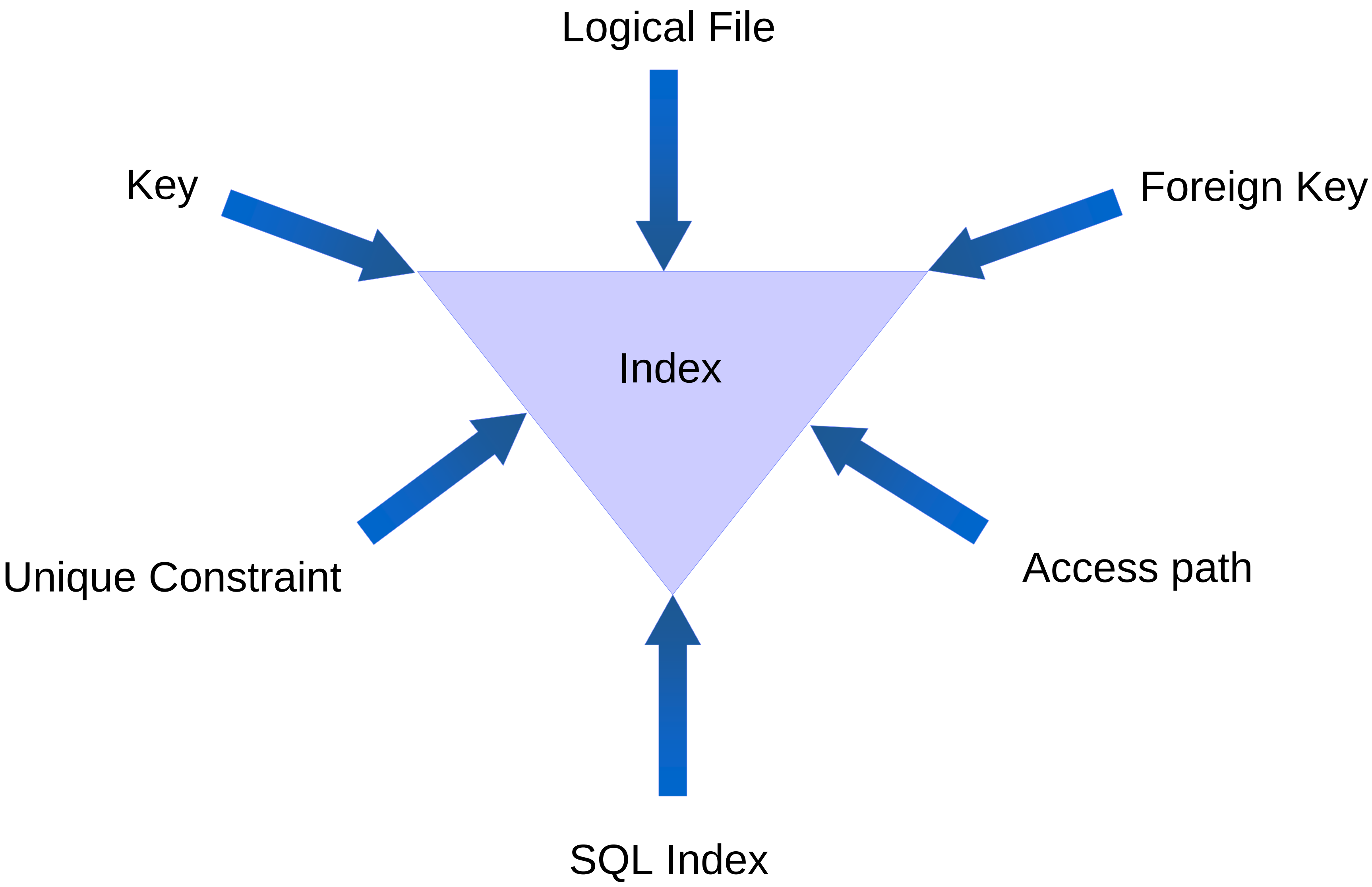
DB2 Indexing:
Tips, Tricks, & MIPS

Mark Holm
Chief Technology Officer

Agenda

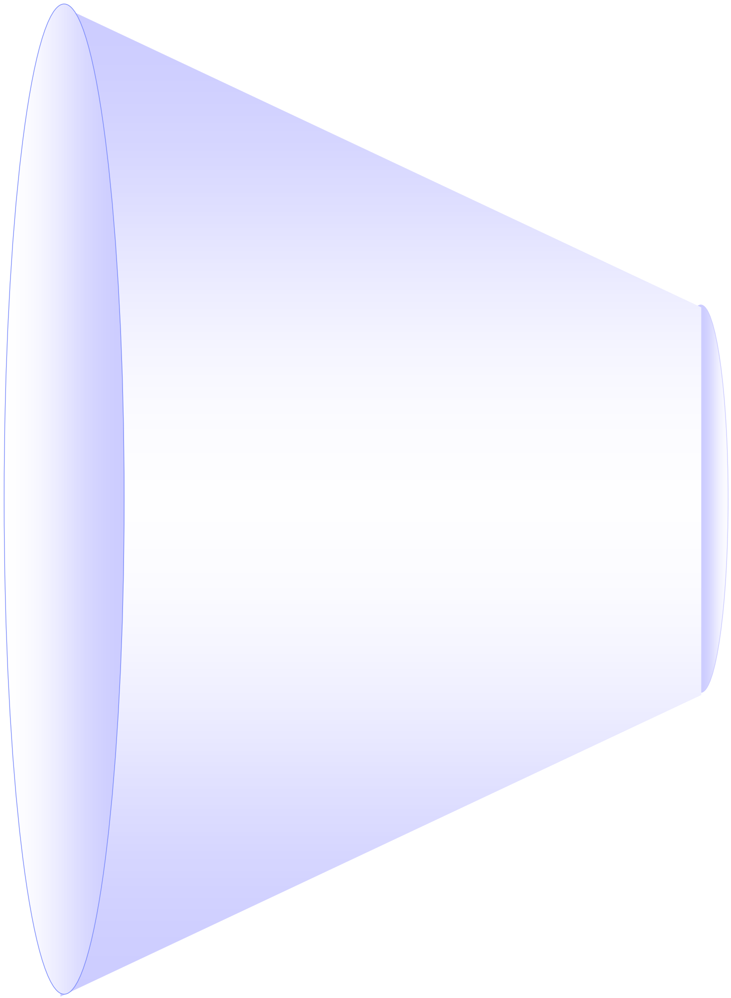
- Terminology
- What is an index – a peek under the covers
- Types of indexes
- Optimizing what you have
- Optimizing what you don’t have
- Summary of tips & benefits



Terminology: A rose by any other name…

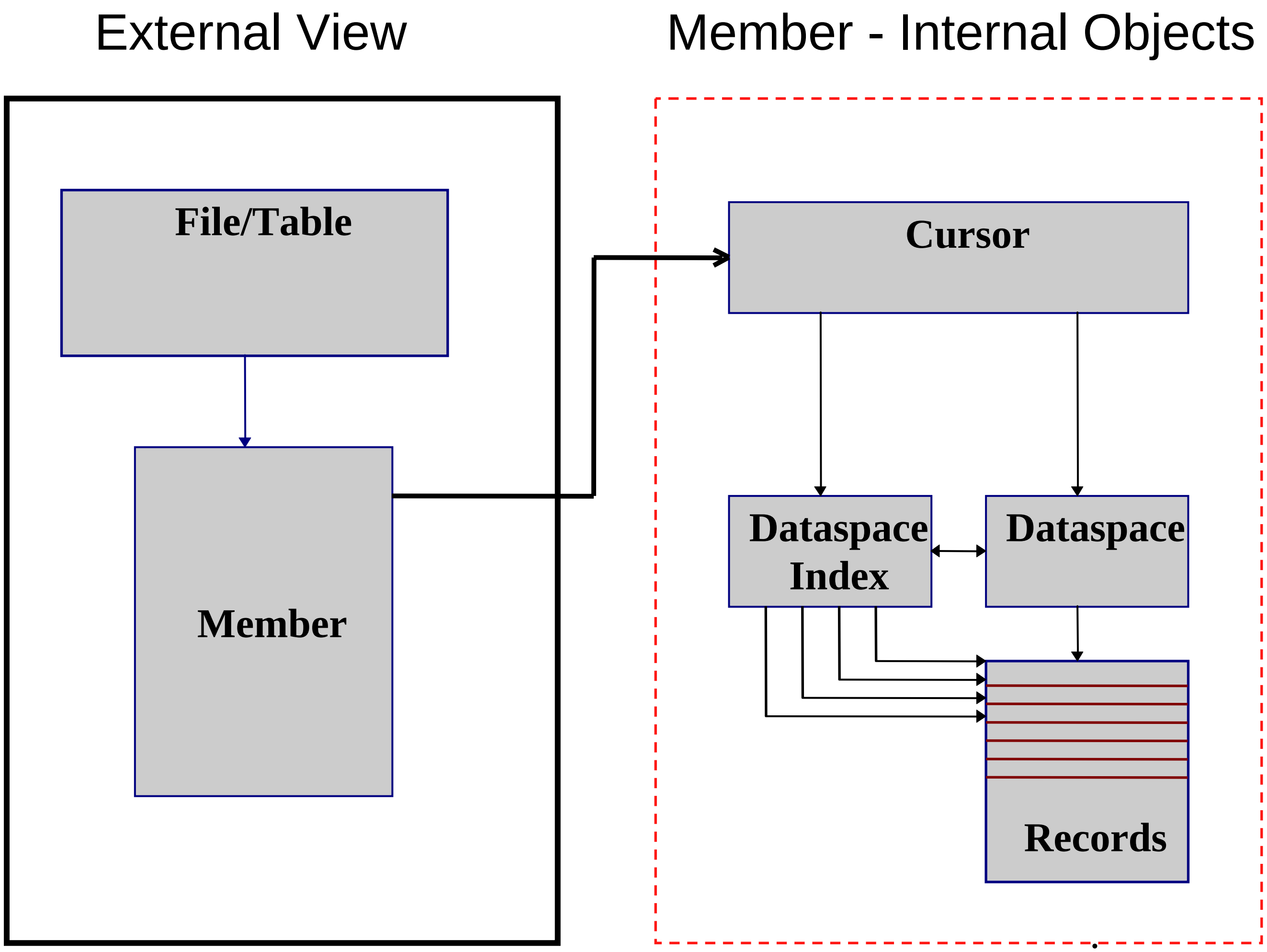
Index Uses

- Random read
  - Million scan march
  - Take 20 compares
- Uniqueness
- Ordering
- Grouping
- Data retrieval
- Orphan prevention
- Statistics

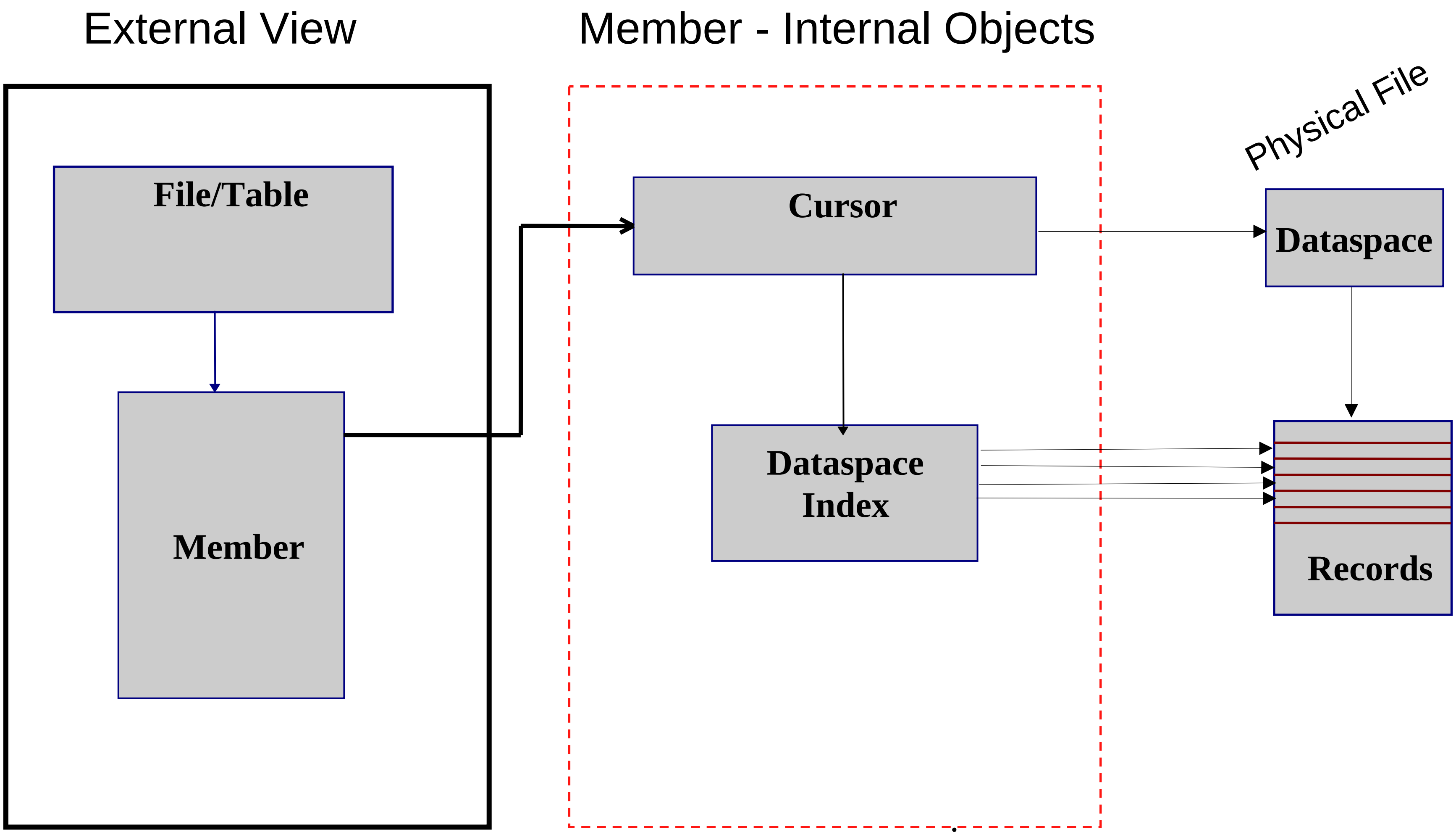


S
P
E
E
D

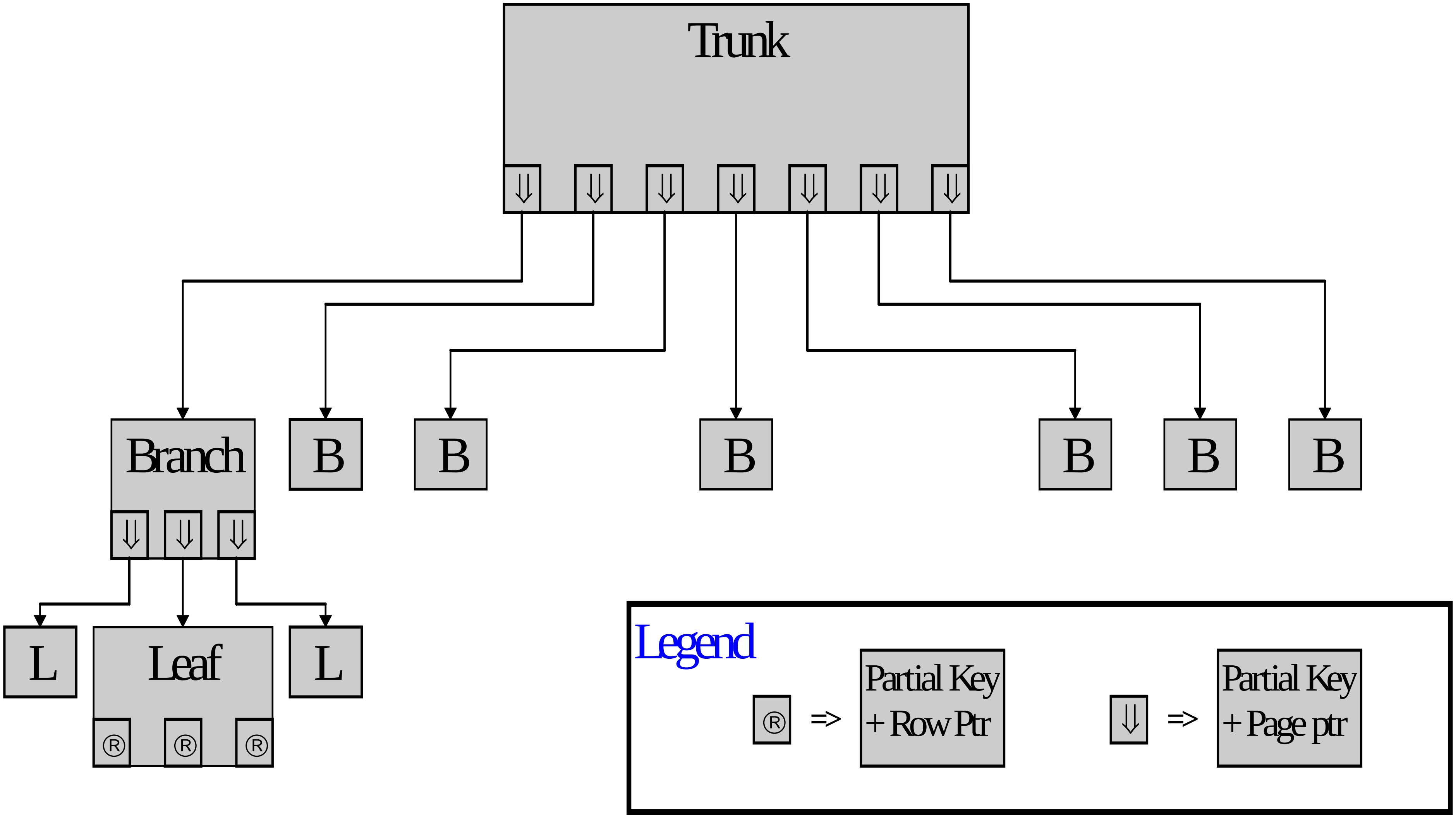
Keyed Physical File



Keyed Logical File



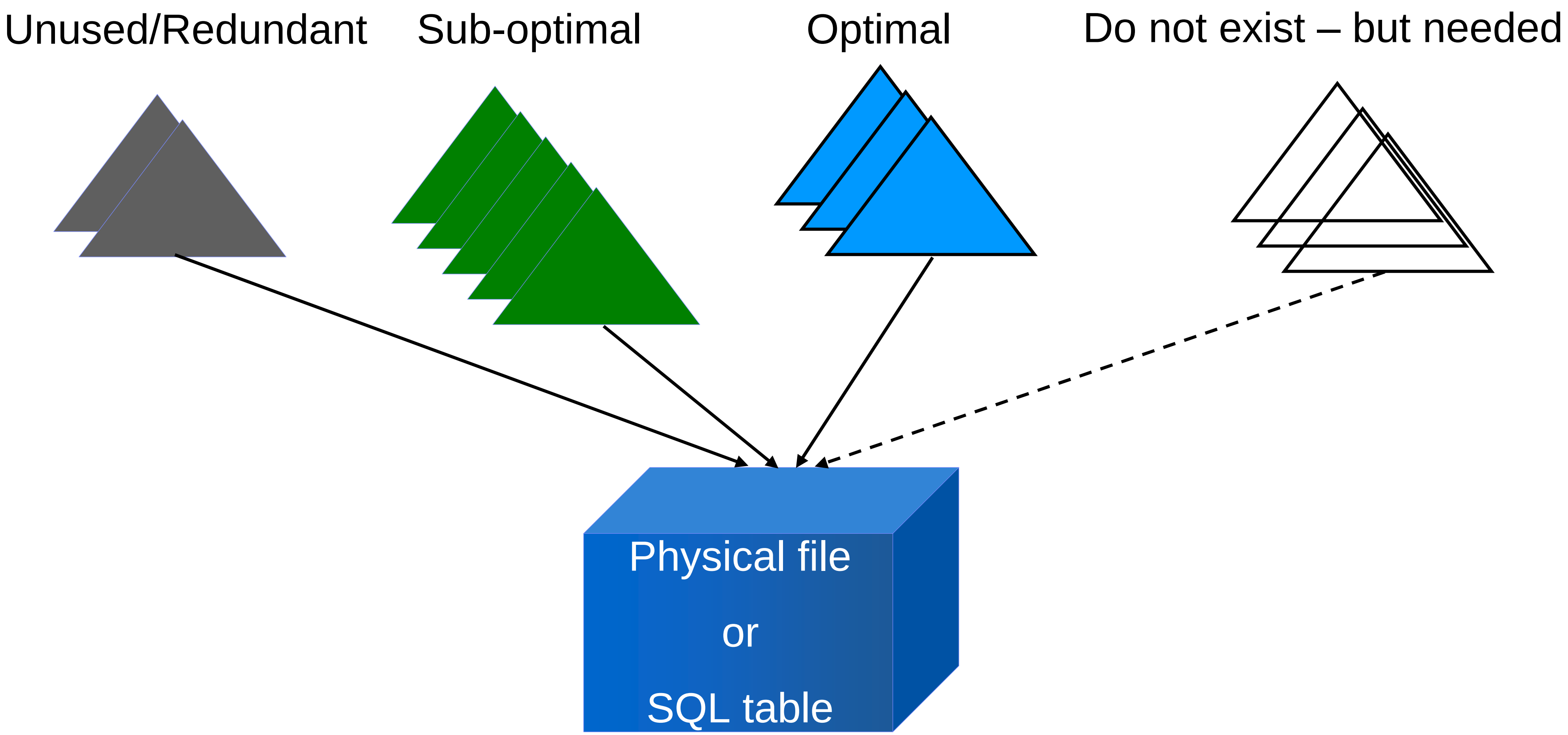
What’s it really look like?



How many type of indexes are there?

(Trick Question)

Index management



Unused/Redundant

Sub-optimal

Optimal

Do not exist – but needed

Physical file

or

SQL table

Unused or Redundant Indexes

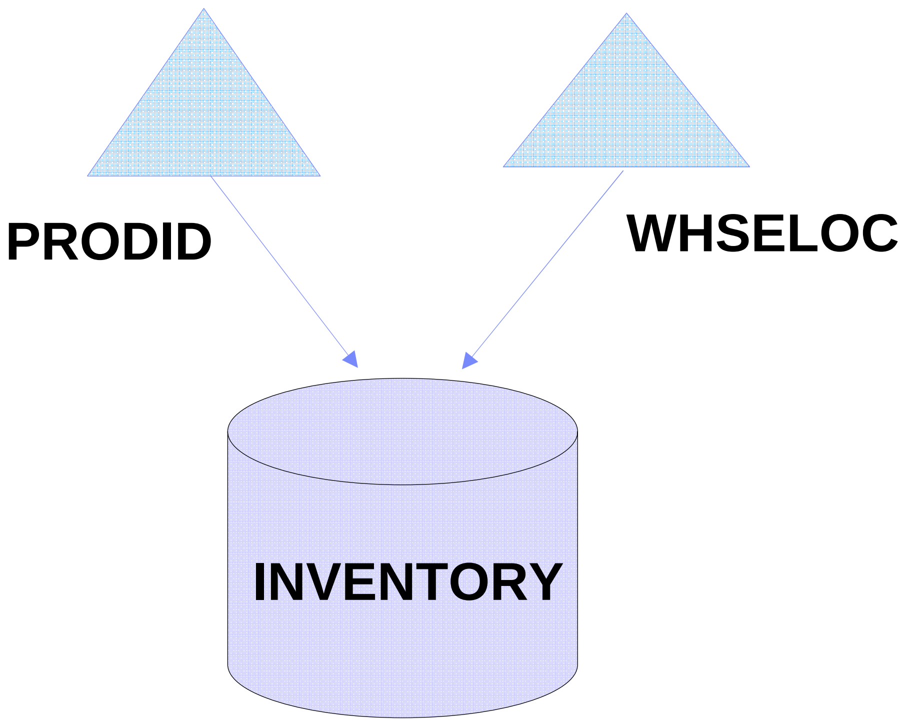
Unused index – examples

- Un-referenced – logical files
  - Not "required" (e.g. unique data, referential integrity)
  - Not used (or rarely) used by application programs
  - Not used by DB2 query optimizer to derive costing information
- SQL indexes
  - Never used for stats or to implement a query
  - Index A has subset of keys from Index B
- Maximum logical file sharing has not been done

Unused

- Used once but no longer
- Last used date not enough to identify
- Index can be used to determine what NOT to do!
- If ‘CA’ represents 40% of the rows then it makes sense to search with PRODID but WHSELOC used for stats to determine that.

SELECT * FROM INVENTORY WHERE PRODID IN (92828, 95522) AND WHSELOC = ‘CA’



Unused Indexes -- Tips

- Delete them – generally too risky
- Phase out method much less risky and can be done over a longer period of time but with immediate benefits:

CHGLF: *IMMED → *DLY → *RBLD → DLTF

- Caveat: Need to take into account “stats use” before undertaking changes.

SQL indexes

- SQL indexes created only for query performance
- Performance experimentation and/or application changes can lead to index proliferation
- Example: two indexes
  - One with key of ORDERNUM
  - One with key of ORDERNUM, LINENUM
  - First index is likely not necessary!
- CAVEAT: The cost for DB2 to use an index with more keys is higher

SQL indexes - Tips

- Identify redundant indexes by looking at all indexes above a table
  - DSPDBR, DSPFD
  - iSeries Navigator
- Remove redundant indexes (DROP INDEX)
- Based on caveat about indexes with more keys, look at actual use by optimizer before deleting the index

Logical files -- Sharing

- CRTLF FILE(PROD/A) →

```
P1KF   PFILE(P3KU)
K  F3
K  F4
```

- CRTLF FILE(PROD/B) →

```
P1KF   PFILE(P3KU)
K  F3
K  F4
K  F5
```

- Should be one index with keys F3, F4, F5

Logical files – Sharing Tips

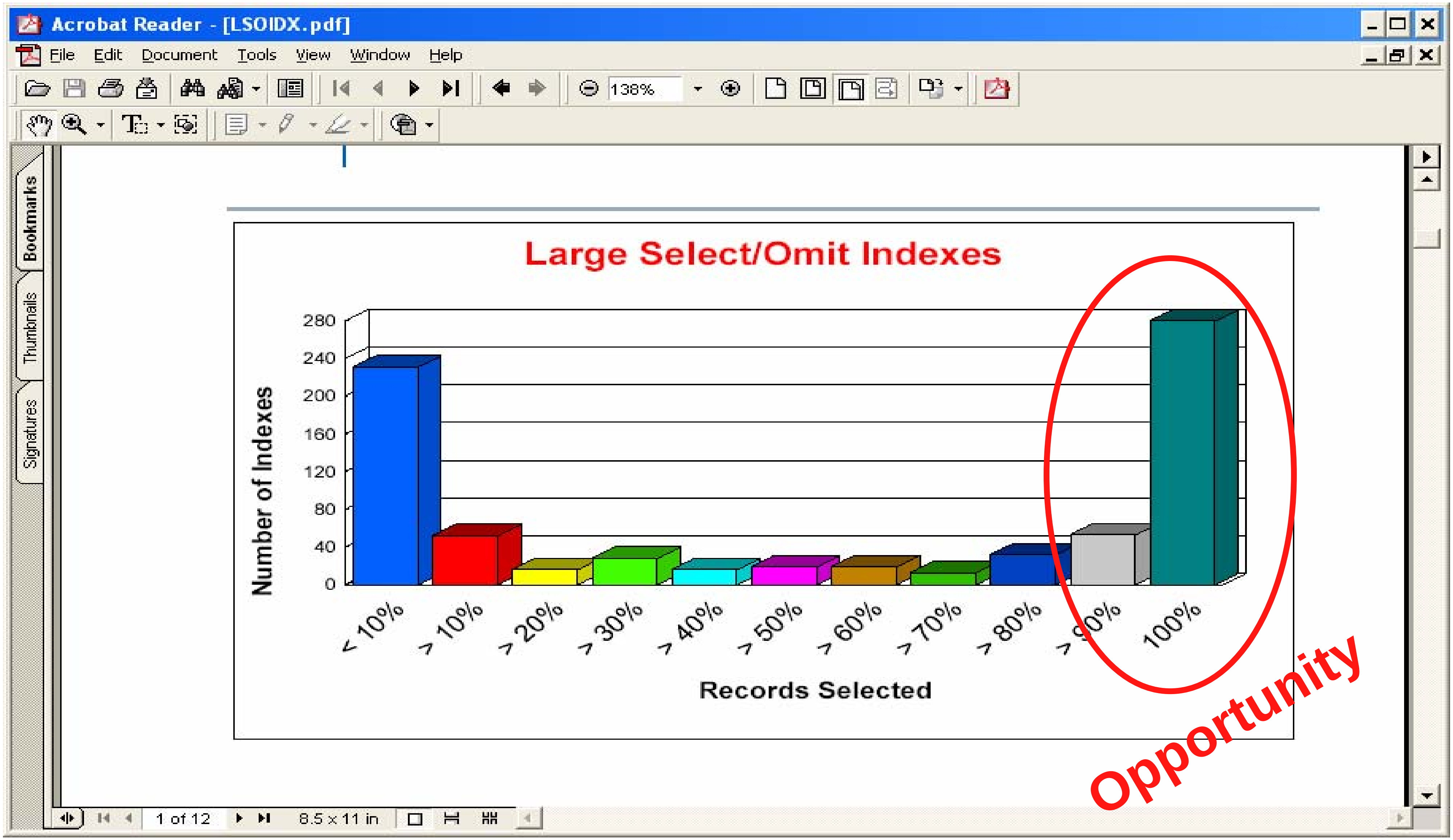
- Two options to get sharing after the fact
  - Recreate the files
    - DLTF
    - CRTLF in longest to shortest order
  - Save/restore
    - SAVOBJ/SAVLIB
    - DLTF – otherwise restore will just replace
    - RSTOBJ/RSTLIB – DB2 figures it out for you

Suboptimal Indexes

Optimizing an existing index - examples

- Select/Omit logical files
- *MAX4GB vs. *MAX1TB indexes
- Small vs. large logical pages
- Clustered indexes
- Force access path

Select/Omit logical files



Opportunity

Use of DYNSLT - Tip

```
R P1KF                 PFILE(P3KU)
K F3
K F4
S F3                   COMP(LT 0)
```

```
                       DYNSLT
R P1KF                 PFILE(P3KU)
K F3
K F4
S F3                   COMP(LT 0)
```

File level: 1070305162543

Format level: 432937C5F3D41

File level: 1070305162543

Format level: 432937C5F3D41

NET: No recompile required

*MAX4GB vs. *MAX1TB

- Determines the maximum size of the index
- More importantly, determines the STRUCTURE of the index and the DB2 CODE that processes it
  - 1TB indexes much more friendly to key field changes
  - Reduces internal locks and increases intra-job parallelism
- Default for CRTLF on all supported releases, but old indexes are not automatically converted

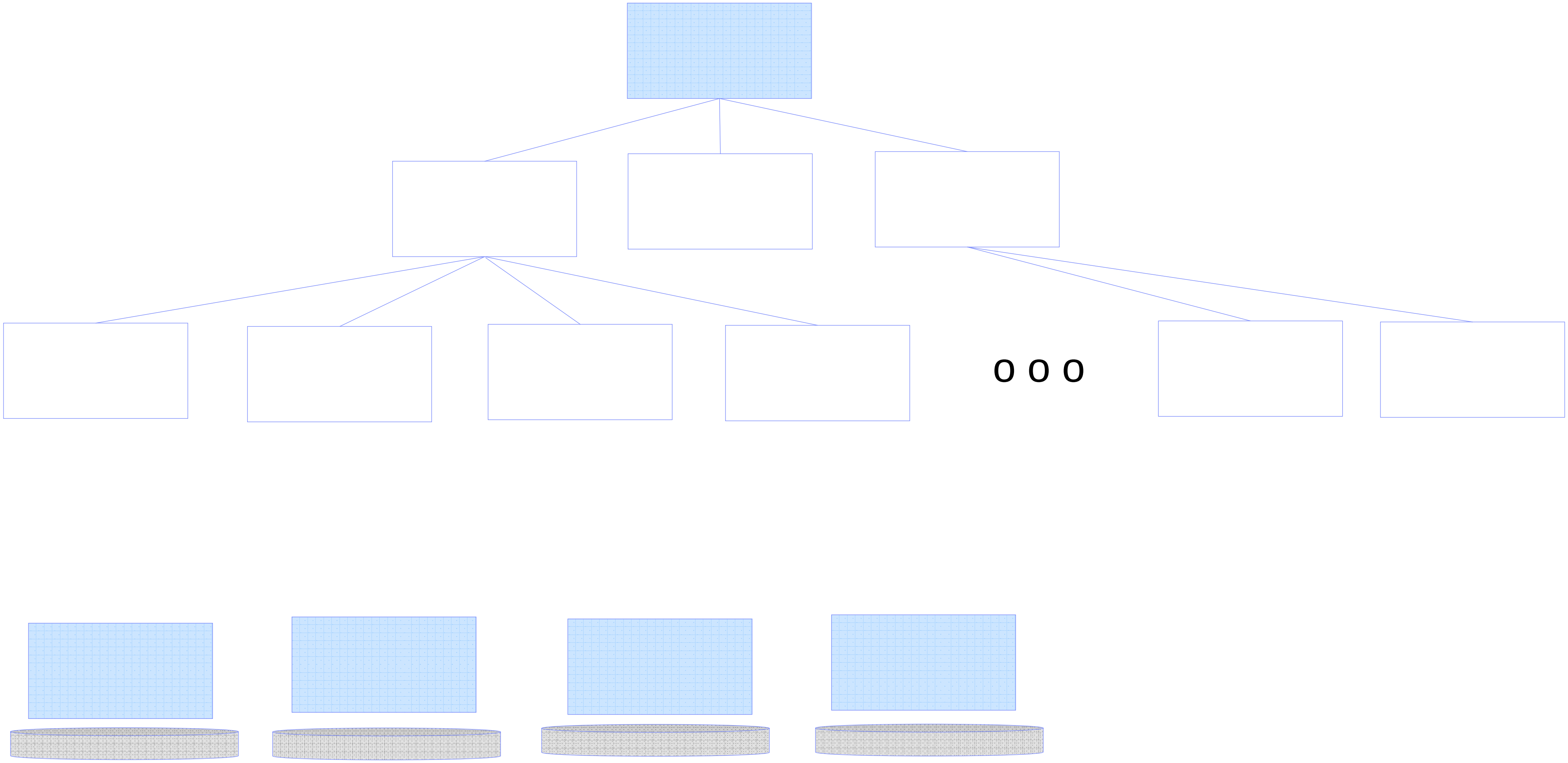
*4GB vs. *1TB - Tip

- For highly updated files convert all logical files to 1TB:
  - CHGLF FILE(lib/file) ACCPTHSIZ(*MAX1TB)
- If a file is rarely updated *MAX4GB works fine and is slightly more efficient

Logical Page Size

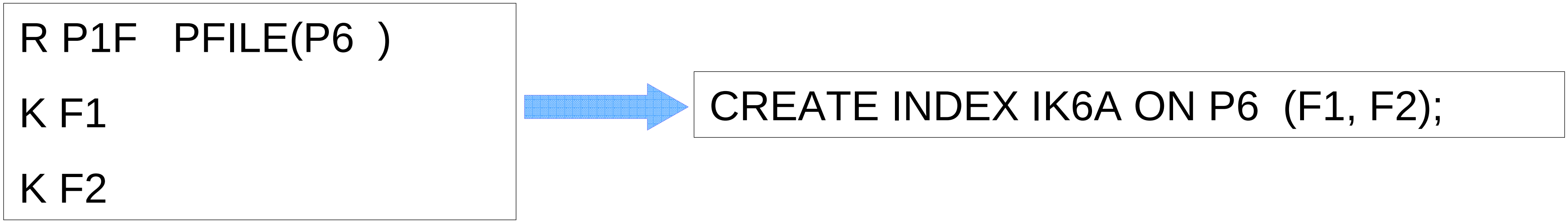
- DDS files default to 8K logical pages
- SQL indexes default to 64K logical pages
- Larger pages mean much better performance if:
  - The index is processed sequentially (A→Z) often or by time-sensitive batch work or reporting
  - The system is not memory constrained

Logical page size - affect on disk I/O



Logical Page Size – Tip #1

- Slip SQL indexes under logical files
  - “Trick” system into using SQL index for logical file
  - Get the benefit of large pages without code changes
  - Only needed on V5R3 or prior releases
- Sequence is 1) DLTF 2) CREATE INDEX 3) CRTLF or 1) SAVOBJ 2) CREATE INDEX 3) RSTOBJ



```
R P1F   PFILE(P6  )

K F1

K F2
```

```
CREATE INDEX IK6A ON P6  (F1, F2);
```

Logical Page Size – Tip #2

- On V5R4 logical page size can be explicitly set
  - CRTLF PAGESIZE parameter
  - CREATE INDEX PAGESIZE keyword
  - 8K to 512K supported
  - Default sizes are the same as previous releases
- Only use on indexes that you know are processed sequentially!

```
CRTLF FILE(*CURLIB/L6KB) PAGESIZE(32)
```

```
CREATE INDEX IK6A ON P6  (F1, F2) PAGESIZE(256)
```

Clustered index

- Basic idea – get the records/rows in the same order as an index built over the table
- Essentially the same idea as large logical pages
  - For each trip to the disk, get a lot of data to process
  - **Sequential index** processing can be dramatically faster
- Ancient concept – not implemented for DB2 on the iSeries/System i
- Prior to V5R3, clustering data was simply not practical for most shops
- “Poor mans” clustered index is now possible with a bit of work

Clustered index - Tip

- Clustering is done with RGZPFM – concurrent flavor
- Example
  - RGZPFM FILE(XCD0003/P6) KEYFILE(XCD0003/L6KA M1) RBDACCPTH(*NO) ALWCANCEL(*YES) LOCK(*SHRUPD)
- File must be journaled
- Physical order of the rows must not matter to an application (sensitive to RRN)
- High Availability load needs to be considered

Force Access Path

- FRCACCPTH(*YES) parameter used on one of the following commands:
  - CRTPF
  - CRTLF
  - CHGPF
  - CHGLF
- ALWAYS A MISTAKE! High cost – no benefit.
- Simply change to FRCACCPTH(*NO)

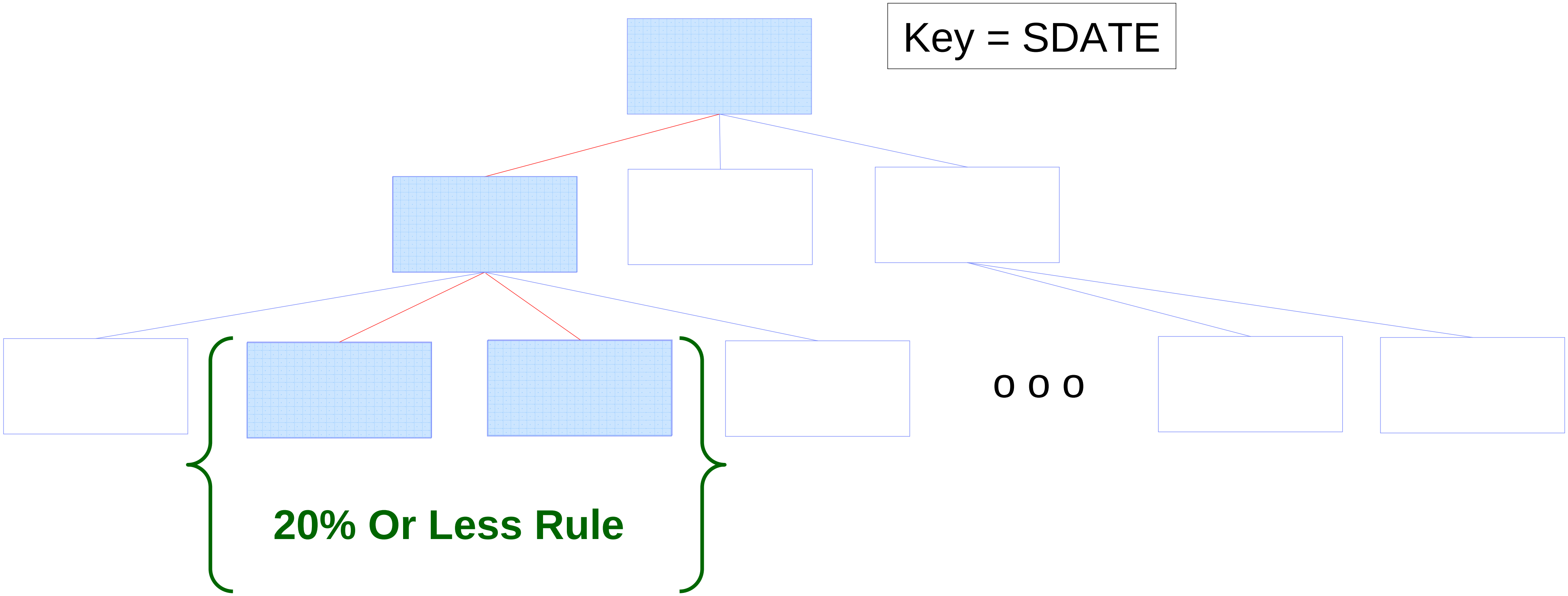
Needed Indexes

Why create indexes? By example….

- Speed – queries/SQL
  - WHERE clause
  - Joining tables
  - Grouping
  - Ordering
  - Index Only Access
  - Statistics
- Binary Index vs. Encoded Vector Index

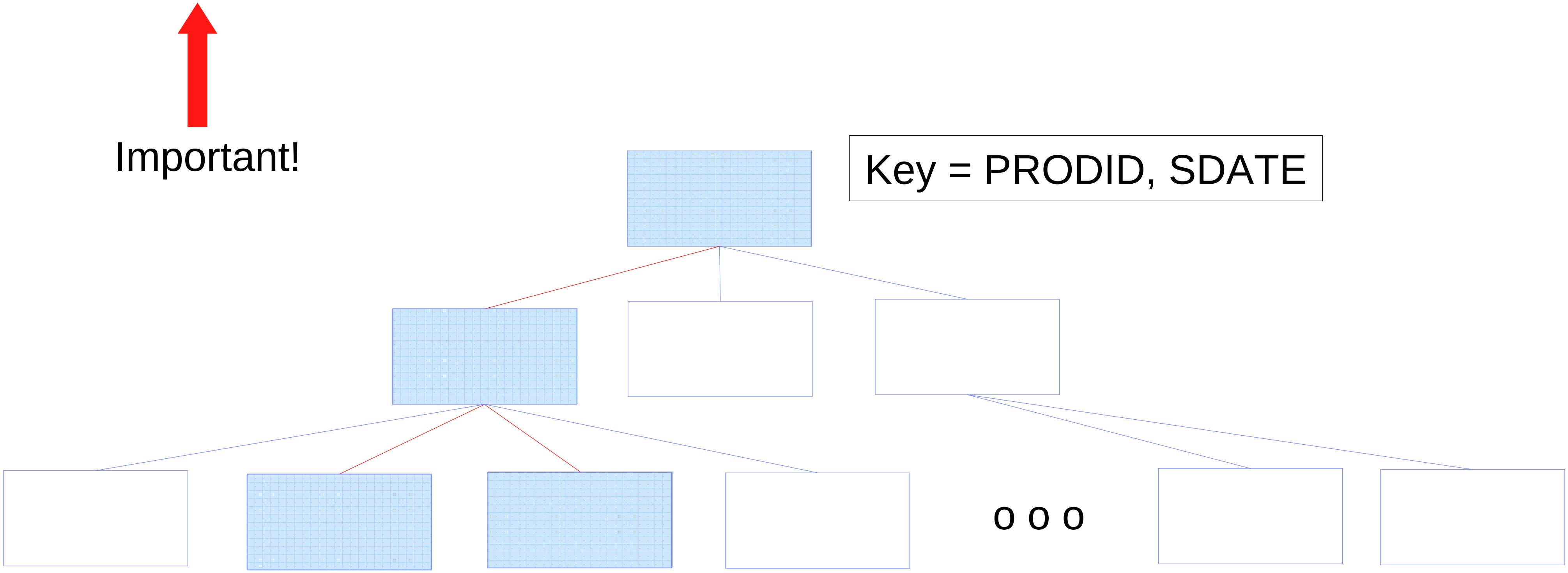
WHERE clause – example #1

- SELECT SUM(AMOUNT) FROM SALES WHERE SDATE BETWEEN :lowdate and :highdate



WHERE clause – example #2

- SELECT SUM(AMOUNT) FROM SALES WHERE PRODID = :pid **AND** SDATE BETWEEN :lowdate and :highdate

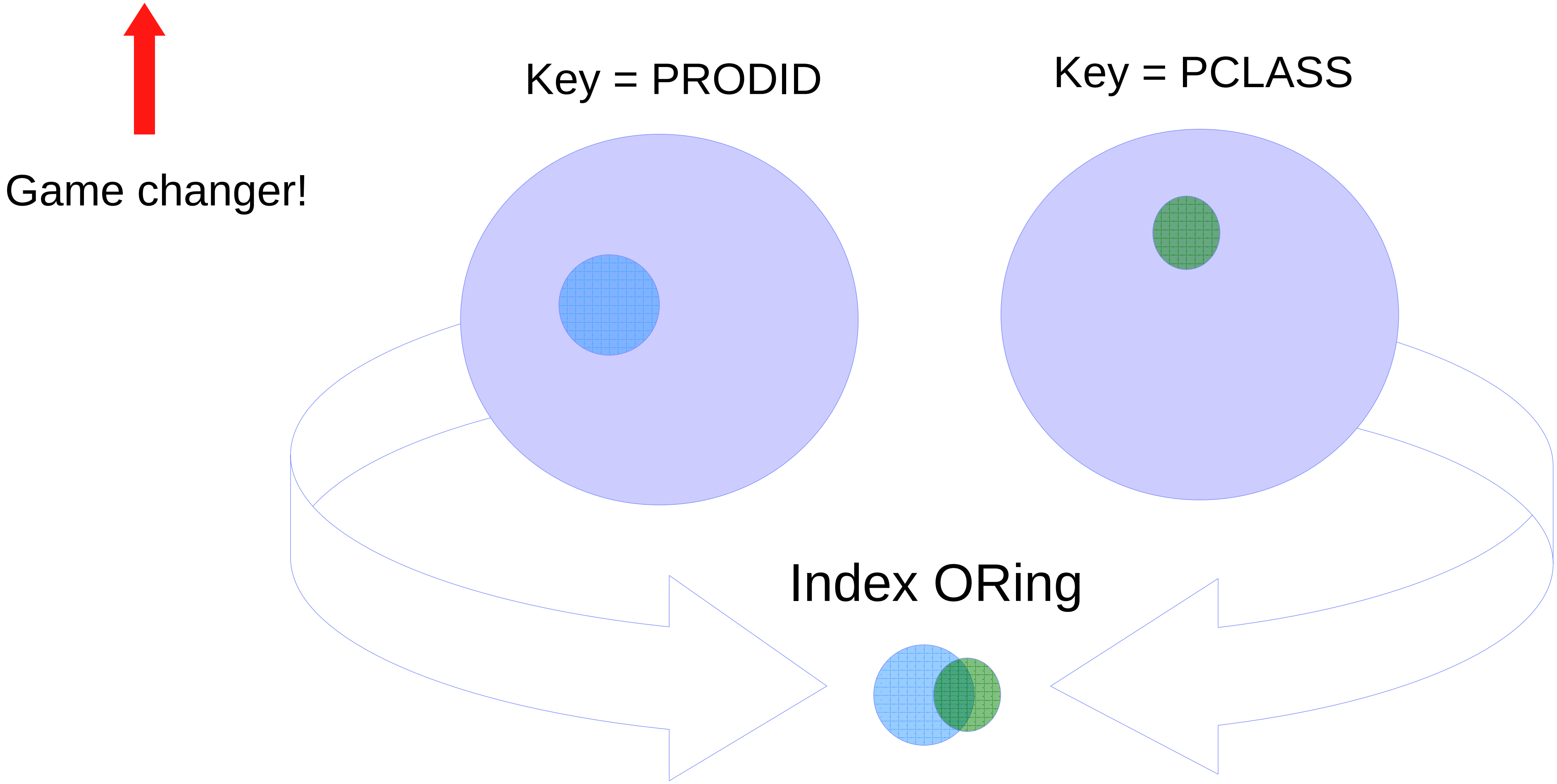


Important!

Key = PRODID, SDATE

WHERE clause – example #3

- SELECT SUM(AMOUNT) FROM SALES WHERE PRODID = :pid **OR** PCLASS BETWEEN :minclass and :maxclass



Game changer!

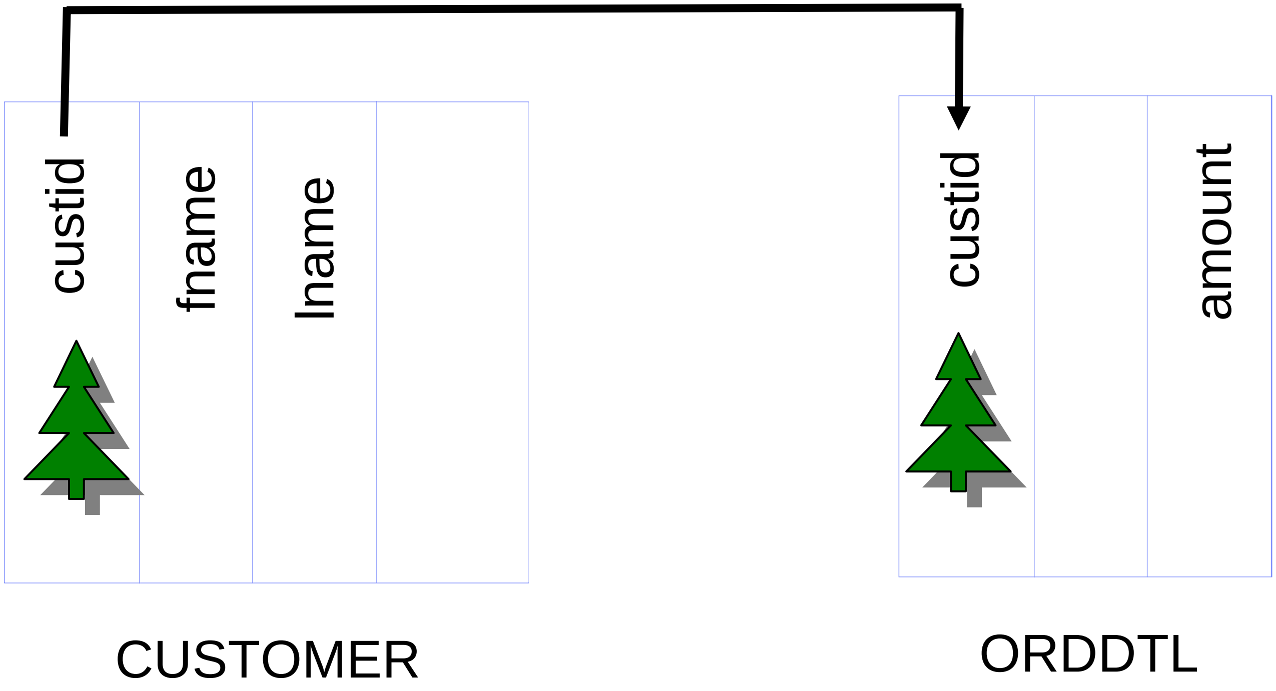
Key = PRODID

Key = PCLASS

Index ORing

Joins – example #1

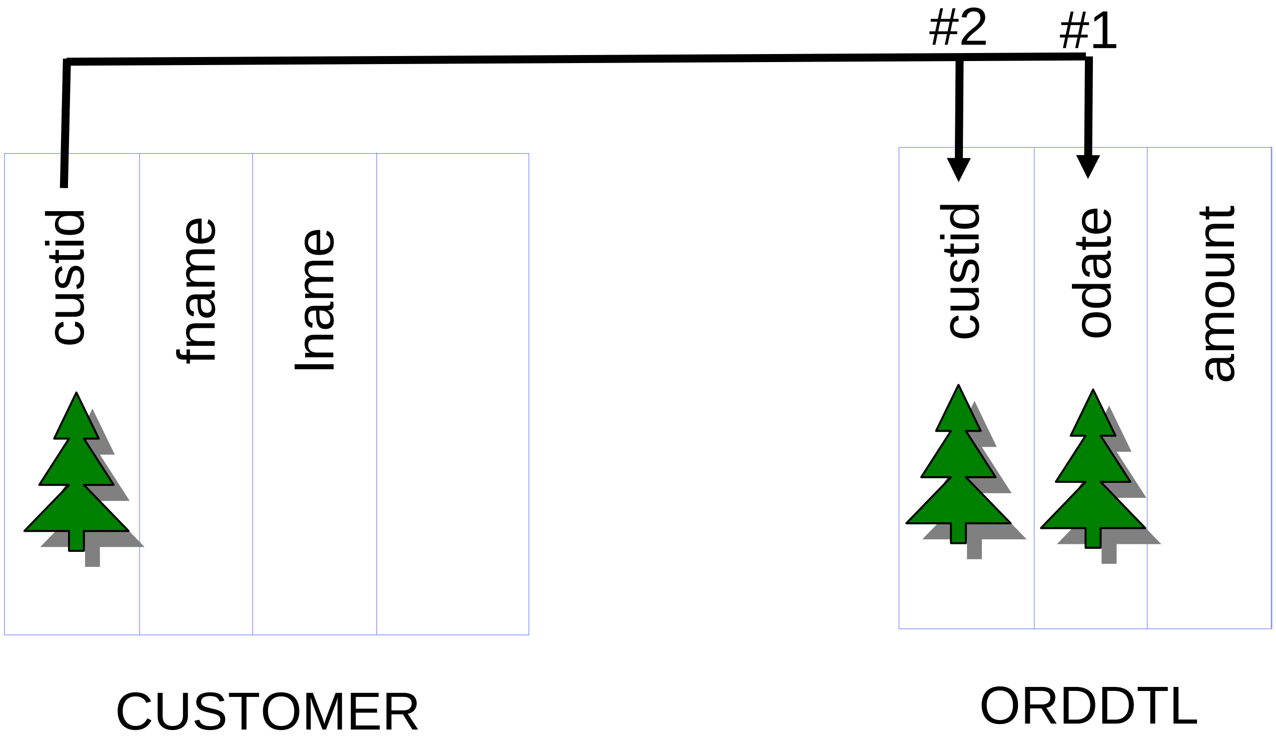
- SELECT C.FNAME, C.LNAME, O.AMOUNT FROM CUSTOMER C, ORDDTL O WHERE C.CUSTID = O.CUSTID



Joins – example #2

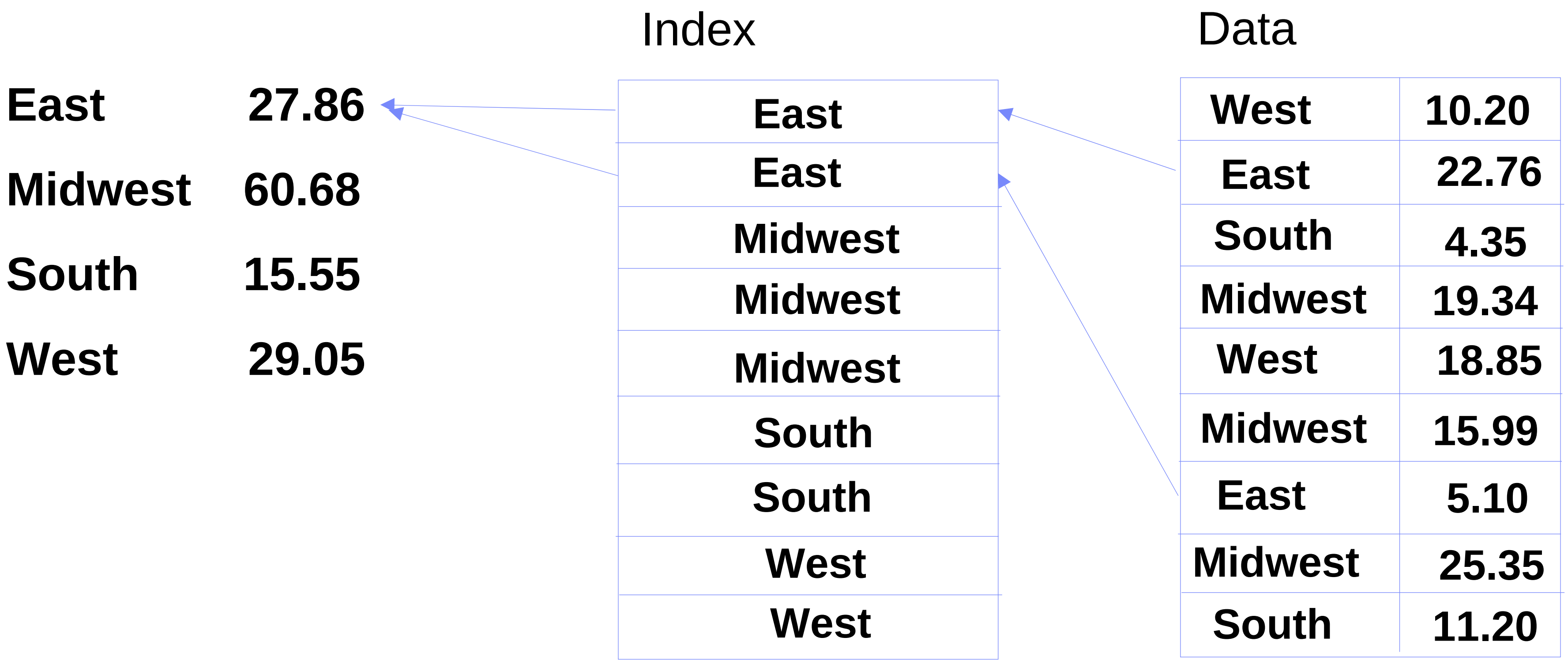
- SELECT C.FNAME, C.LNAME, O.AMOUNT FROM CUSTOMER C, ORDDTL O WHERE C.CUSTID = O.CUSTID AND O.ODATE = :orderdate

Key order → Perfect Index



Grouping - example

- SELECT REGION, SUM(AMOUNT) FROM SALES WHERE PRODID = :pid GROUP BY REGION



East	27.86
Midwest	60.68
South	15.55
West	29.05

Index

East
East
Midwest
Midwest
Midwest
South
South
West
West

Data

West	10.20
East	22.76
South	4.35
Midwest	19.34
West	18.85
Midwest	15.99
East	5.10
Midwest	25.35
South	11.20

Grouping - Tip

- Index over grouping columns useful for two reasons
  - DB2 can use it to easily correlate members of the group
  - Statistics – Query optimizer can easily determine how many groups there are in the table as a whole
    - Low number of groups allows the optimizer to also consider a “hash grouping” algorithm
- Perfect indexes can also be built for grouping queries (WHERE PRODID = :pid GROUP BY REGION – key can be PID, REGION)

ORDERing - example

- SELECT REGION, SUM(AMOUNT) FROM SALES WHERE PRODID = :pid ORDER BY REGION

Perfect index = PRODID, REGION

Index

East
East
Midwest
Midwest
Midwest
South
South
West
West

Data

West	10.20
East	22.76
South	4.35
Midwest	19.34
West	18.85
Midwest	15.99
East	5.10
Midwest	25.35
South	11.20

Index only access - example

- SELECT ordnum, prodid, custid FROM ORDDTL WHERE ordnum = :ordernum
- If index has key of ordnum, prodid, custid then the row does not need to be accessed
  - All needed data is in the key
  - Will save disk I/O and CPU and memory

Stats - example

- Just because an index is not “used” doesn’t mean it isn’t useful – how can this be?
- Answer: If it tells the query optimizer an important fact about the data itself
- Examples of information an index can provide:
  - The number of rows that equal a specific value or range of values
  - The number of groups in the table for key values
  - The number of distinct values for a column or set of columns

Encoded Vector Indexes – quick mention

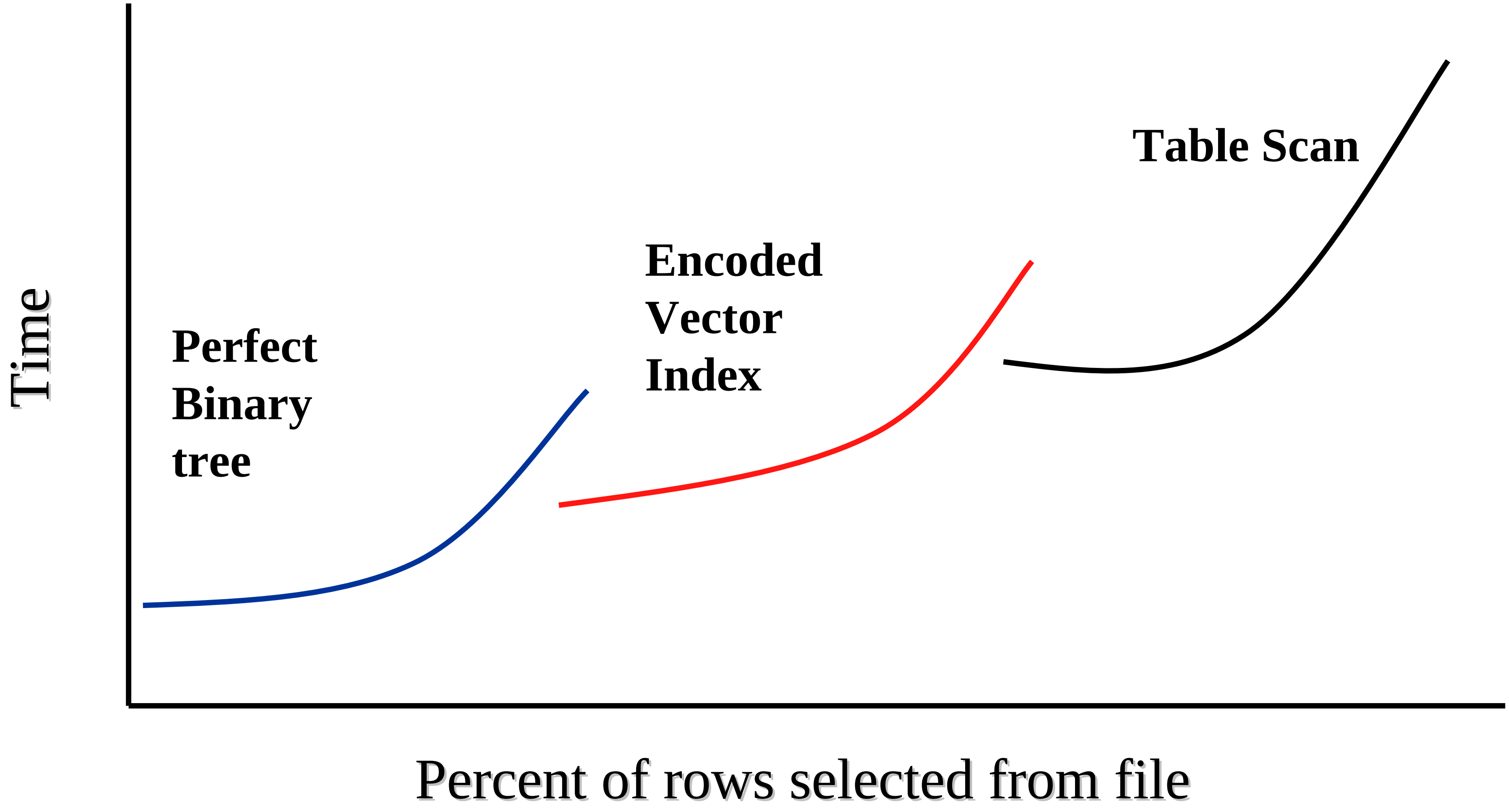
Symbol table

Key	Code	First row	Last row	Count
Amsterdam	1	2355	5321	120
Anaheim	2	32	6490	266
Boise	3	7568	7639	5
….				
Washington	1503	5576	8792	320

Vector

3	1503	3456	…...	32

Encoded Vector – where they fit



Encoded Vector - Tips

- Generally not built over tables that are read much more than updated
- Generally built over columns with a modest number of distinct values (avoid unique columns or data types like Timestamp)
- Generally are focused on queries that must process between 20% and less than 70% of the rows in the table

Summary

- Optimizing unused/redundant improves:
  - Disk space utilization and disk I/O
  - Memory use (due to update activity)
  - CPU to maintain them
- Optimizing indexes that are not efficient improves:
  - Disk I/O for reading data
  - Disk I/O for updating the index
  - Seize contention
- Adding indexes improves:
  - Disk I/O for reading data
  - CPU use for queries
  - Query optimizer’s choice of an appropriate algorithm

For further information or help

www.centerfieldtechnology.com

- Out in Left Field bi-monthly publication
- FAQ: http://www.centerfieldtechnology.com/asktheexpert_viewquestions.asp
- Custom DB2 education
- expert@centerfieldtechnology.com

www.ibm.com

- ibm.com/servers/enable/site/education/abstracts/indxng_abs.html

Publications

- http://publib.boulder.ibm.com/infocenter/iseries/v5r3/topic/rzahg/rzahgdb.htm
- http://publib.boulder.ibm.com/infocenter/iseries/v5r4/topic/rzahg/rzahgdb.htm

Thanks for Coming!