S68

Secondary Index Performance

Kyle Lindvall

IMS Technical Conference

Miami Beach, FL October 22-25, 2001

RETURN TO INDEX

Performance Basics

The rule is:

~~Secondary Index Databases~~

Best overall performance, however, may require Secondary Indexes

Examples: Access by name when number is sequence field
Access to low level segment in large database

Options: Scan the database or use Secondary Index

Many, many I/Os vs 3-5 I/Os

All There Is To Know

- If you must use Secondary Indexes, then:
  - Direct pointer
  - Unique key
  - Sparse is possible

- Avoid HISAM as target

What is a Secondary Index?

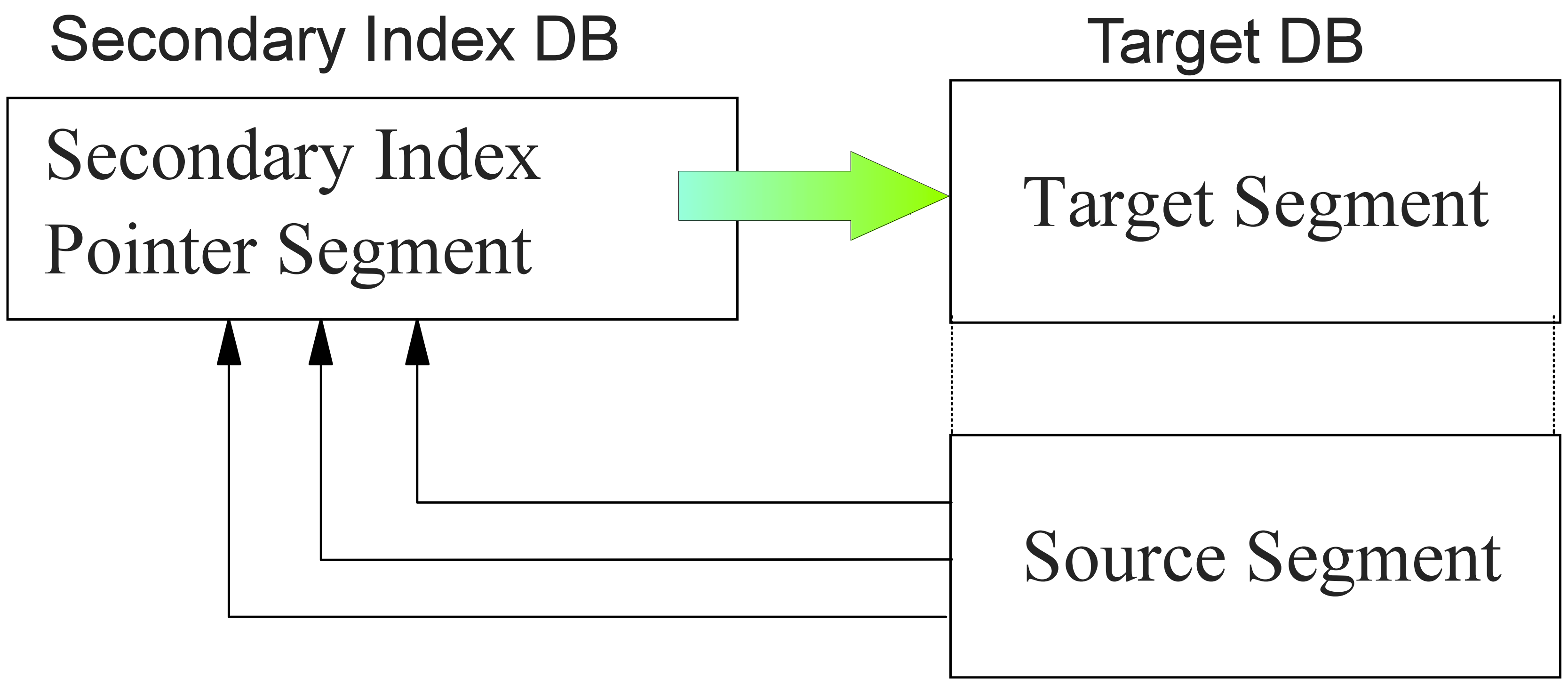
- **IMS GIM: "Secondary indexing allows you to access database records in a sequence other than that defined by the physical hierarchy."**
  - Secondary Indexing provides many benefits
  - A Secondary Index can be used:
    - To change processing sequence -
    - To provide direct access to a low-level segment
    - As a database to avoid processing the target database

➡ Can access via non-key segment

Costs

- More data sets
- More DASD
- More buffers
- More I/Os
- More processing
- More complicated recovery
- More complicated reorganization

Fundamentals



- Secondary Index DB uses KSDS (possibly also ESDS)
- Target DB may be HISAM, HIDAM, or HDAM
- Source segment may be the target segment or a dependent of the target segment

Implementation Choices

• Pointer:	Direct or symbolic
• Key:	Unique or non-unique
• Sequence:	Determines source segment
• Target:	Is segment returned to program
• Use as DB	Include duplicate data

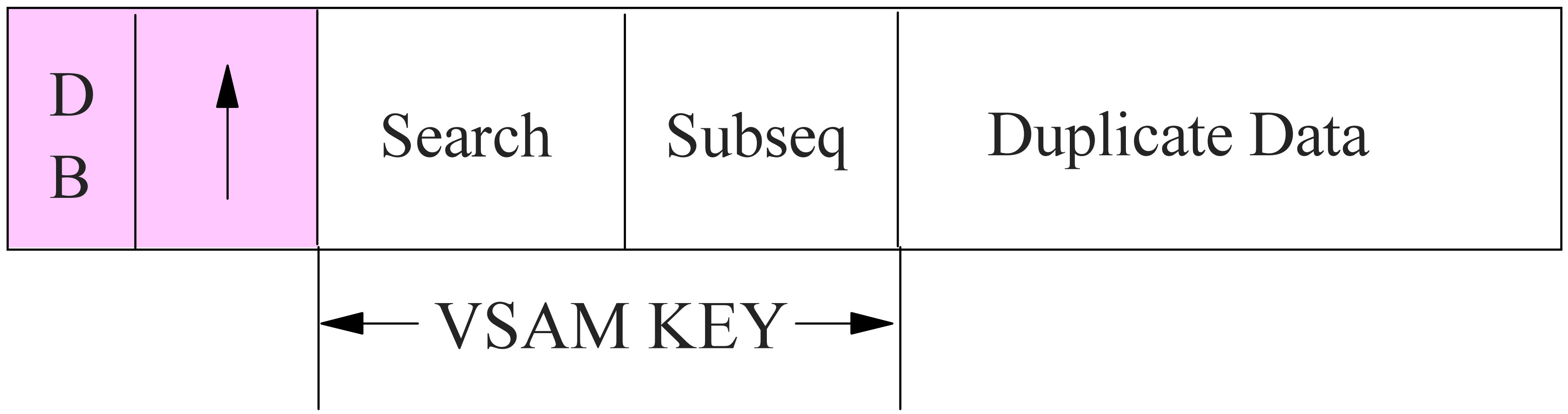
Implementation Specification

Reference: XDFLD Statement

In the target database:

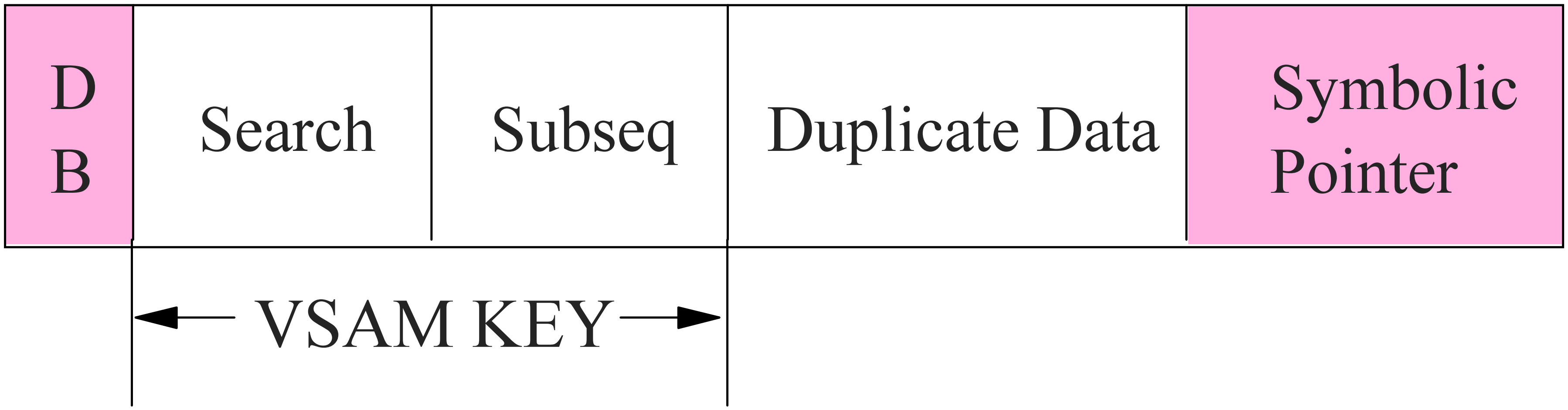
```
XDFLD   NAME=fieldname
        ,SEGMENT=segname
        ,CONST=
        ,SRCH=list
        ,SUBSEQ=list
        ,DDATA=list
        ,NULLVAL=value
        ,EXTRTN=name
```

Direct Pointer



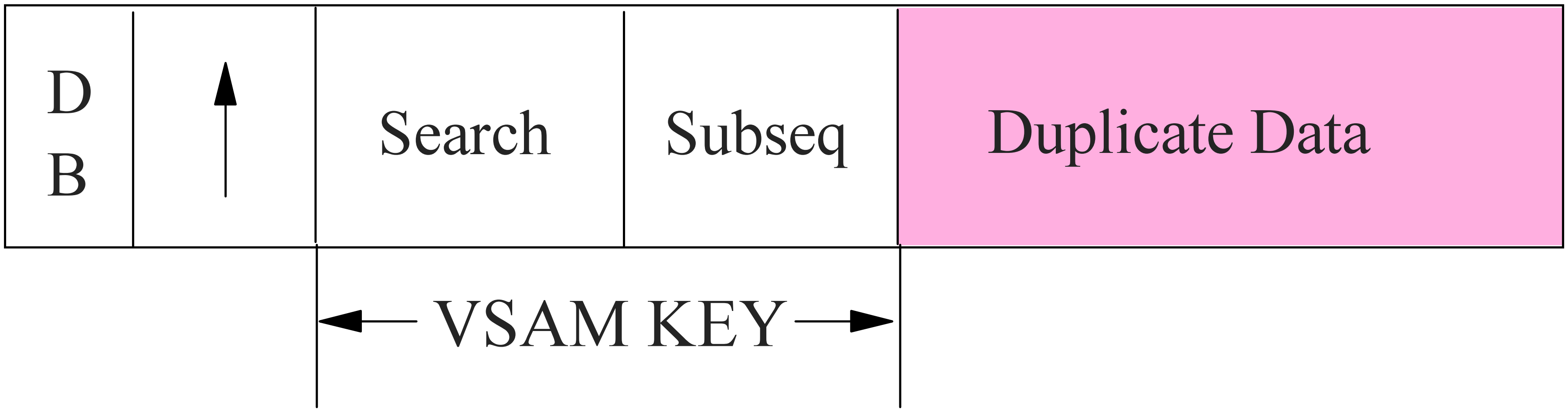
- Four byte pointer is RBA of target
- One I/O from pointer to target segment
- Must rebuild if reorganize target database
- Can use for HIDAM or HDAM target

Symbolic Pointer



- May be more I/Os to target than direct pointer
- Not necessary to rebuild if reorganize target DB
- Must use for HISAM target
- More DASD than direct pointer
- Requires unique keys from root to target

Duplicate Data



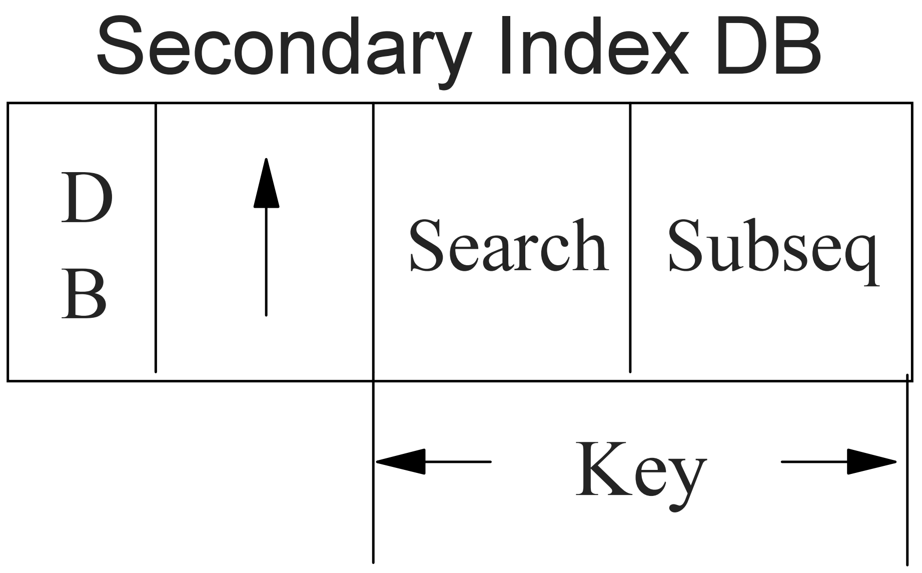
- Can only be accessed if Secondary Index is processed as a stand-alone database

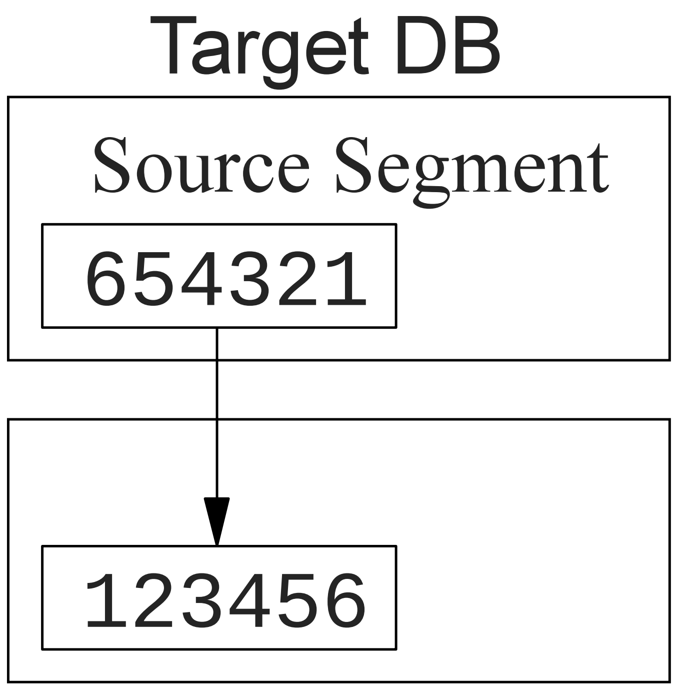
Key: Unique versus Non-unique

- **Unique Key:** **Secondary Index is single KSDS**
- **Non-unique key:** **Secondary Index requires KSDS and ESDS**
  - Duplicate keys in ESDS are chained LIFO
  - Secondary Index Reorganization reverses chain
  - Target database Reorganization rebuilds chain
- **"Nearly transparent" to programmer**
- **Key feedback area will be different**
- **Use subsequence**

```
SUBSEQ=/SXcccccc
SUBSEQ=/CKcccccc   (unique?)
```

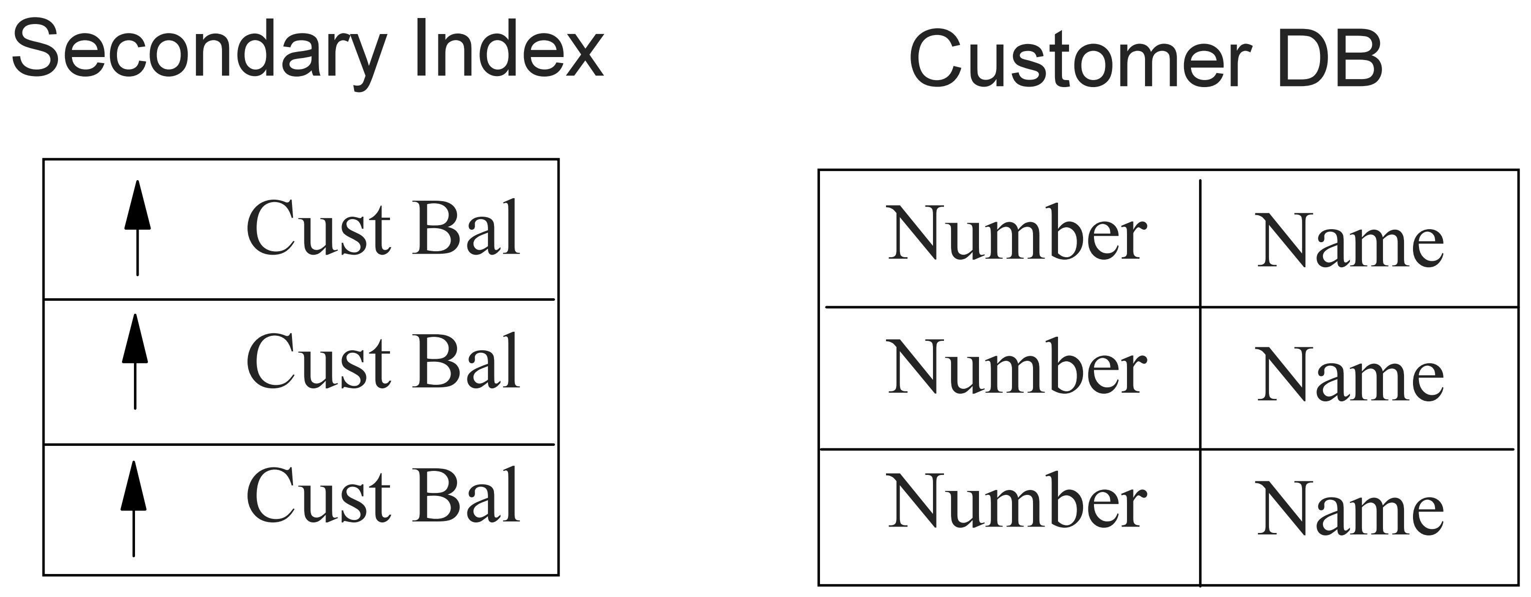
Update Concerns





- **Add pointer segment: 2-3 I/Os**
- **Change pointer segment: 4-6 I/Os**
- **Must delete and insert pointer segment**
  - Avoid volatile source segment
  - Avoid volatile search / subsequence fields

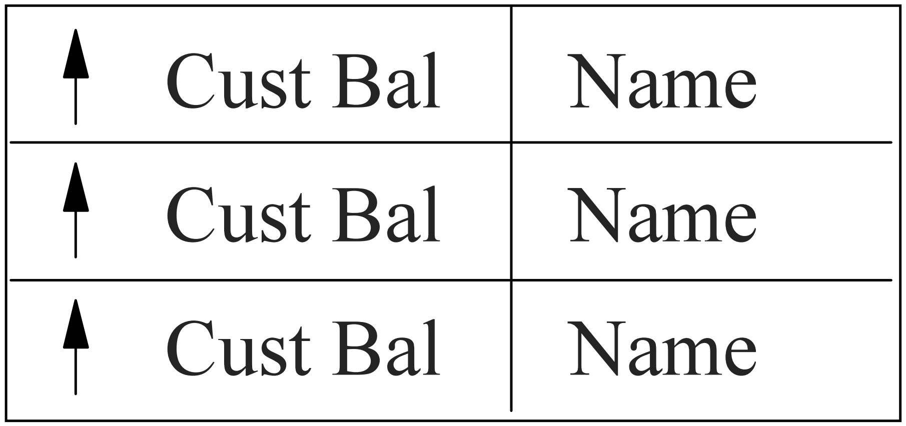
Secondary Index as a Database



- **Example: Using Secondary Index to find customer names by amount owed**

Secondary Index as a Database ...

Secondary Index



↑	Cust Bal	Name
↑	Cust Bal	Name
↑	Cust Bal	Name

- **Add Customer Name to Secondary Index**
- **Read Secondary Index as database**
  - No need to read target database
  - Duplicate data must be stored and maintained
- **Symbolic pointer may be useful**

Sparse Secondary Index

Secondary Index

↑	1111
↑	2222
↑	3333

Customer DB

Number	Bal=0000
Number	Bal=3333
Number	Bal=2222
Number	Bal=1111

- **Smaller, Faster to access**
- **Faster to build**
- **Less update activity**
- **More logic**

Secondary Index as a Sort

- **Possible to misuse Secondary Index to access:**
  - All HDAM roots in key sequence
  - All customer segments in name sequence where sequence field is customer number
  - ...
- **Do not use Secondary Index as a Sort**
  - Sort is cheaper, faster
  - Extract records in physical sequence and sort
- **Use Secondary Index for alternative access**

Using Secondary Index

- **Given Target Database:**

```
SEGM   NAME=CUSTSEGM,BYTES=...
FIELD  NAME=(CUSTNUMB,SEQ,U),BYTES=...
FIELD  NAME=CUSTNAME,BYTES=...
LCHILD NAME=(...,...),...
XDFLD  NAME=XREFNAME,SRCH=CUSTNAME,SUBSEQ=...
```

- **Call is:**

```
GU  CUSTSEGM(CUSTNAME= LINDVALL  )

1. Get pointer segment
2. Follow pointer to target
3. See if CUSTNAME=LINDVALL
4. If not equal, then goto 1.
5. If equal, then done
```

- **Call works but many I/Os**

Using Secondary Index ...

- **Given Target Database:**

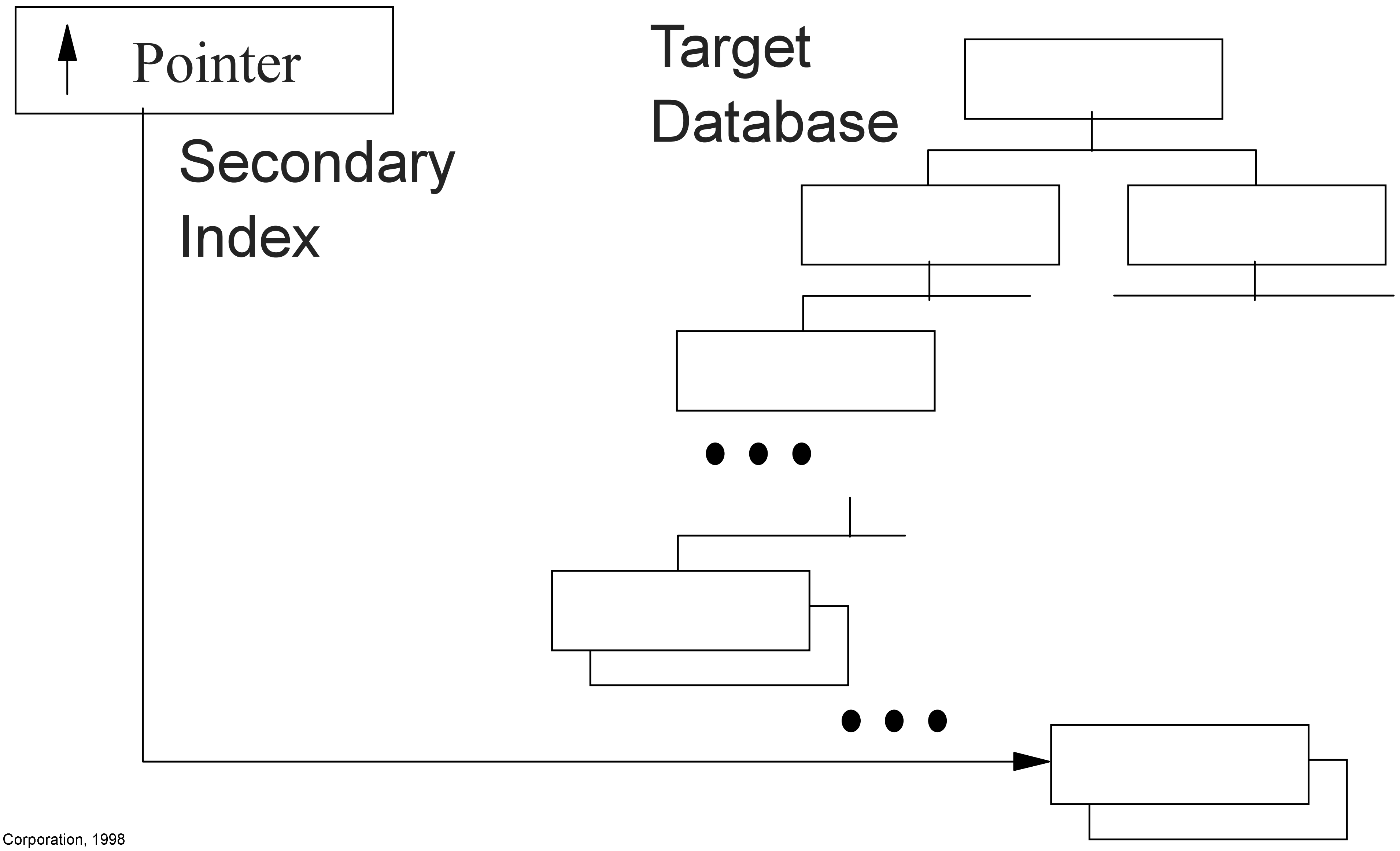
```
SEGM   NAME=CUSTSEGM,BYTES=...
FIELD  NAME=(CUSTNUMB,SEQ,U),BYTES=...
FIELD  NAME=CUSTNAME,BYTES=...
LCHILD NAME=(...,...),...
XDFLD  NAME=XREFNAME,SRCH=CUSTNAME,SUBSEQ=...
```

- **Call is:**

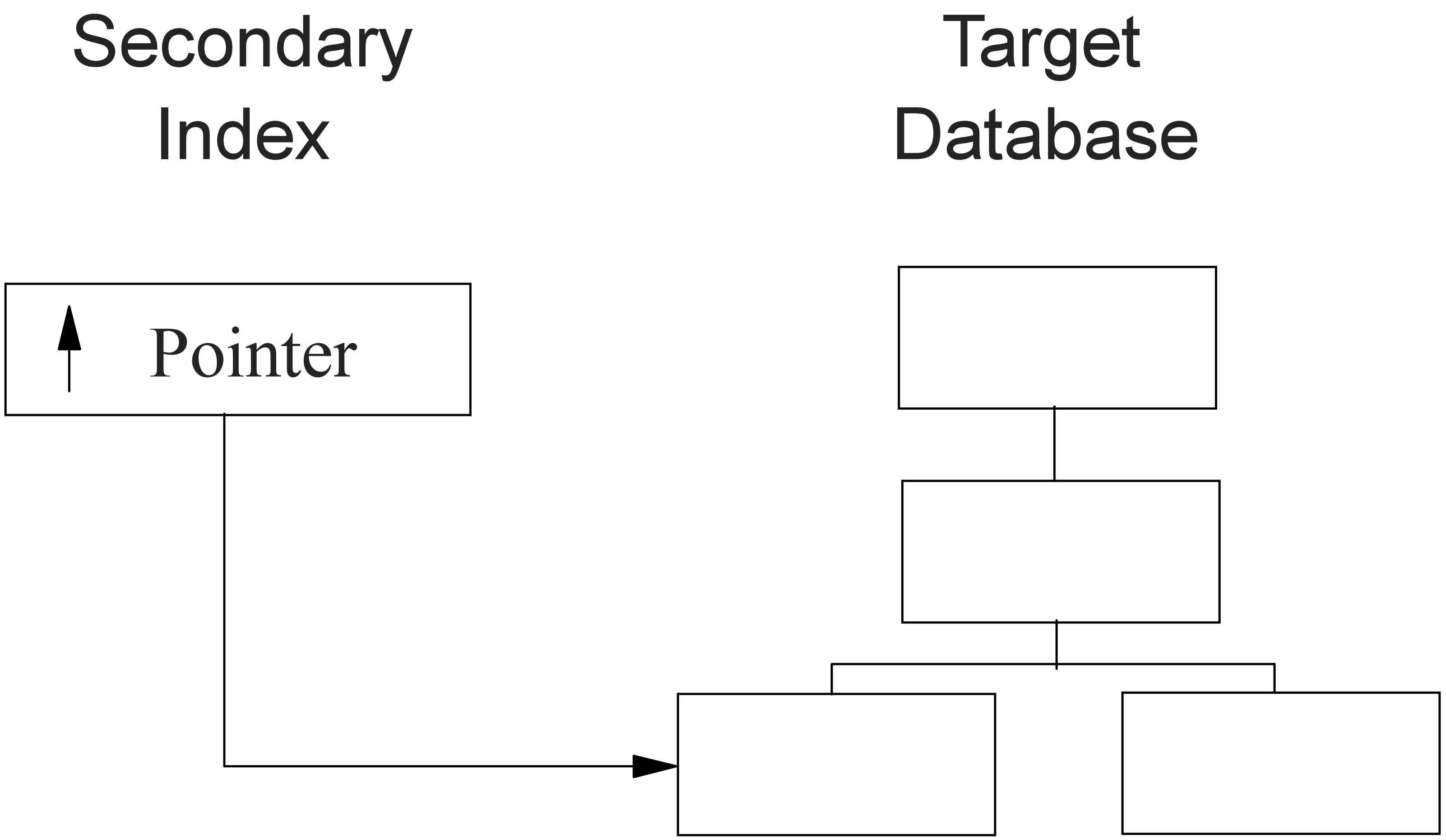
```
GU  CUSTSEGM(XREFNAME= LINDVALL  )

1. Get pointer segment
2. See if key field = LINDVALL
3. If not equal, then goto 1.
4. If equal, then done
```

- **Use correct call**

Direct Access to Low Level



Direct Access to Low Level . . .



Direct Access to Low Level ...

- **"Secondary Structure" results when target is not the root**
- **Rules: In PCB, code target segment as `....,PARENT=0`**
  - Access to root from target using Physical Parent pointers
  - Access to dependents from target segment as usual
- **Benefits: Concatenated key or target available `/CKccc`**

  Parents, root of target segment available

Shared Secondary Index

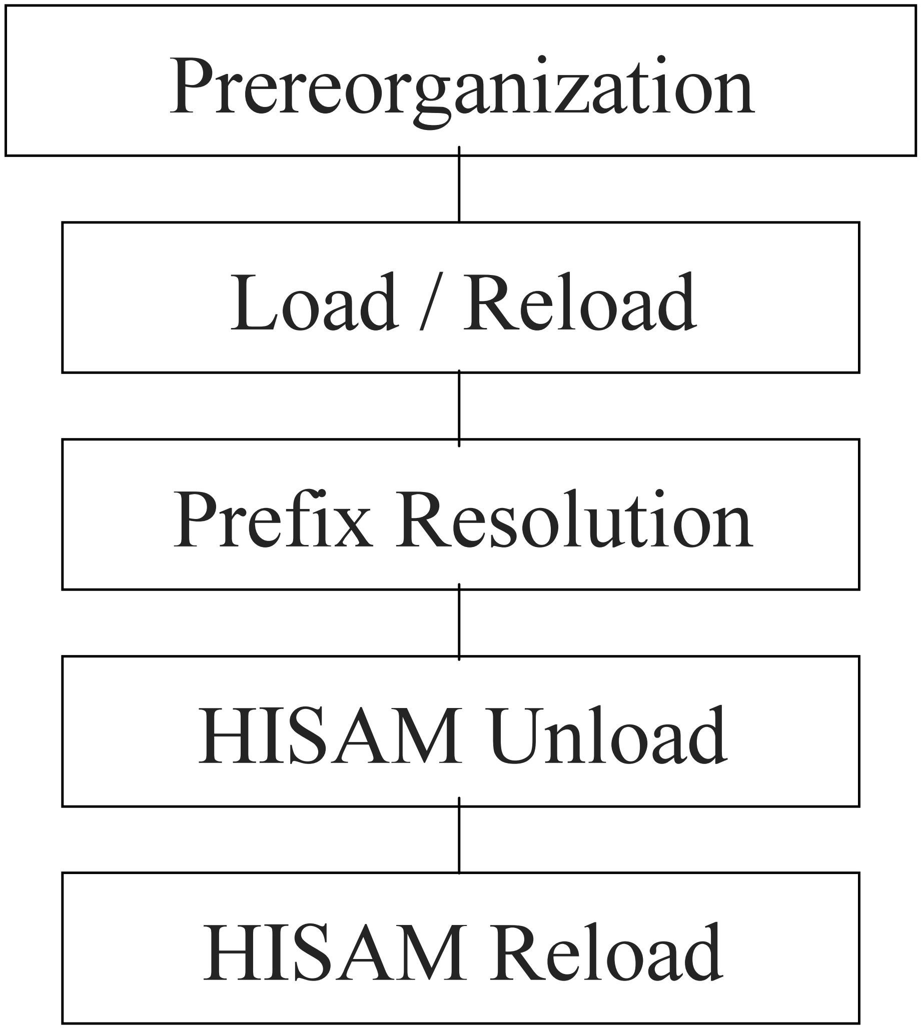
- **Do not use**
- **Concept: put multiple Secondary Indexes into single KSDS**
  - Reason lost in ancient VSAM
- **Complicated to define and create**
  - All Secondary Indexes must have same key length and offset
  - Uses one character constant in key to separate indexes
- **Easy to separate**

Reduce or Avoid Lock Conflicts

- **Use ERASE=NO in DFSVSMxx and DFSVSAMP**
  - Changes ERASE to REPLACE (delete bit)
  - REORG drops deleted records
- **Use a small DATA CISIZE**
- **Use extra CI FSPC if inserts expected**
- **Switch to Update PCB and update as late in sync interval as possible**
- **Reorg (REPRO) KSDS ASAP after a mass delete against a KSDS (or rebuild index)**

Building and Reorganizing

- **Traditional sequence of events**



Building and Reorganizing ...

- **Prereorganization** $\rightarrow$ **Builds control data set**
- **Load / Reload** $\rightarrow$ **Builds Index work data set**
  Tuning: VSAM LSR Buffer `//DFSVSAMP   DD   *`
- **Prefix Resolution** $\rightarrow$ **Sorts Index work data set**
  Tuning: Sort

Building and Reorganizing ...

- **HISAM Unload → Formats file for loading**
  - Reads work data set once for each secondary index to be loaded
  - Buffers ≥ (2 X Number of CIs per track)
  - Tuning: VSAM LSR Buffer `//DFSVSAMP DD *`
- **HISAM Reload → Loads Secondary Index**
  - Buffers ≥ (2 X Number of CIs per track)
  - Tuning: VSAM LSR Buffer `//DFSVSAMP DD *`
  - Don't use → Use Database Recovery (faster by ~ 10 : 1)

Building and Reorganizing ...

For parallel Secondary Index loading:

- Split index work file after Prefix Resolution
  - "Roll your own"
  - No DSECT / mapping

- Other alternatives

Building and Reorganizing ...

- **Reorganizing the Secondary Index itself**
  - For KSDS: Image Copy / Recover or VSAM Repro
  - Uses VSAM record read
  - Re-establishes VSAM Freespace
- **Non-Unique keys?**
  - Choice is HISAM Unload / Reload

Building and Reorganizing ...

- **Initial load of target database**
  - Use **`PROCOPT=L`**
  - Use utilities to create Secondary Index
  - Do not use **`PROCOPT=I`** to "load" target segments
    - KSDS performance will suffer
    - Expect many CI, CA splits

Building and Reorganizing ...

IBM IMS INDEX BUILDER

- Provides **fast** and flexible way to rebuild primary and secondary indices
- Easy to use
- IBM Program Product (product number 5655-E24)

Building and Reorganizing ...

IMS INDEX BUILDER...

- Rebuild all or some secondary indices of an IMS database using as input:
  - Output from initial load or reload after a reorg (DFSURWF1)
  - DL/I scan of the IMS database
  - Output from prefix resolution (DFSURIDX)
- Fully supports:
  - Empty secondary indices
  - Addition of new secondary indices
- Splits dfsurwf1 INPUT while rebuilding secondary indices
- Rebuilds HIDAM primary indices

Building and Reorganizing ...

- **IMS INDEX BUILDER...**
  - If secondary indices can be rebuilt, then register them to DBRC as **NON-RECOVERABLE**:
    - Less logging - better performance
    - Shorter Archive, Change Accumulation
    - No need to Image Copy

Building and Reorganizing ...

IMS HIGH PERFORMANCE REORG TOOLS

- High Performance Reorg Components:
  - Fast Unloading
  - Fast Reorg
  - Fast Reloading
  - Fast Prefix Resolution
  - Fast IMS Index Builder
  - Scan

Summary

- Use of Secondary Index requires more resources
- Avoid volatile source segment, search and subsequence fields
- Use correct calls (qualify on XDFLD name field)
- Do not use Secondary Index to sort
- Traditional load and reorganization utilities should be examined closely

Summary ...

- **A Secondary Index can be used:**
  - To change processing sequence
  - To provide direct access to a low level segment
  - As a database
- **All there is to know:**
  - Direct pointer
  - Unique key
  - Sparse if possible

RETURN TO INDEX