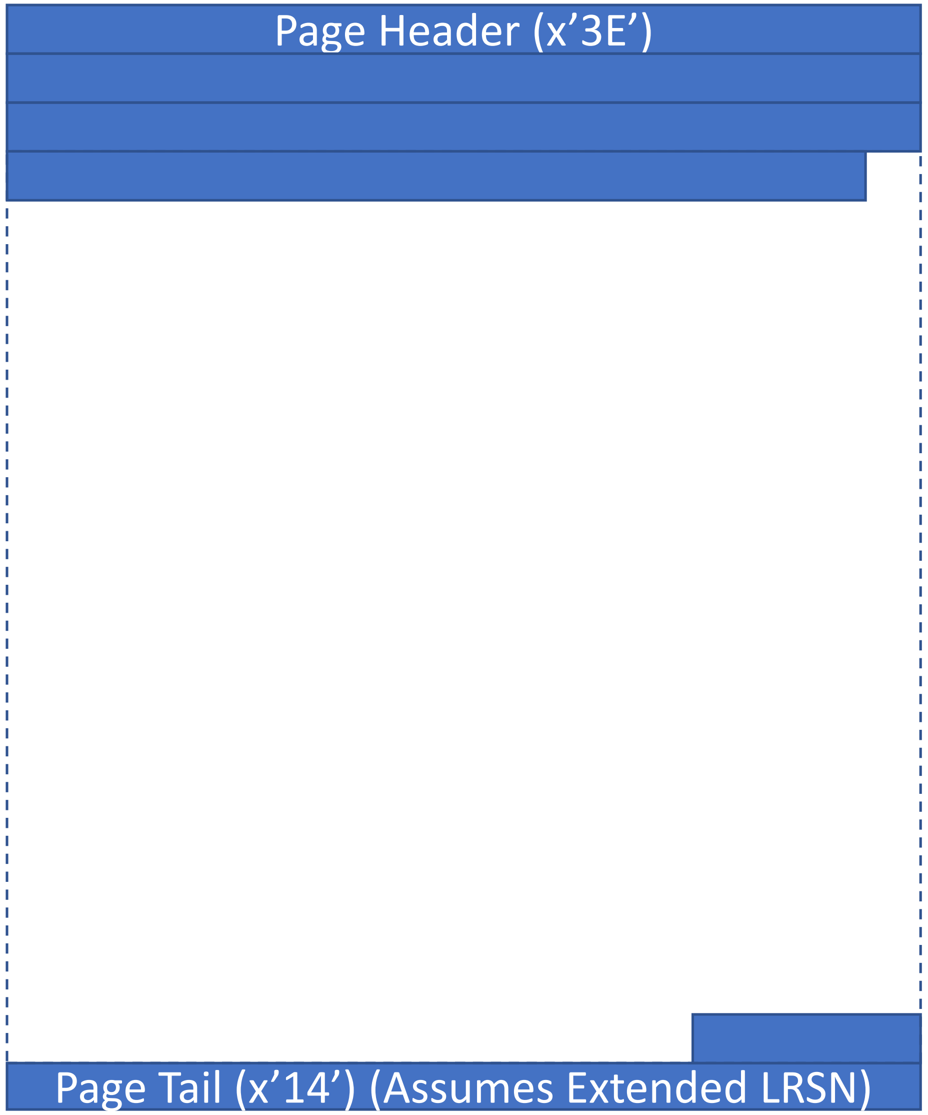
The Header page



Page Header (x'3E')
All kinds of additional information
Page Tail (x'14') (Assumes Extended LRSN)

- The STOGROUP name.
- The VCAT name.
- Recovery information.
- Flags that indicate:
  - It is partitioned or not.
  - It contains variable length keys or not.
  - It has unique keys or not.
  - It is a DPSI or not.
  - It is compressed or not.
  - Keys contain included columns or not.
  - And much more in the Diagnosis Guide.

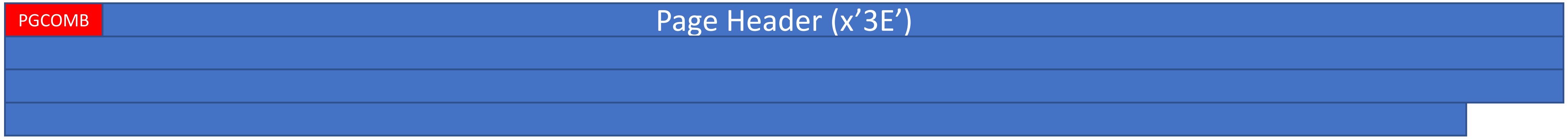
The Page Header (and Page Tail)



Not to be confused with the Header Page.

- Every page has a Page Header and Page Tail
  - For this presentation I will concentrate on the Extended LRSN Page Tail.
- Both the Page Header and Page Tail contain pretty much the same information regardless of page type.
  - With IBM there are always exceptions.
  - One exception is the Directory Page.

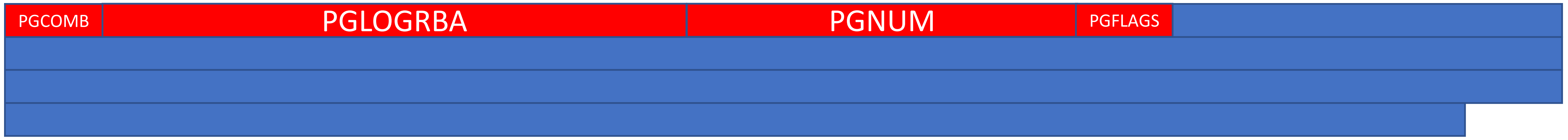
A much closer look at the Page Header.



Information contained in the Page Header:

- +x’00’ – PGCOMB, is one byte of flags:
  - b’1... ....’ – This bit says the page was read with an I/O error.
  - b’.1.. ....’ – Page was modified by the REPAIR Utility.
  - b’...1 ....’ – Last byte of the page is x’52’ (otherwise x’42’).
    Or x’D5’ or x’C5’ if not Extended LOGRBA/LRSN
  - b’.... ...1’ – Page was copied by DSN1COPY.

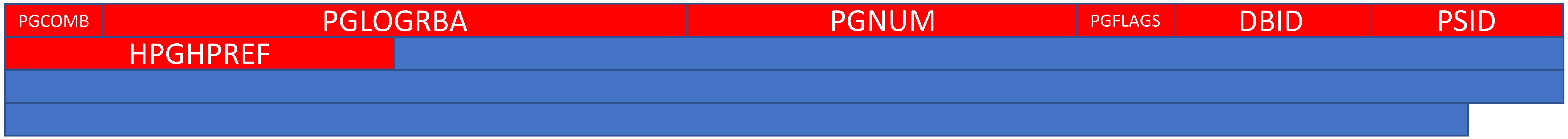
A much closer look at the Page Header.



Information contained in the Page Header:

- +x’01’ – PGLOGRBA, is the 6-byte LOG RBA of the last update.
  - Not used when the LOGRBA/LRSN is Extended format (more later).
- +x’07’ – PGNUM, is the 4-byte page number.
  - Page numbers are a whole wealth of information by themselves.
- +x’0B’ – PGFLAGS, another byte of flags:
  - b’1... ....’ – Means this page contains inconsistent data.
  - b’..11 11..’ – A combination of these bits indicates this is a Header Page
    - Header Page in a segmented page set, non-segmented page set, index page set, etc.

A much closer look at the Page Header.

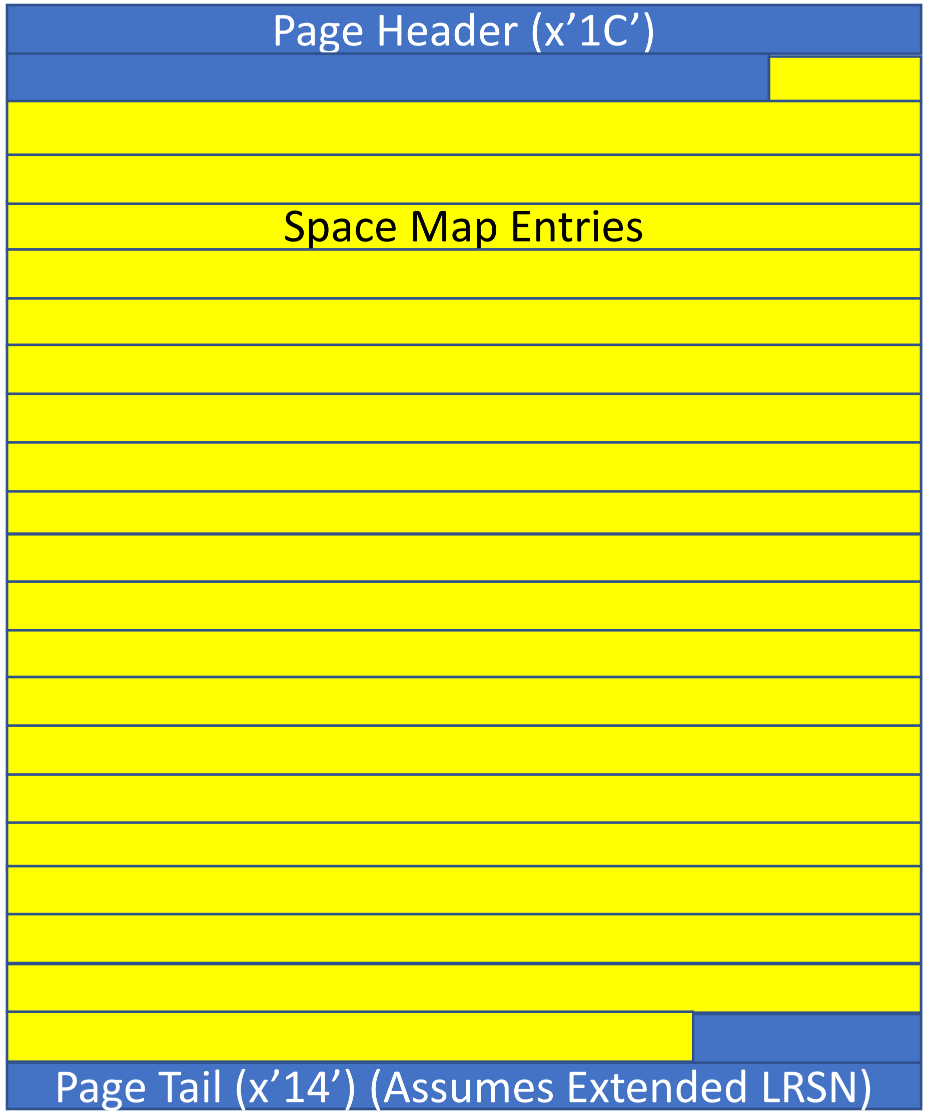


Information contained in the Page Header:

- +x’0C’ – HPGDBID, is the 2-byte DBID of this object.
- +x’0E’ – HPGPSID, is the 2-byte OBID or ISOBID.
- +x’10’ – HPGHPREF, is the 4-byte highest formatted page number.

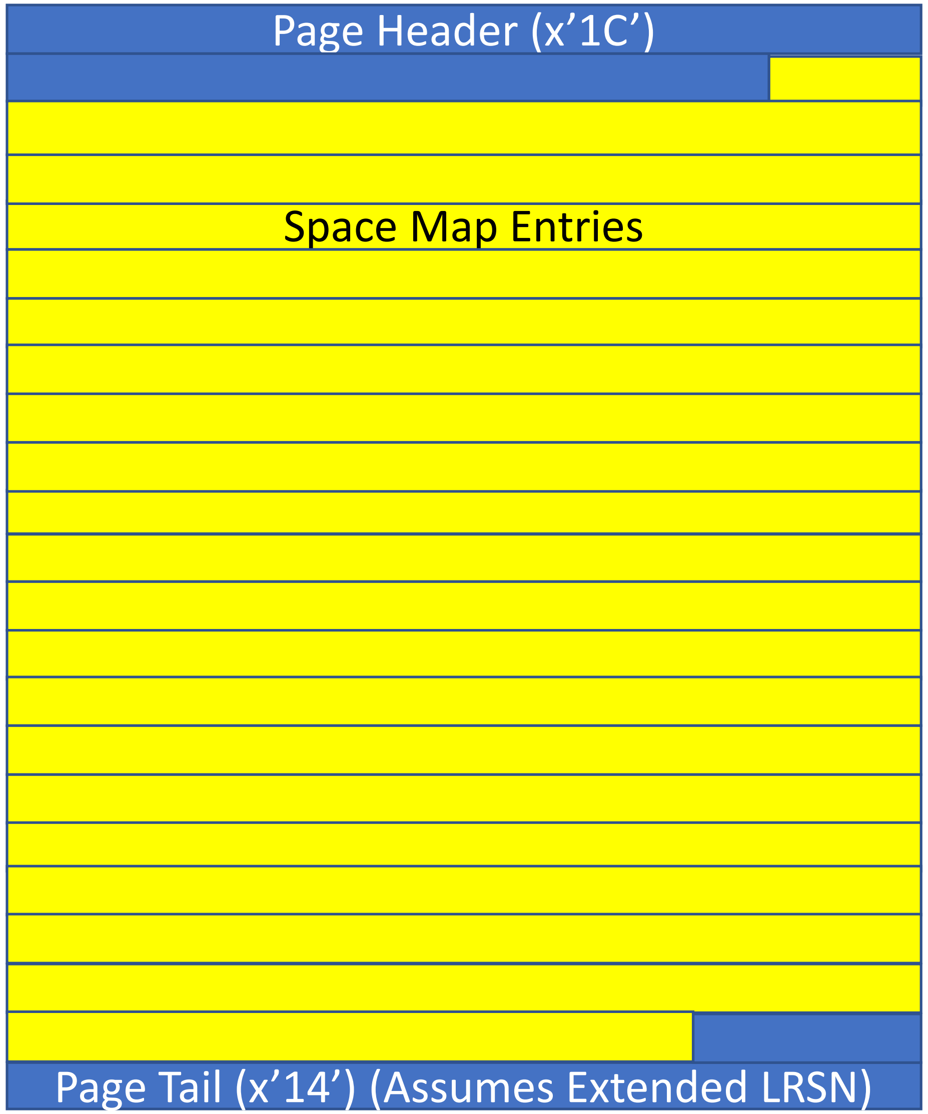
That is enough for now. More Page Header specifics as we talk about contents of pages.

The Space Map



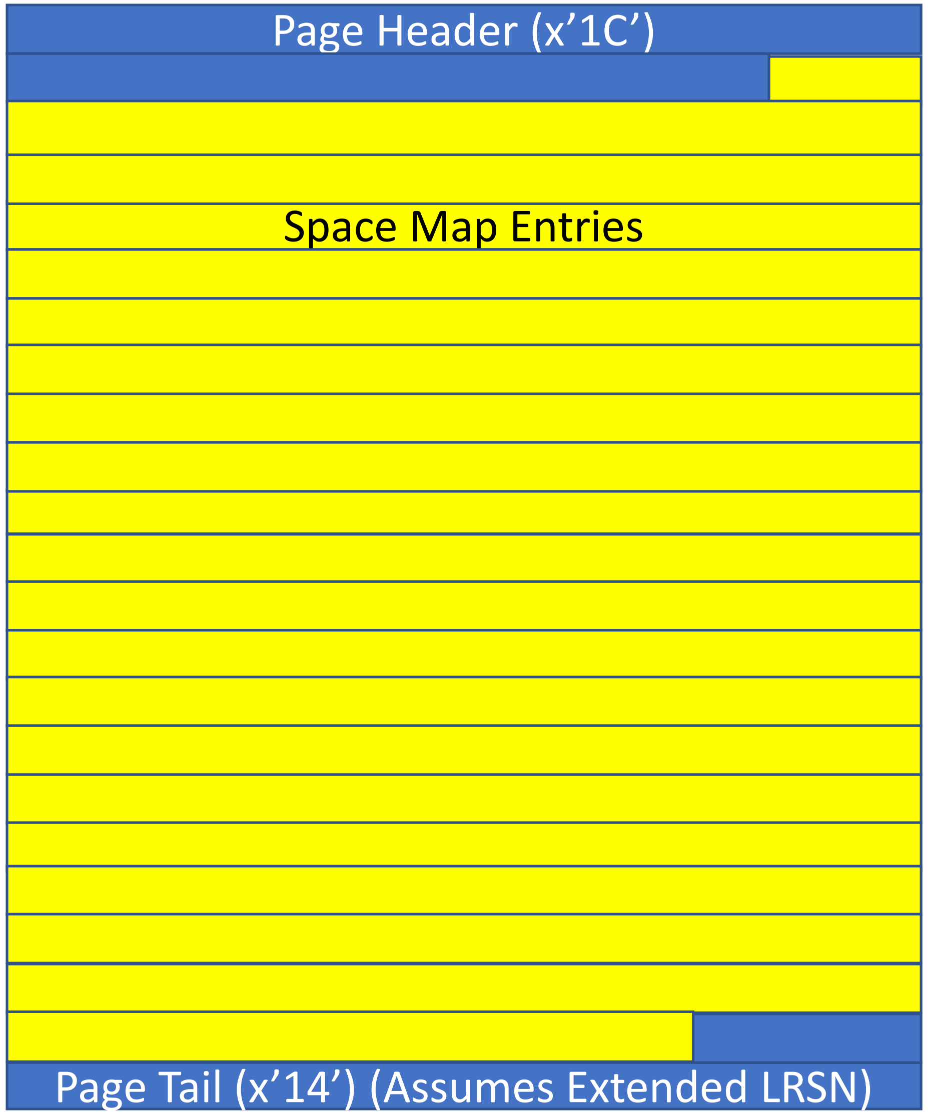
- I say “the” Space Map, but there can be more than one.
- The first Space Map is always page number 1 (the second page).
- The Space Map Page Header is shorter than the normal Page Header for other pages.
  - Additional information is not needed.
  - More room for Space Map Entries.

The Space Map



- At offset x’14’ is the number of pages covered by this Space Map.
  - Multiply the number of this Space Map by this number and add one to get the next Space Map page number.
  - This works for every Space Map.
    - Well, almost.
    - For index Space Maps it is pretty consistent.

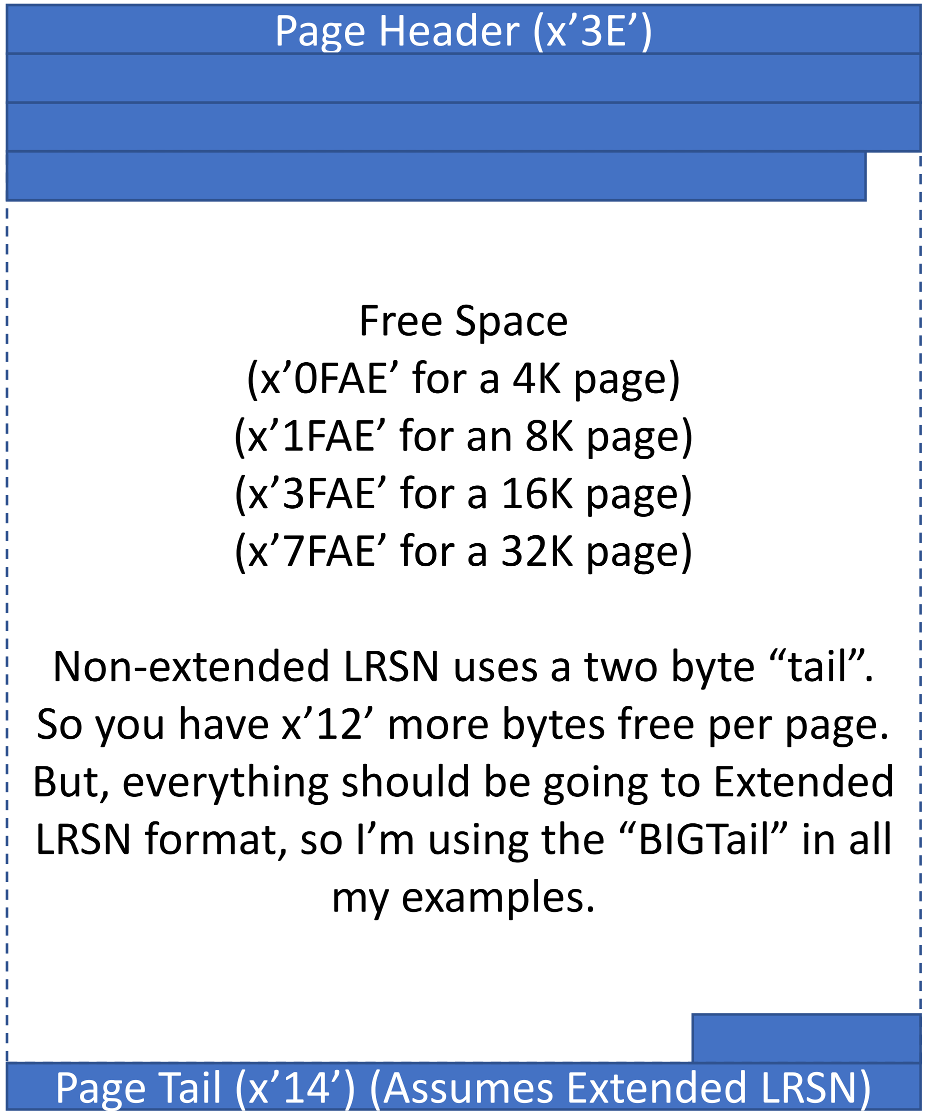
The Space Map



- The space map contains:
- A half-byte of flags for each page addressed.
- These half-byte page entries tell:
  - If the page is used.
  - If the page is empty.
  - If the page is possibly uncommitted.

Leaf Pages

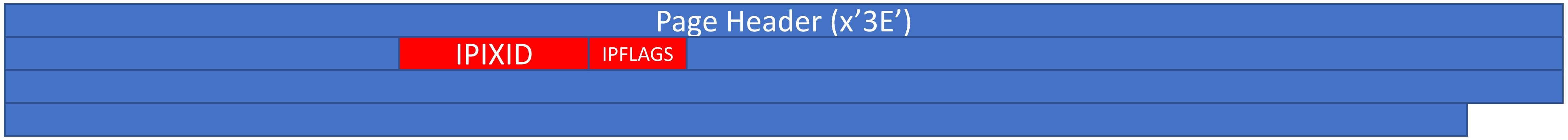
What does an “empty” index page look like?


Page Header (x’3E’)
Free Space
(x’0FAE’ for a 4K page)
(x’1FAE’ for an 8K page)
(x’3FAE’ for a 16K page)
(x’7FAE’ for a 32K page)
Non-extended LRSN uses a two byte “tail”.
So you have x’12’ more bytes free per page.
But, everything should be going to Extended LRSN format, so I’m using the “BIGTail” in all my examples.
Page Tail (x’14’) (Assumes Extended LRSN)

This is what an empty page looks like.

- It has a Page Header.
  - Page Header for an index leaf page is always x’3E’ in size (62 bytes).
- It has a Page Tail.
  - Page Tail for an index leaf page is always x’14’ in size (20 bytes).
- In an empty 4K page, free space is always x’0FAE’ in length (4014 bytes).
- Other page sizes still have the same size Page Header and Page Tail, leaving more space for keys.

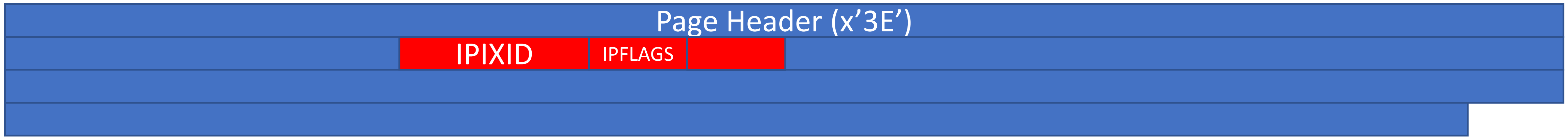
Now More Page Header Information.



More information contained in the Index Page Header:

- +x’14’ – IPIXID, is the 2-byte OBID of this index.
- +x’16’ – IPFLAGS, a byte of flags.
  - b’1... ....’ – Means this is the Root Page.
  - b’.1.. ....’ – Means this is a Leaf Page.
  - b’..1. ....’ – Means this is a parent of a leaf page.
  - b’...1 ....’ – Means this index is defined UNIQUE WHERE NOT NULL

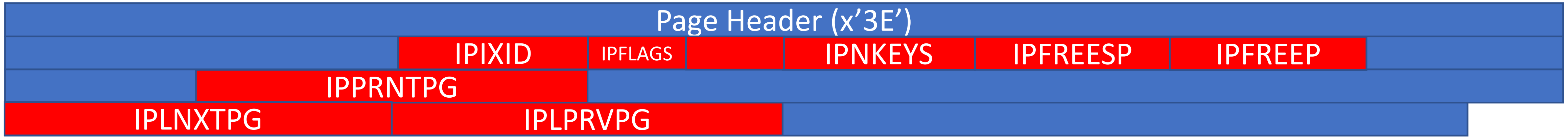
Now More Page Header Information.



More information contained in the Page Header:

- +x'16' – IPFLAGS, a byte of flags (continued).
  - b'.... 1...' – Means this index is defined non UNIQUE.
  - b'.... .1..' – Means this index is defined NOT PADDED.
  - b'.... ..1.' – Means this index has had a structure modification.
  - b'.... ...1' – Means all data on this index has been committed.
- +x'17' – Another byte of flags, but I'm not going into detail.

Now More Page Header Information.



More information contained in the Page Header:

- +x'18' – IPNKEYS, is a 2-byte count of keys on this page.
- +x'1A' – IPFREESP, is a 2-byte number of free bytes on this page.
- +x'1C' – IPFREEP, is a 2-byte offset to contiguous free space.
- [some other stuff]
- +x'24' – IPPRNTPG, four byte page number of the parent of this page.
- [more stuff]
- +x'30' – IPLNXTPG, four byte page number of the next leaf page.
- +x'34' – IPLPRVPG, four byte page number of the previous leaf page.

Now is a good time for more information…

What is meant by “next”, “previous”, and “parent”?

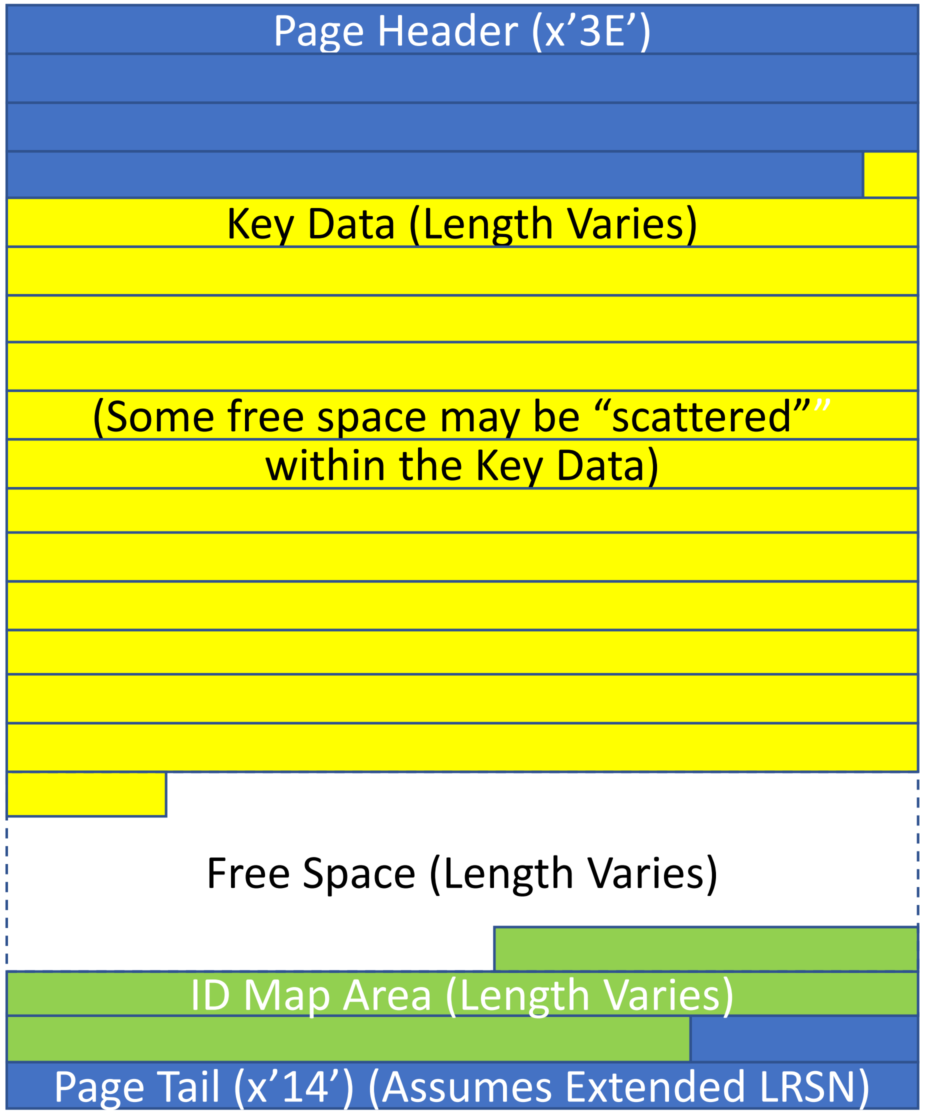
- Leaf Pages, and Non-Leaf Pages, for that matter, are “linked” in key sequence.
- All of the keys on a page must be in sequential order.
- This means if you find the “logical” first Leaf Page, read the keys from it in physical sequence, go to the “next” page and read its keys in physical sequence, etc., you will get all the keys in the index in sequence.
- No need to sort.

Now is a good time for more information…

What is meant by “next”, “previous”, and “parent”?

- Likewise, if you find the logical last page in an index, read its keys backwards, and go to the previous page and do the same, etc. you will get all the keys in reverse sequence.
- Done less often, but can still be done.
- “Parent” refers to the Non-Leaf Page that points to this Leaf Page. We will explain more about Non-Leaf Pages later.
- For now, back to the page contents.

What does a “typical” page look like?

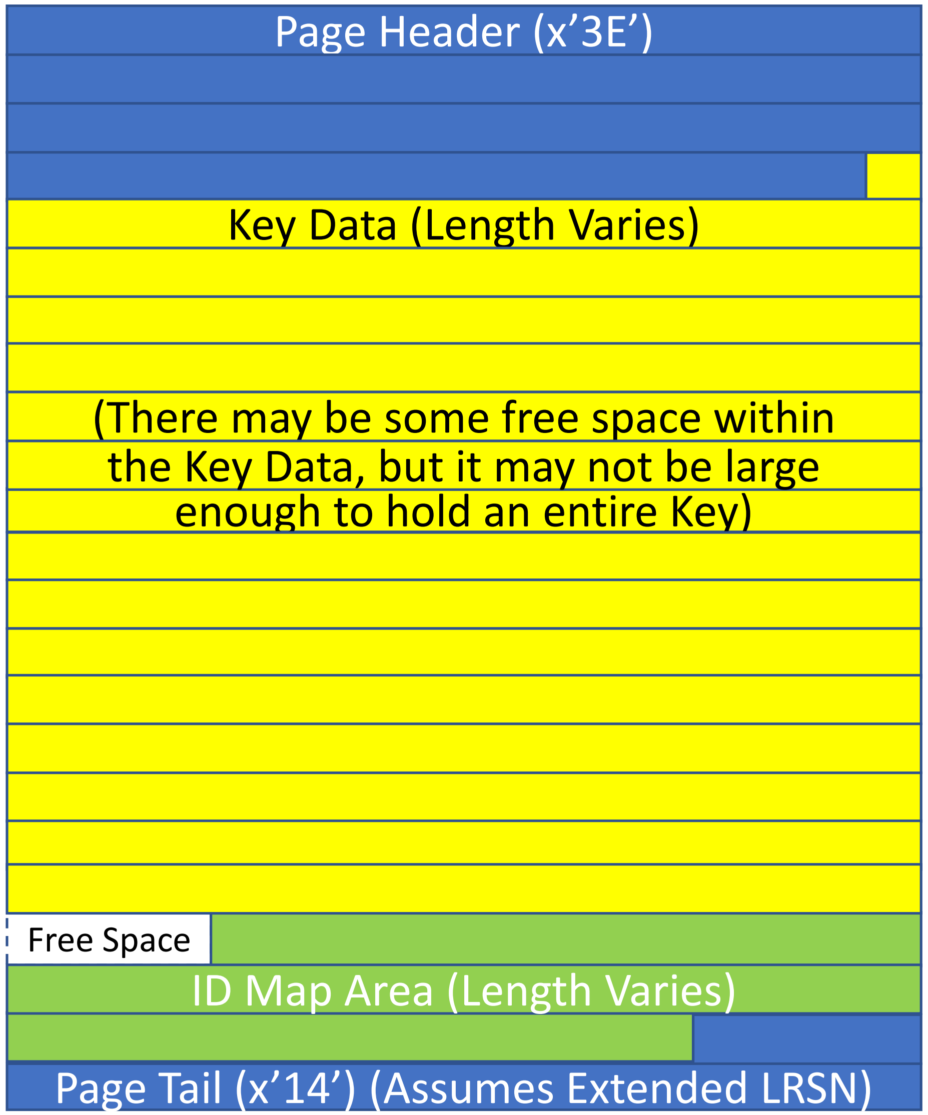


Page Header (x’3E’)
Key Data (Length Varies)
(Some free space may be “scattered”” within the Key Data)
Free Space (Length Varies)
ID Map Area (Length Varies)
Page Tail (x’14’) (Assumes Extended LRSN)

This could be a “typical” page.

- The Page Header and Page Tail are still present (in blue).
- Keys (in yellow) begin immediately after the Page Header.
- At the bottom of the page is the ID Map (in green).
- Between the Keys and the ID Map is free space where new Keys can be inserted.
- Some free space may be interspersed within the Key area.

What does a “full” page look like?



Page Header (x’3E’)
Key Data (Length Varies)
(There may be some free space within the Key Data, but it may not be large enough to hold an entire Key)
Free Space
ID Map Area (Length Varies)
Page Tail (x’14’) (Assumes Extended LRSN)

This is a full page.

- Notice the Keys (in yellow) and the ID Map (in green) have grown to almost no free space.
- Typically there is a small amount of free space, but not enough to hold a Key.
- So the page is “logically” full.

This boring information does lead somewhere…

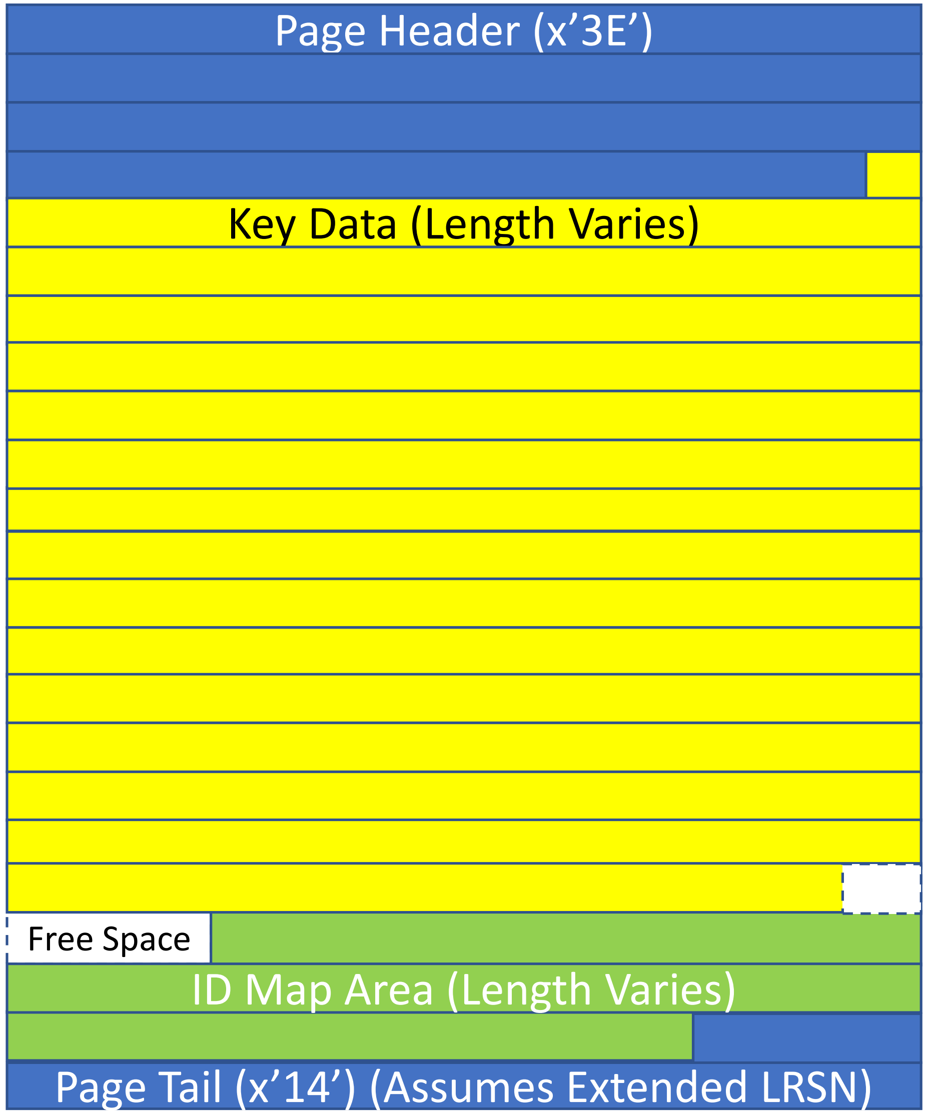
This is leading to probably the most important piece of information I will provide today.

- REORG YOUR INDEXES!
- Now we will get into why…

Page Splits

Page Splits

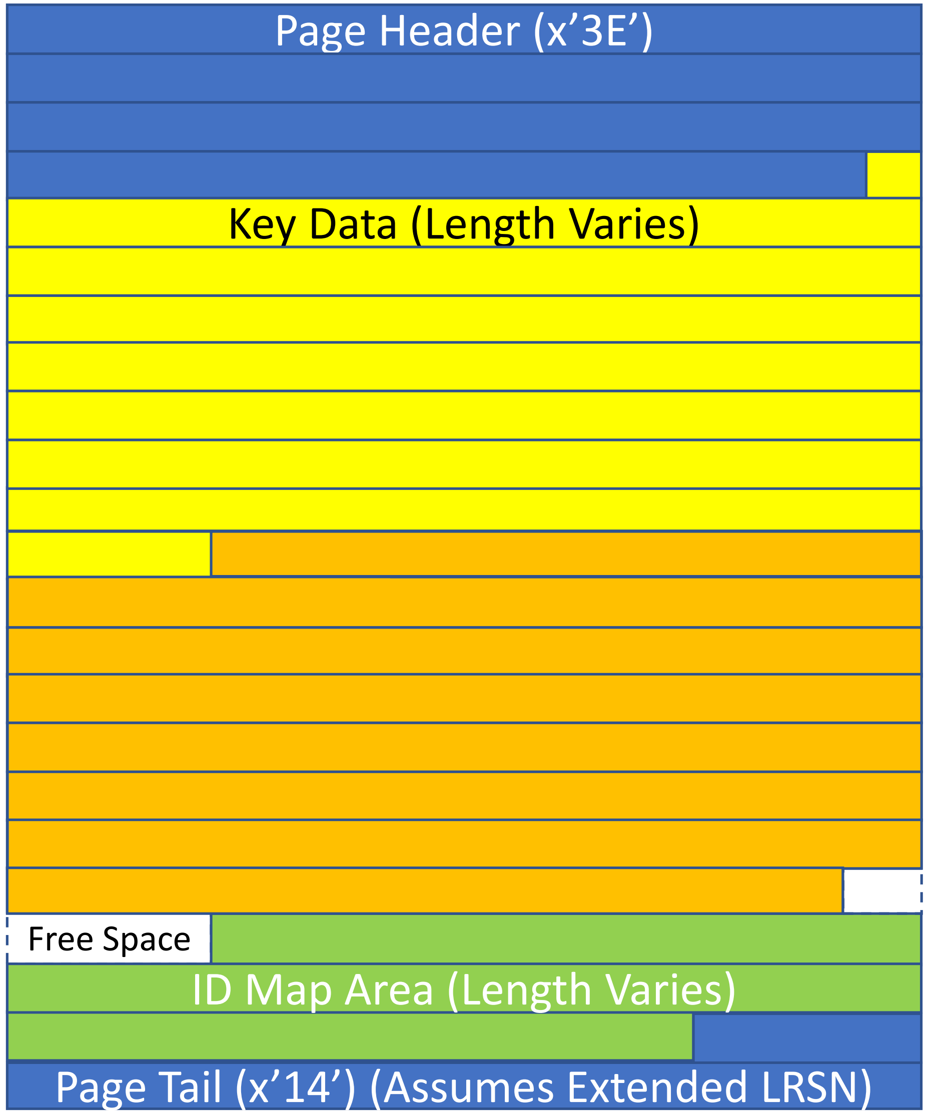
So what happens when you INSERT a key?


Page Header (x’3E’)
Key Data (Length Varies)
Free Space
ID Map Area (Length Varies)
Page Tail (x’14’) (Assumes Extended LRSN)

Good question!

- Remember, the Keys on a page must be in order and the pages must be in order by Key.
- So how do you fit a new Key that is bigger than the free space on the page?
  - First Db2 will “gather” all the scattered free space from the Key area.
  - If there is still not enough space, then the page is “split”.

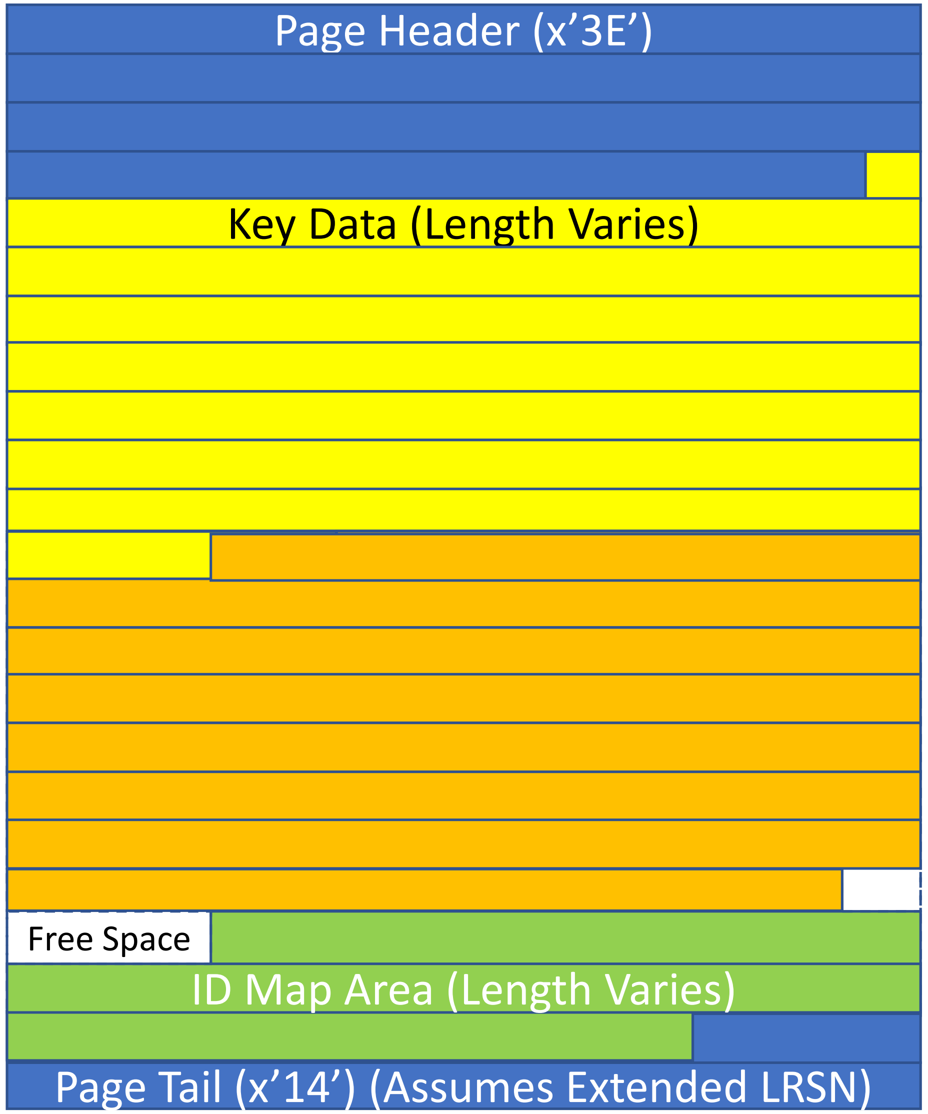
So what is a page split?


Page Header (x'3E')
Key Data (Length Varies)
Free Space
ID Map Area (Length Varies)
Page Tail (x'14') (Assumes Extended LRSN)

When no new Keys will fit on a page, Db2 "splits" that page.

- First Db2 goes to the "physical center" of the Key area.
- For a 4K page, that is offset x'0815'.
- If the physical center is within a key (and it probably is), then Db2 finds the "logical center" of the page.
  - The logical center is at the end of the key if the physical center is beyond the center of the key.
  - Otherwise it is the beginning of the key.

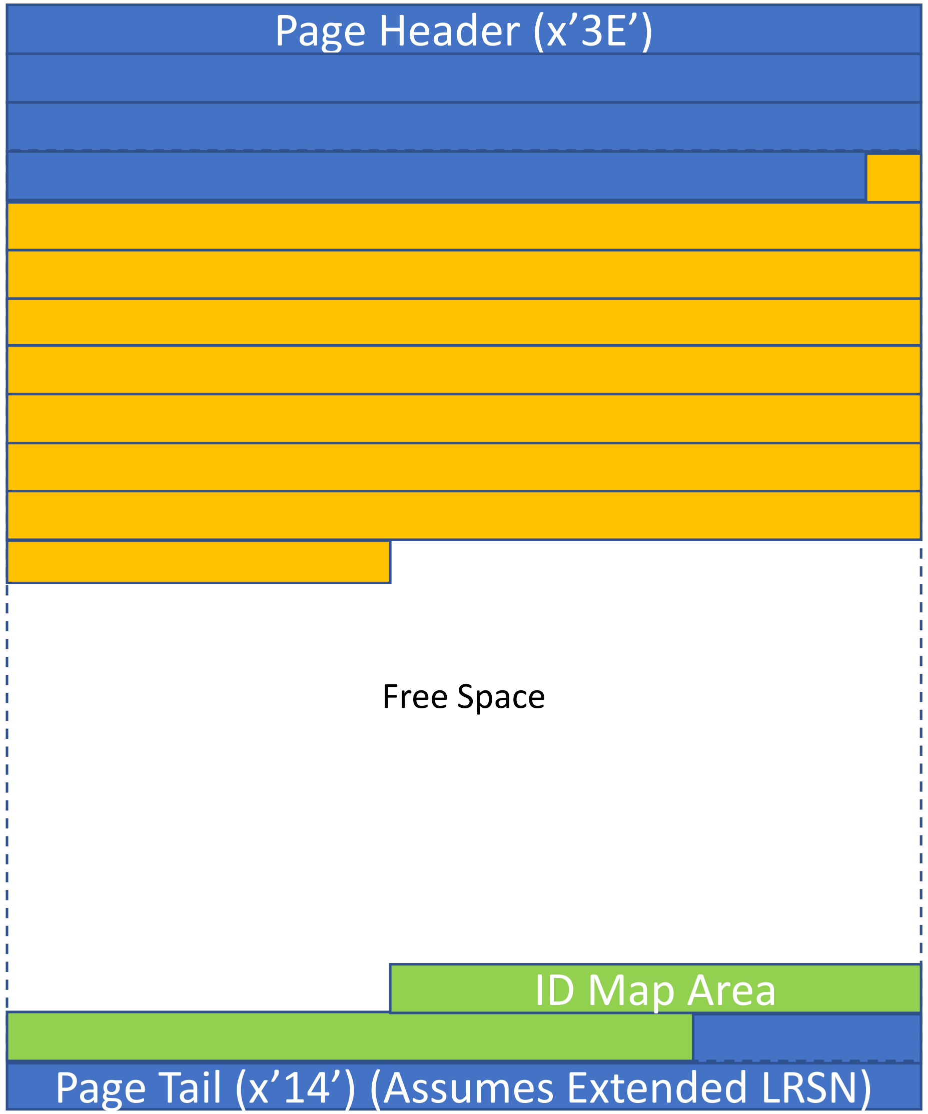
What is a “page split”?


Page Header (x’3E’)
Key Data (Length Varies)
Free Space
ID Map Area (Length Varies)
Page Tail (x’14’) (Assumes Extended LRSN)

Now that we have the center of the Key Data, the split occurs.

- The “bottom” half of the Key Area is removed.
- And free space now replaces the area where the keys once were.
- Oh No! What happened to all those keys?
- Not to worry, they aren’t gone.

What is a “page split”?

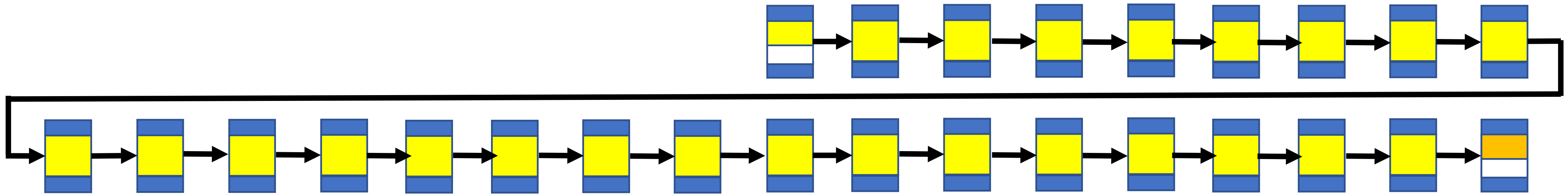


A new page is created.

- All the keys from the bottom half of the split page are moved here.
  - If the page was split “before” the inserted key, it is on this new page.
  - If the split was “after” the new key, it will be on the page that was split.
- Free space is now approximately half the new page.

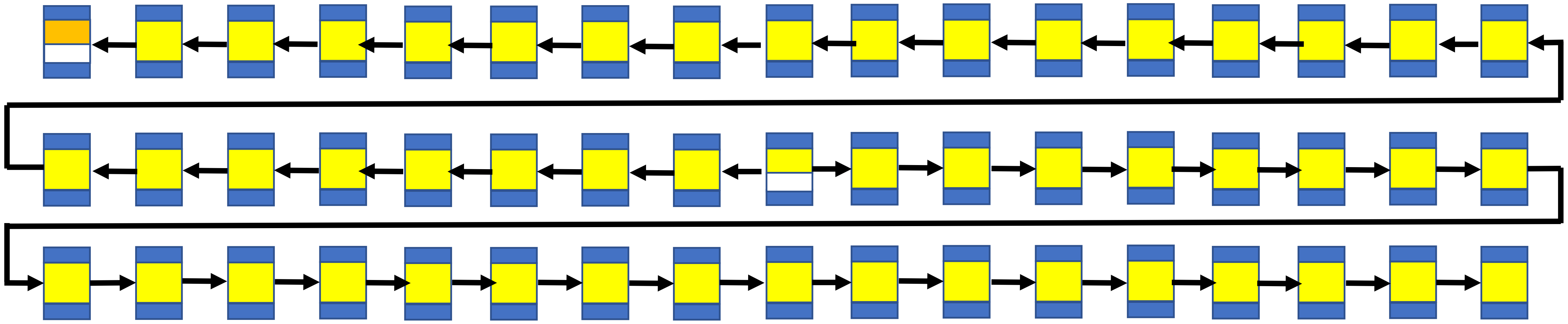
But where does this new page go?

- Starting with the page that was split, Db2 looks “forward” up to 127 pages for an empty page.
  - If it finds one, it puts the new page there.



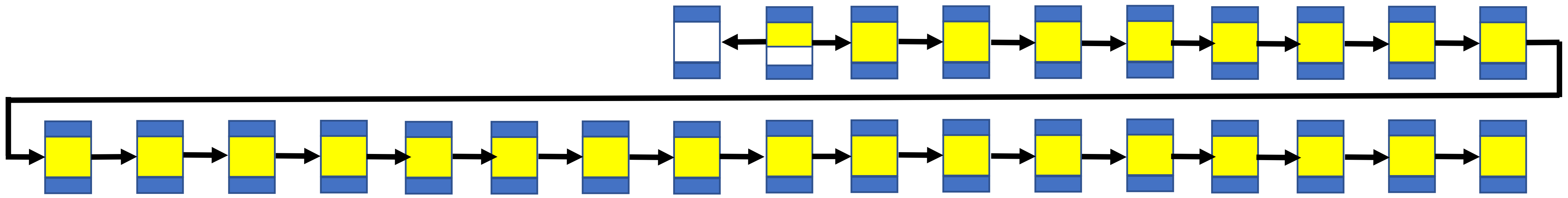
But where does this new page go?

- Starting with the page that was split, Db2 looks "forward" up to 127 pages for an empty page.
  - If it finds one, it puts the new page there.
- If not found looking forward, it looks "backward" up to 128 pages.
  - If it finds one, it puts the new page there.



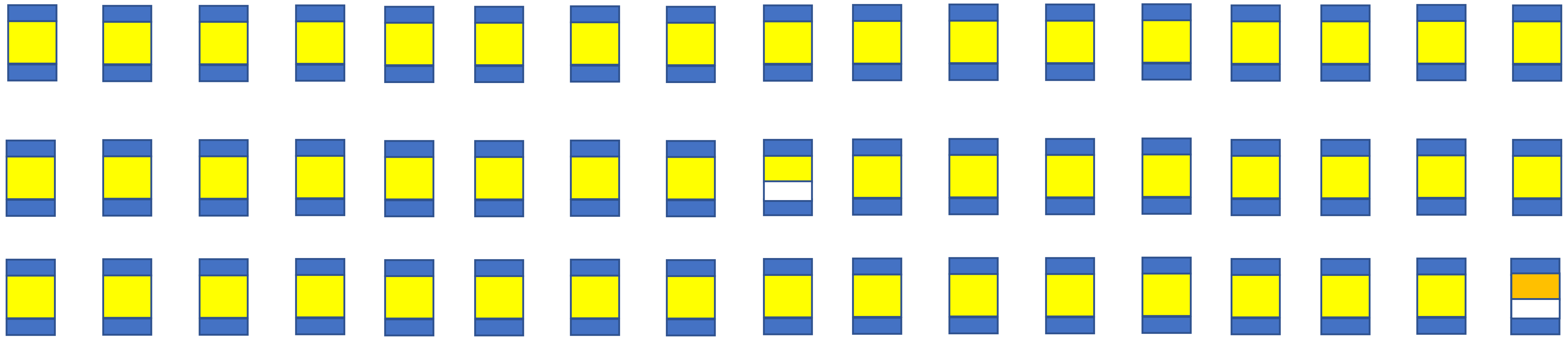
But where does this new page go?

- The odd part is, it will always look forward before it looks backward.
  - An empty page might be right behind you, but it is not seen unless the 127 pages ***after*** the split page are searched and no empty page is found.
- Now we can talk about the major reason to REORG an index.
  - REORGLEAFNEAR and REORGLEAFFAR in SYSIBM.SYSINDEXSPACESTATS.



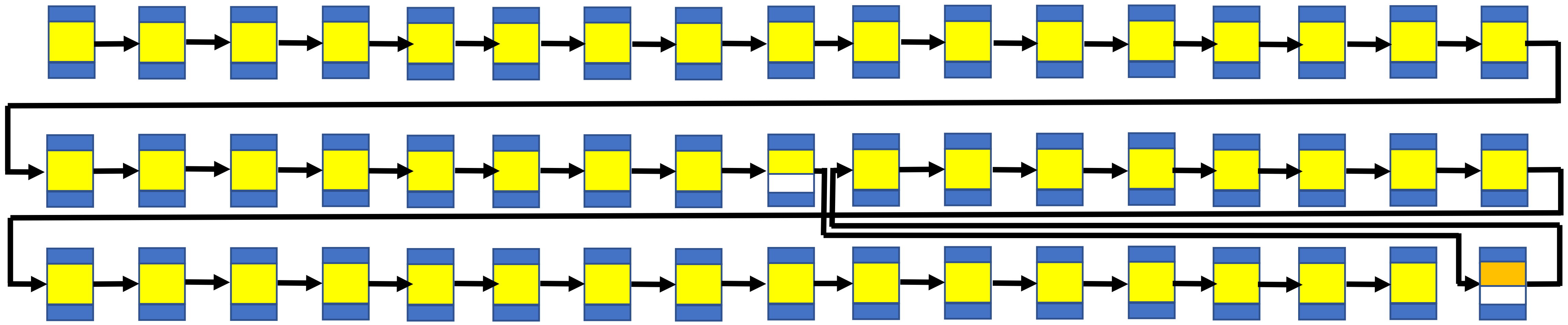
So why should I worry about this?

- Remember Leaf Pages must be maintained in logical sequential order.
  - That means the IPLNXTPG must point to the page with the next higher key.
  - And the IPLPRVPG must point to the page with the next lower key.
- So let’s go back to this scenario.
  - After we have inserted a key, split a page, and used an empty page for the lower half of the split page.



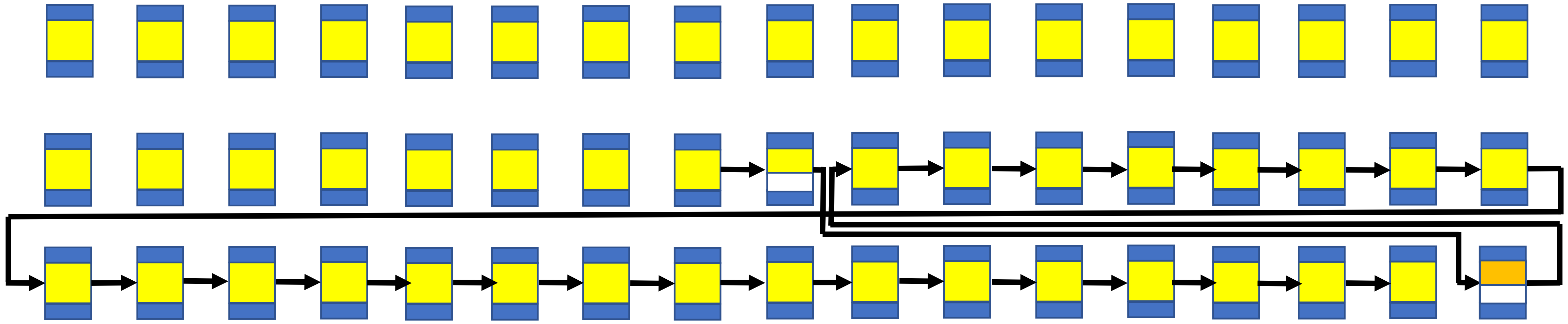
So why should I worry about this?

- Because now this is what the IPLNXTPG pointers look like.



So why should I worry about this?

- Guess what happens if you INSERT a key that needs to go into this page and there is no room?
- Yep, this is how the forward pointer search goes.
- Now, multiply that times 1,000 page splits. 10,000 page splits, 100,000 page splits.



So why should I worry about this?

- This is what is known as REORGLEAFFAR. From the manual:

  The net number of leaf pages located physically far away from previous leaf pages for successive active leaf pages that occurred since the last REORG, REBUILD INDEX, or LOAD REPLACE, or since the object was created.

  The distance between leaf pages is optimal if the difference is 1 and considered far if the distance is greater than 16.

- That’s clear, isn’t it?
  - Basically, if the IPLNXTPG pointer points to a page more than 16 pages away, REORGLEAFFAR is incremented.
  - Which means most page splits will increment by two – one for the distance to the new page, and one for the distance back to the original next page.

So why should I worry about this?

- There is also REORGLEAFNEAR. From the manual:

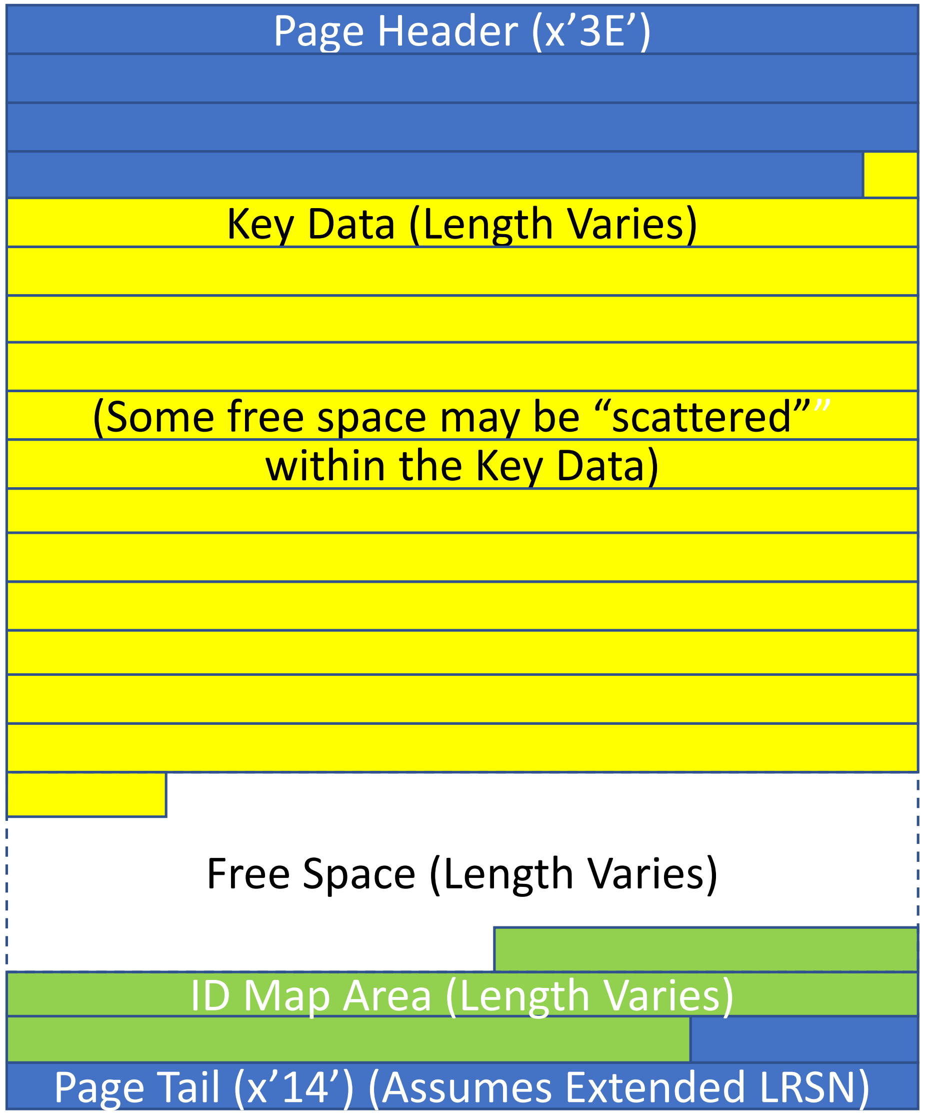
  The net number of leaf pages located physically near previous pages for successive active leaf pages that occurred since the last REORG, REBUILD INDEX, or LOAD REPLACE, or since the object was created.

  The distance between leaf pages is optimal if the difference is 1 and considered near if the distance is 2-16.

- That's clear, isn't it?
  - Basically, if the IPLNXTPG pointer points to a page between 2 and 16 pages away, REORGLEAFNEAR is incremented.
  - Which means most page splits will increment by two – one for the distance to the new page, and one for the distance back to the original next page.

Non-Leaf Pages

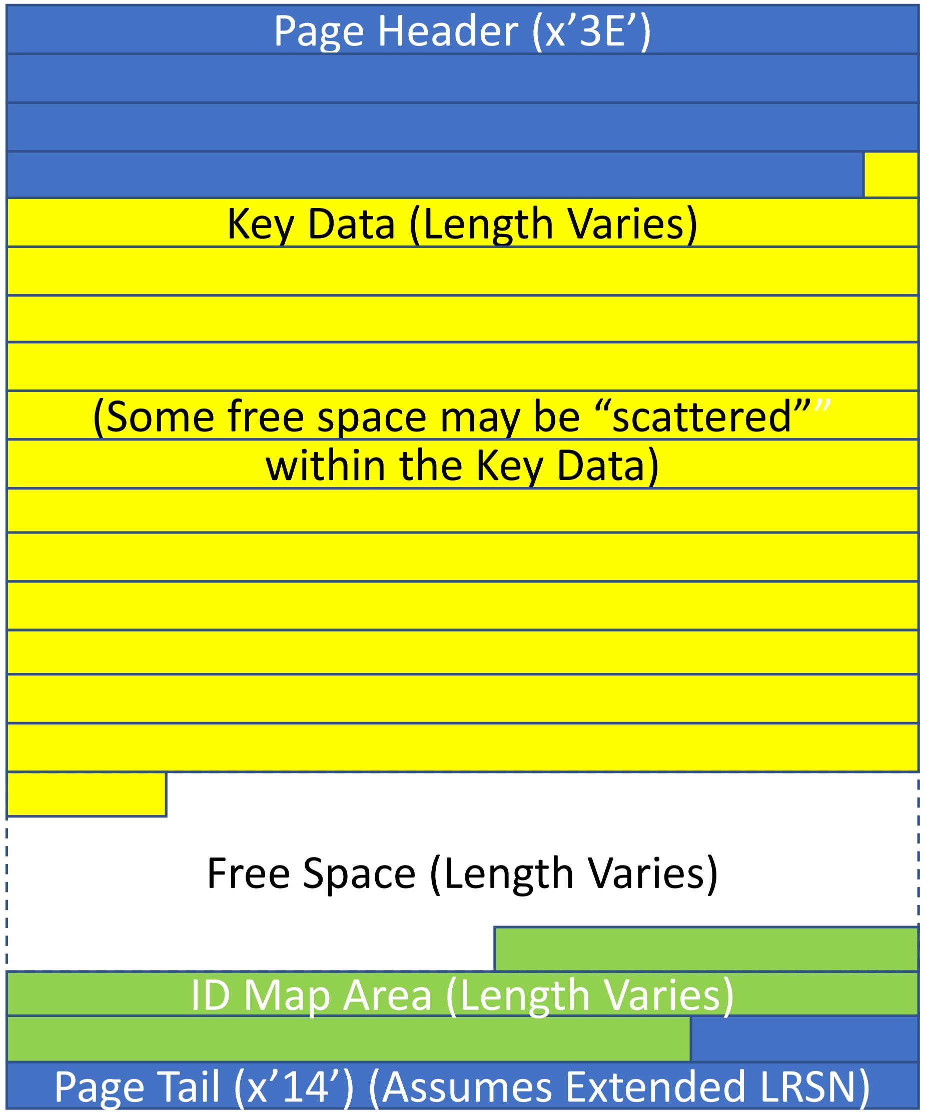
What does a Non-Leaf Page look like?


Page Header (x'3E')
Key Data (Length Varies)
(Some free space may be "scattered"" within the Key Data)
Free Space (Length Varies)
ID Map Area (Length Varies)
Page Tail (x'14') (Assumes Extended LRSN)

Amazingly like a Leaf Page.

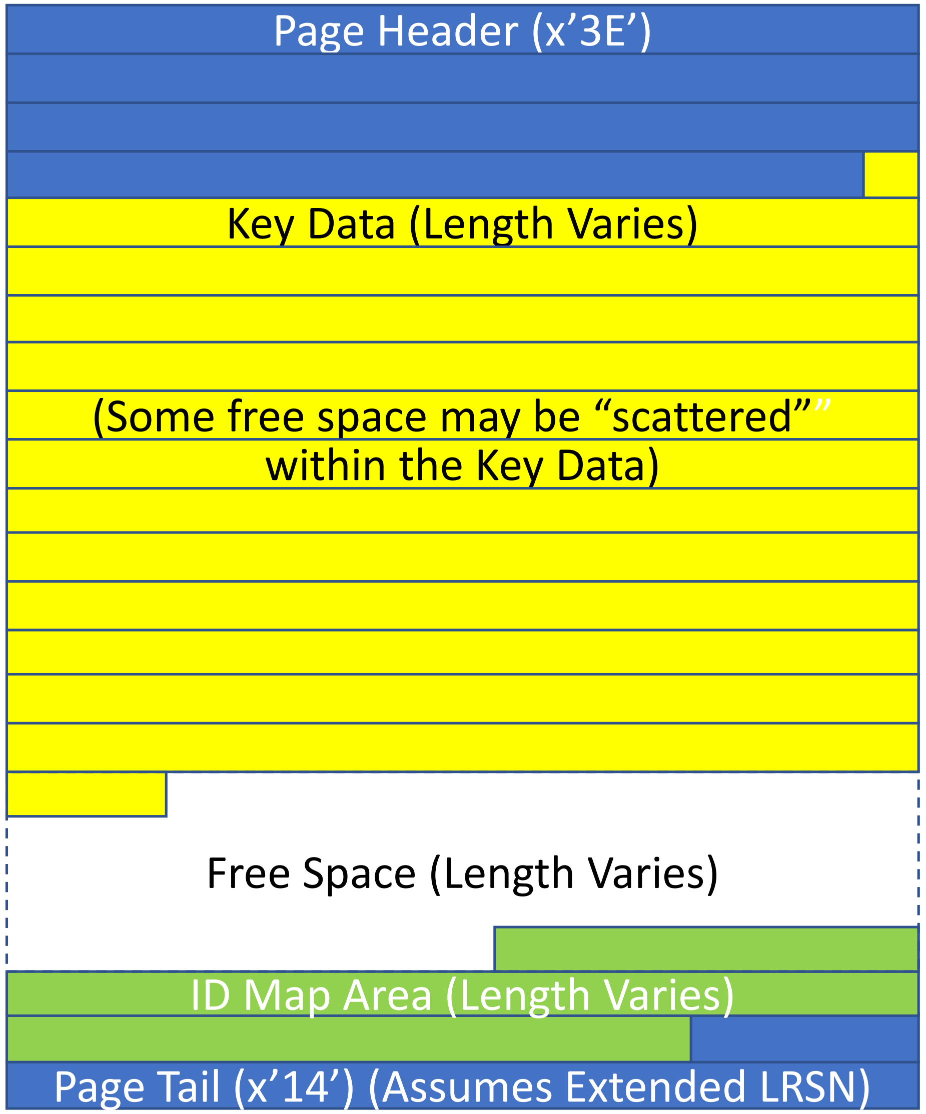
- The Page Header and Page Tail are still present (in blue).
- Keys (in yellow) begin immediately after the Page Header.
- At the bottom of the page is the ID Map (in green).
- Between the Keys and the ID Map is free space where new Keys can be inserted.
- Some free space may be interspersed within the Key area.

What is the index Root Page?


Page Header (x’3E’)
Key Data (Length Varies)
(Some free space may be “scattered”” within the Key Data)
Free Space (Length Varies)
ID Map Area (Length Varies)
Page Tail (x’14’) (Assumes Extended LRSN)

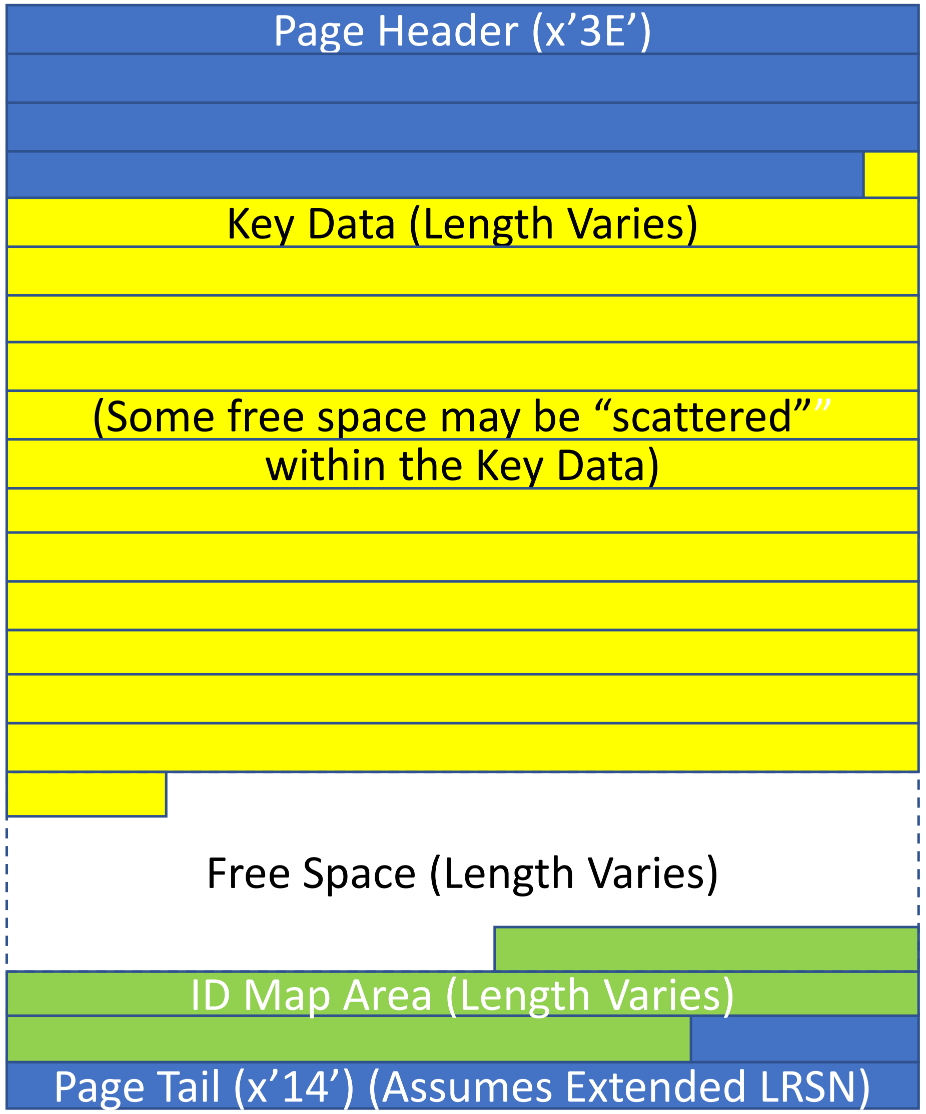
- The index Root Page is always page 2 (the third page in the index).
- What is the “Root Page”?
  - It is the “first” Non-Leaf Page.
  - Or the “highest level”.
  - It is always in the same place (page 2) so Db2 can find it easily.
  - It is how Db2 finds all of the keys in the index.
  - More later…

What does a Non-Leaf Page look like?


Page Header (x’3E’)
Key Data (Length Varies)
(Some free space may be “scattered”” within the Key Data)
Free Space (Length Varies)
ID Map Area (Length Varies)
Page Tail (x’14’) (Assumes Extended LRSN)

- All the same processes and problems exist for Non-Leaf Pages as Leaf Pages regarding page splits.
- It is possible when a Leaf Page splits that the Non-Leaf Page will also have to split. Specifically if PCTFREE is zero.
- Non-Leaf Pages search for a free page to be used just as Leaf Pages do.

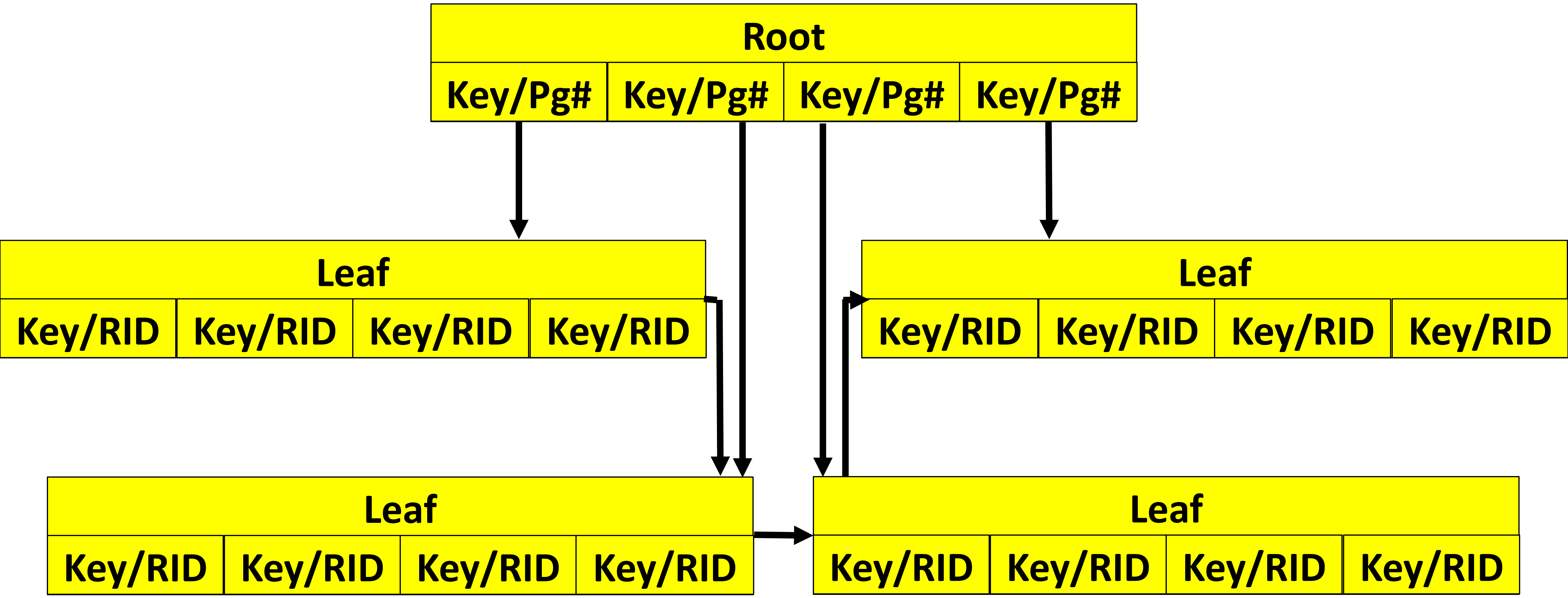
What does a Non-Leaf Page look like?



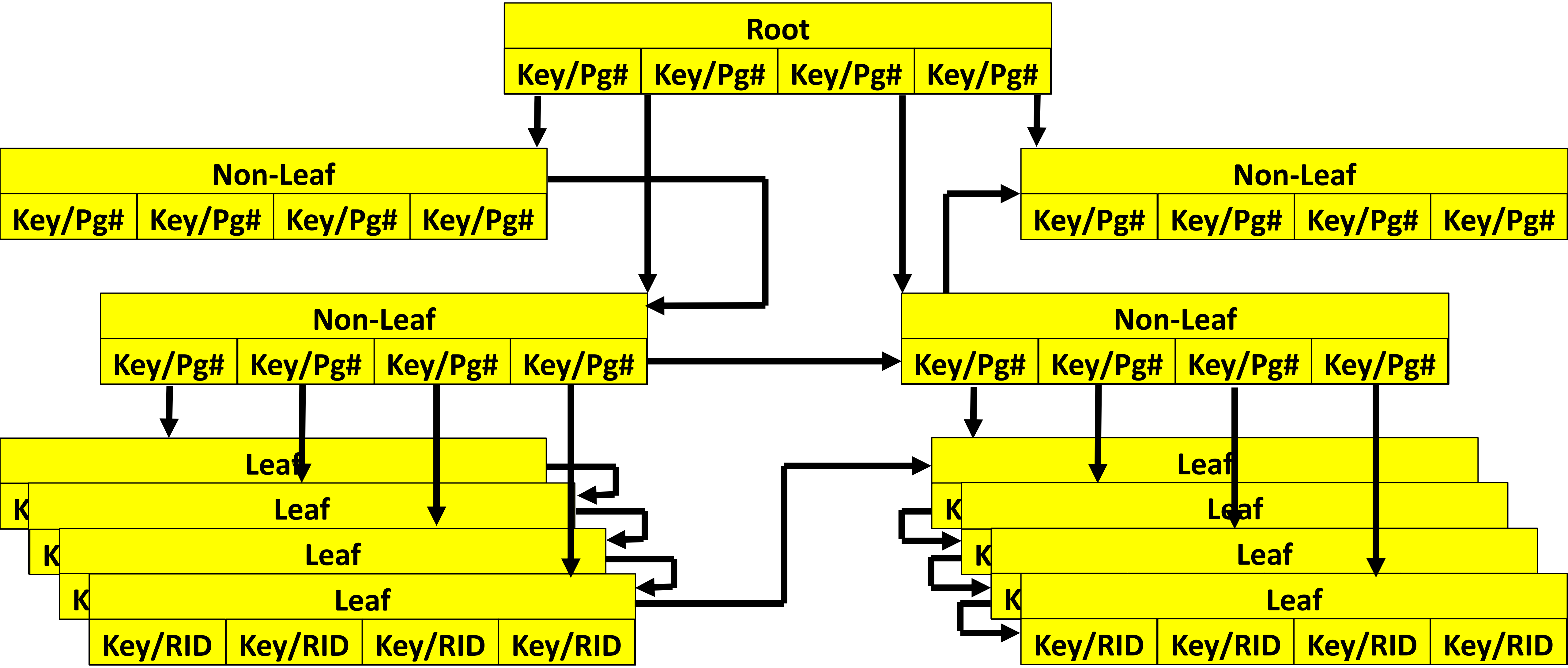
Page Header (x'3E')
Key Data (Length Varies)
(Some free space may be "scattered"" within the Key Data)
Free Space (Length Varies)
ID Map Area (Length Varies)
Page Tail (x'14') (Assumes Extended LRSN)

- Each key entry in a Non-Leaf Page has a pointer to a page within the index and the highest key value on that page.
- A Db2 Index is basically a binary tree.
- There are always at least two "levels".
  - The Non-Leaf level and the Leaf level.
  - Additional levels are added as needed.
  - See the next slides.

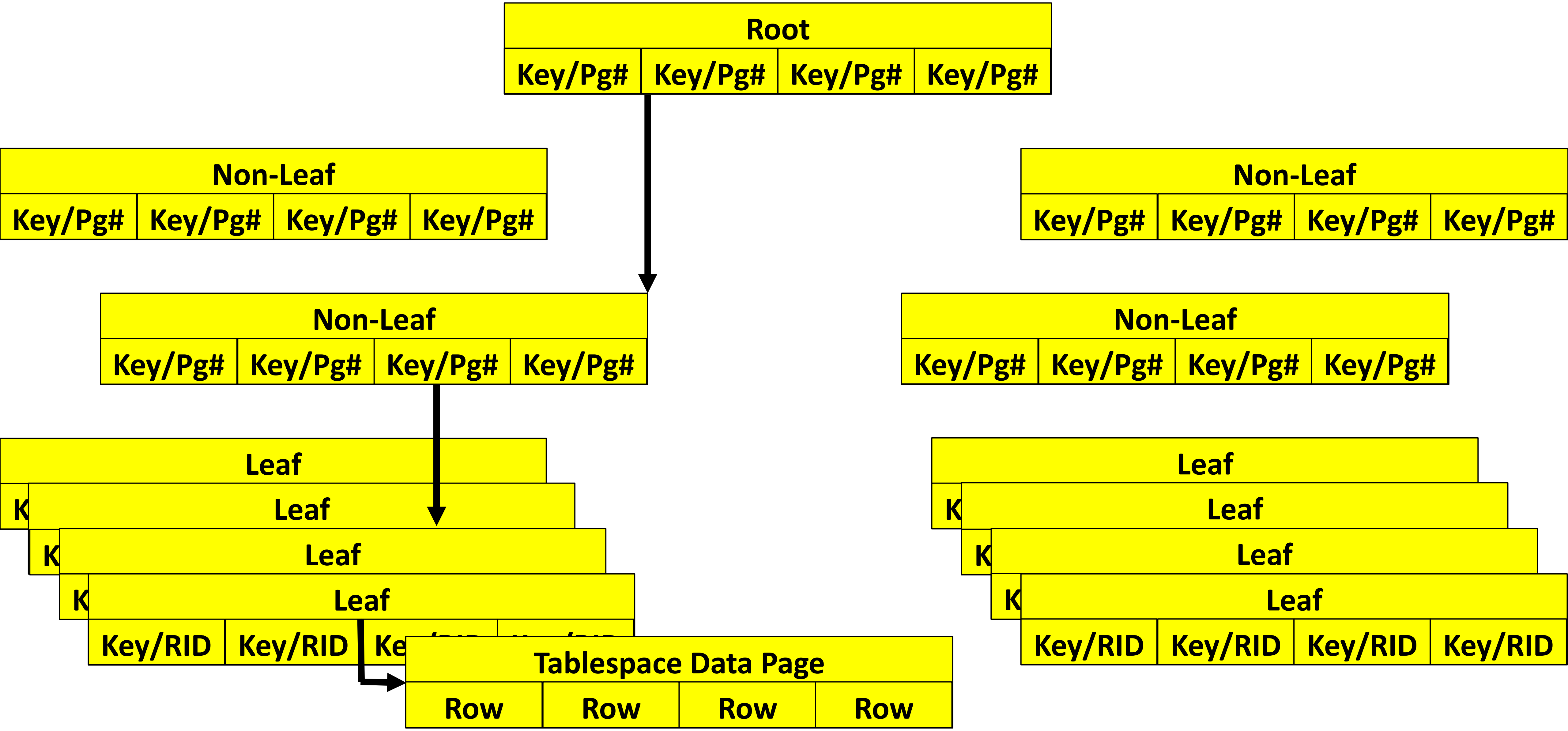
Index Levels



This is a three-level index



How does Db2 Use an Index?



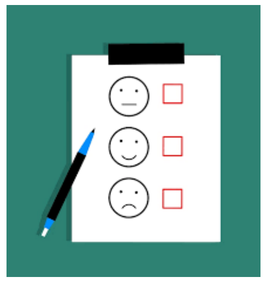
Now we ride off into the west Texas Sunset

We want your feedback!

- Please submit your feedback online at ….
  - ➢http://conferences.gse.org.uk/2018/feedback/IJ

- Paper feedback forms are also available from the Chair person

- This session is IJ



GSE UK Conference 2018

Better, stronger, faster; The Mainframe….. the Machine!

Tuesday 6th November

Start	End	Stream	Room	Title	Speaker
11:45	12:45	IMS	Wellington B	The No Cost Way to Manage the IMS Catalog	David Schipper
15:00	16:00	IMS	Wellington B	Current Trends in IMS Analytics	David Schipper
16:30	17:30	zCMPA	Woodcote	zIIP stealing GCP MSUs for Capacity Management	Donald Zeunert

Wednesday 7th November

Start	End	Stream	Room	Title	Speaker
09:30	10:30	Db2	Nurburgring	Know your onions when it comes to Db2 indexes	Randy Bright
09:30	10:30	IMS	Wellington B	IMS Checkpoint Pacing	David Schipper
10:45	11:45	zCMPA	Woodcote	How many GCP MSU is my CF stealing?	Donald Zeunert