Secondary Indices and Logical Relationships: An Overview

Rod Murchison
BMC Software, Inc.

Monday, August 2 2010

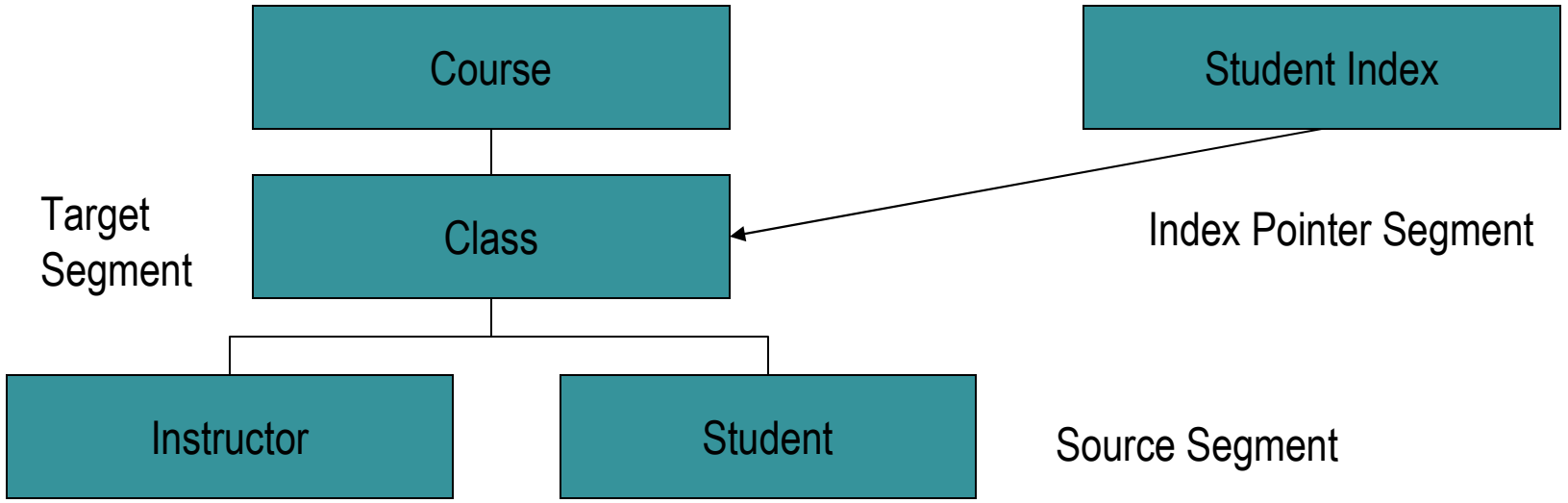
Objectives

- Explain the terminology of secondary indexing
- Understand the contents of the index segment
- See how the secondary data structure works
- Explain the terminology of logical relationships
- Understand the pointers used in logical relationships
- Know the formation of a the concatenated segment
- Understand the rules for creating logical relationships

Secondary Indices

Why Secondary Indices?

- Processing sequence other than the root key
  - Avoids scanning for a non-key field
- Direct access to lower-level segments
  - Faster processing



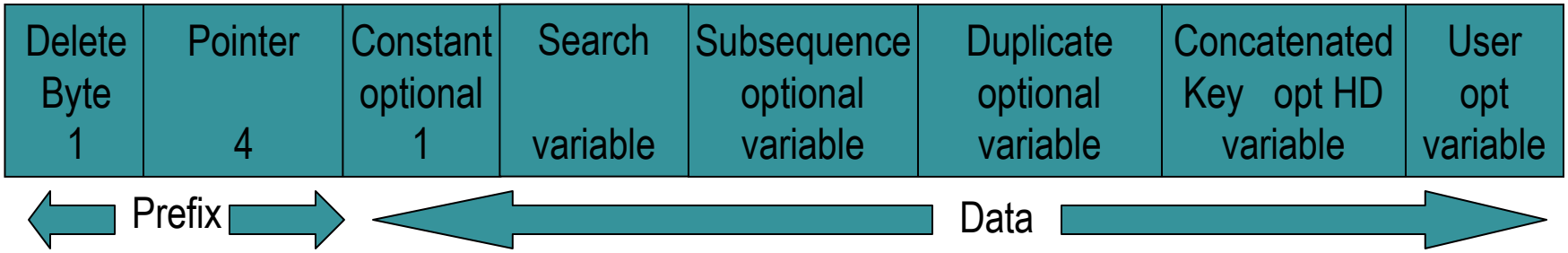
Secondary Index (SI)

- Can be based on HISAM, HIDAM, HDAM, PHIDAM, and PHDAM
- It is a separate database, using VSAM
  - Can be processed on its own
  - Essentially HISAM database
  - KSDS only if key is unique, ESDS if keys are not unique
- Uses fields from the source segment to create a key
- Access via the secondary index is to the target segment
- Source segment can be the target itself or any dependent of the target

Secondary Index (SI) cont.

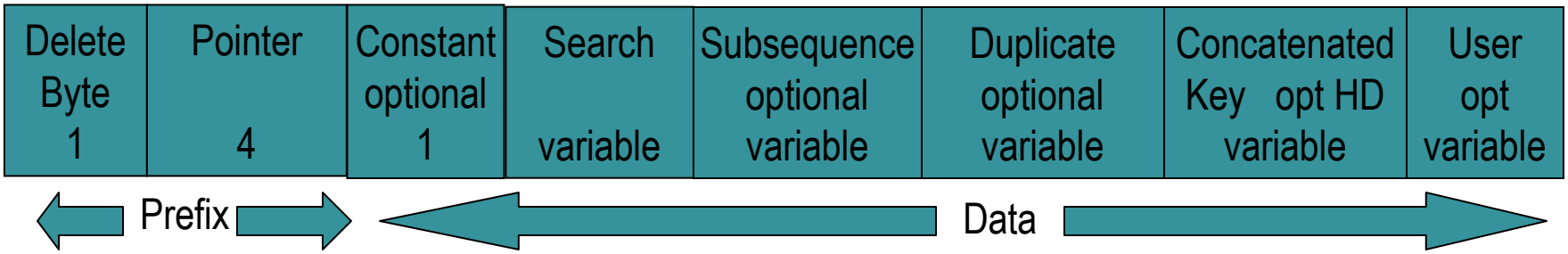
- Limitations on secondary indices
  - 32 secondary indices on one segment type
  - 1000 secondary indices for a database
- Secondary index is a special kind of logical relationship
  - The pointer goes between databases rather than within one database
- Invisible to the application
  - PROCSEQ= in the PCB tells IMS to use the secondary index for access
  - Can have PROCSEQ= and normal PCBs in the same PSB
  - Application must use the XDFLD name in the SSA
    - If it uses the field name it will cause sequential scanning

Fields in the Index Pointer



- Pointer is used when the target is in an HD database
- Constant is used for shared secondary indices
  - More than one SI in the same database
- Search field is made of 1 to 5 fields from the source segment
- Subsequence field is made of 1 to 5 fields from the source or IMS-generated values
  - The SI key is the search field and the subsequence field
  - The subsequence field is used to make the SI key unique

Fields in the Index Pointer



- Duplicate Data is 1 to 5 fields from the source segment
- User Data is anything that you want to put into the pointer segment
  - Duplicate and User Data fields are only used when processing the SI as a database
- Concatenated key is the key of the target segment. Required when target is HISAM.
  - Not allowed when target is PHDAM or PHIDAM

IMS-Generated Values

- Additional help to create unique keys for the secondary index
- Defined in the DBD for the target database
  - Under the SEGM statement for the source segment
  - FIELD NAME=/SXaaaaa
    - Field contains the 4-byte RBA of the source segment for HISAM, HDAM, HIDAM
    - Field contains the 8-byte ILK for PHDAM or PHIDAM
    - Guarantees uniqueness
    - Appears in the subsequence field of the pointer segment
  - FIELD NAME=/CKaaaaa
    - Field contains the concatenated key of the source segment
    - Can be in either the subsequence or duplicate data fields of the pointer segment
    - BYTES= and START= can be used to select a portion of the key
- These fields do not appear in the source segment itself
  - They only appear in the pointer segment

Secondary Data Structure Rules

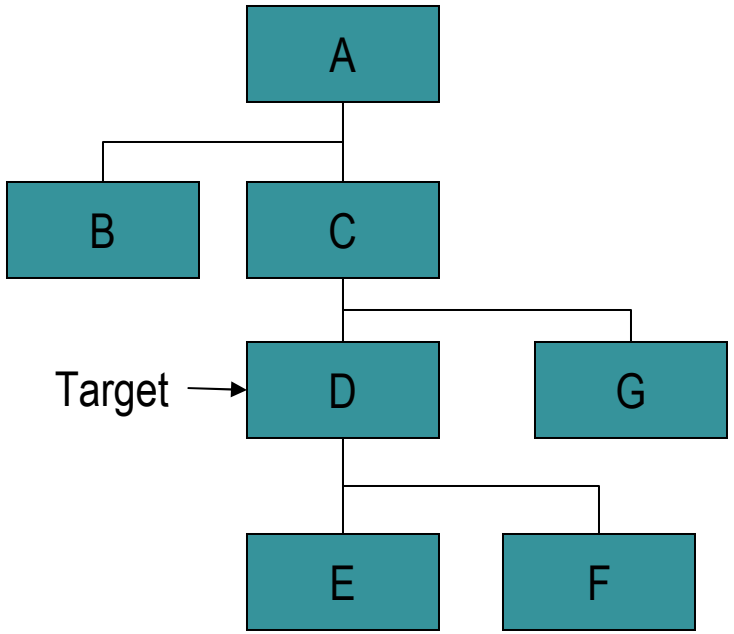
- The target segment becomes the root
- The dependents under the target remains the same
- The parent segments of the target up to the root appear inverted as the leftmost dependents of the target/root
- The dependents of those segments appear normally
  - The same segment type will appear in multiple places
  - The logical DBD must assign different names to these segments
    - Otherwise PSBGEN gives ‘SEG150’ when it tries to process duplicate SENSEGs

Secondary Data Structure Rules

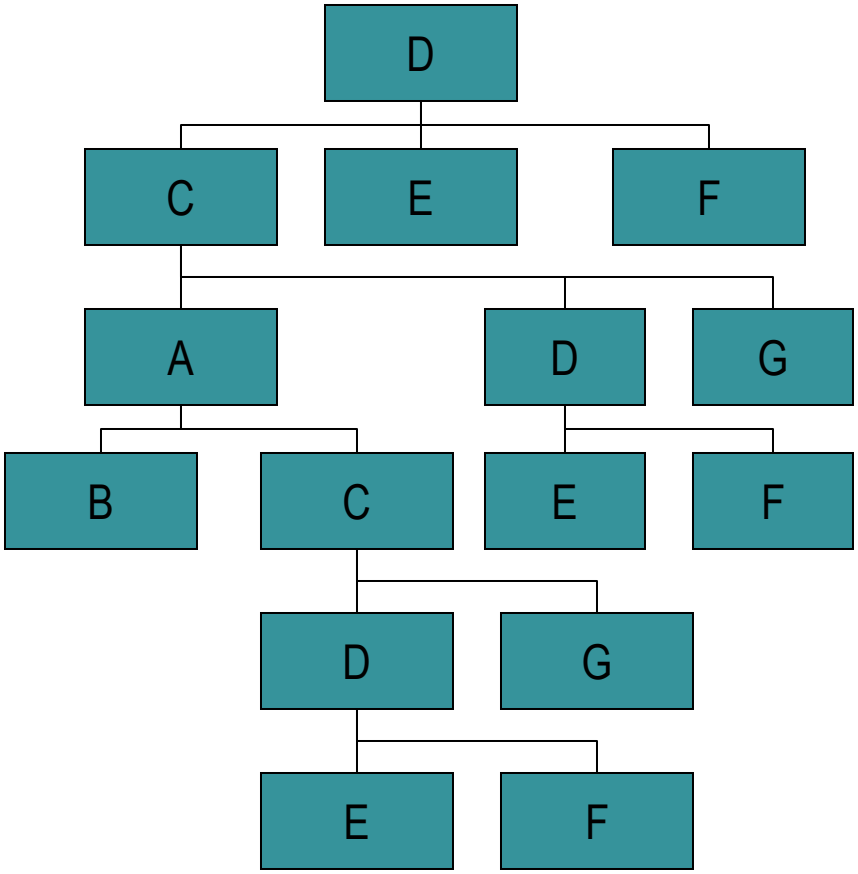
- The target segment and its parent segments up to the root may not be inserted or deleted while processing the secondary data structure
  - Would violate normal hierarchical rules
- The hierarchical sequence of the secondary data structure is called the secondary processing sequence

Secondary Data Structure

Physical Structure



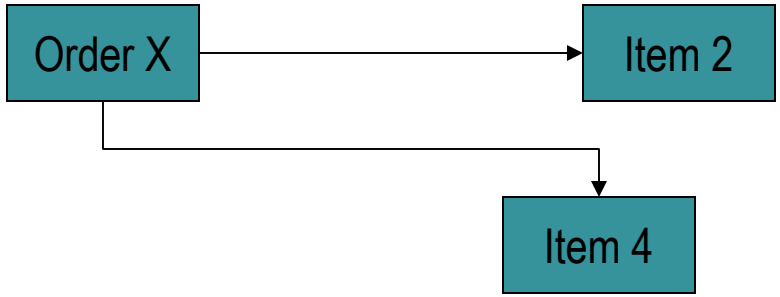
Secondary Data Structure



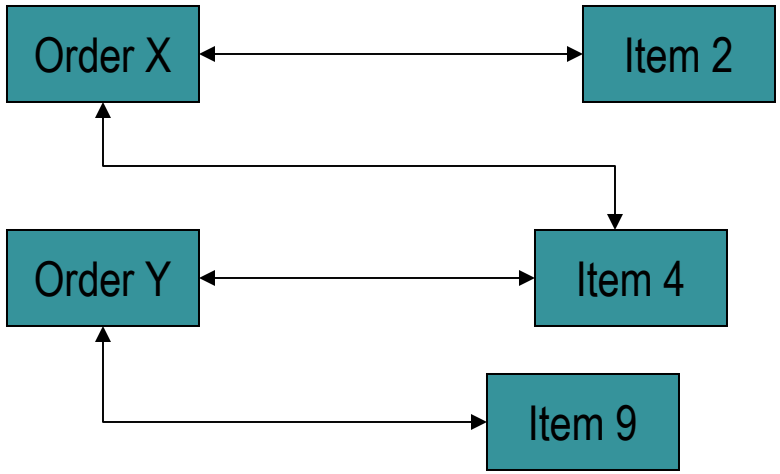
Logical Relationships

Types of Logical Relations

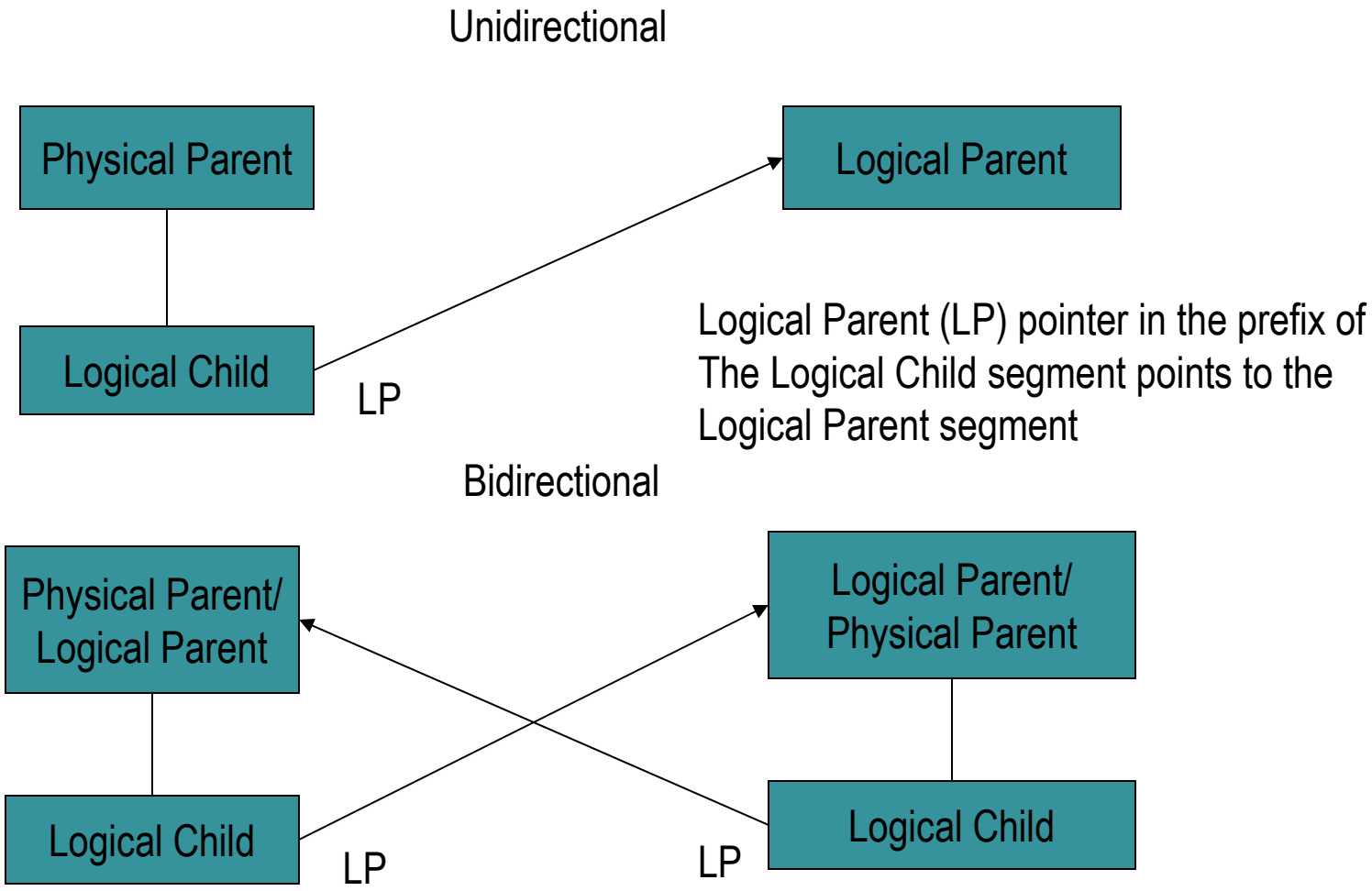
- Unidirectional
  - One way relationship from one database to another
  - Always start from the same side



- Bidirectional
  - Two way relationship between database records
  - Can start on either side
  - IMS maintains both sides of the bidirectional relationship



How They are Implemented



Bidirectional Physical Pairing

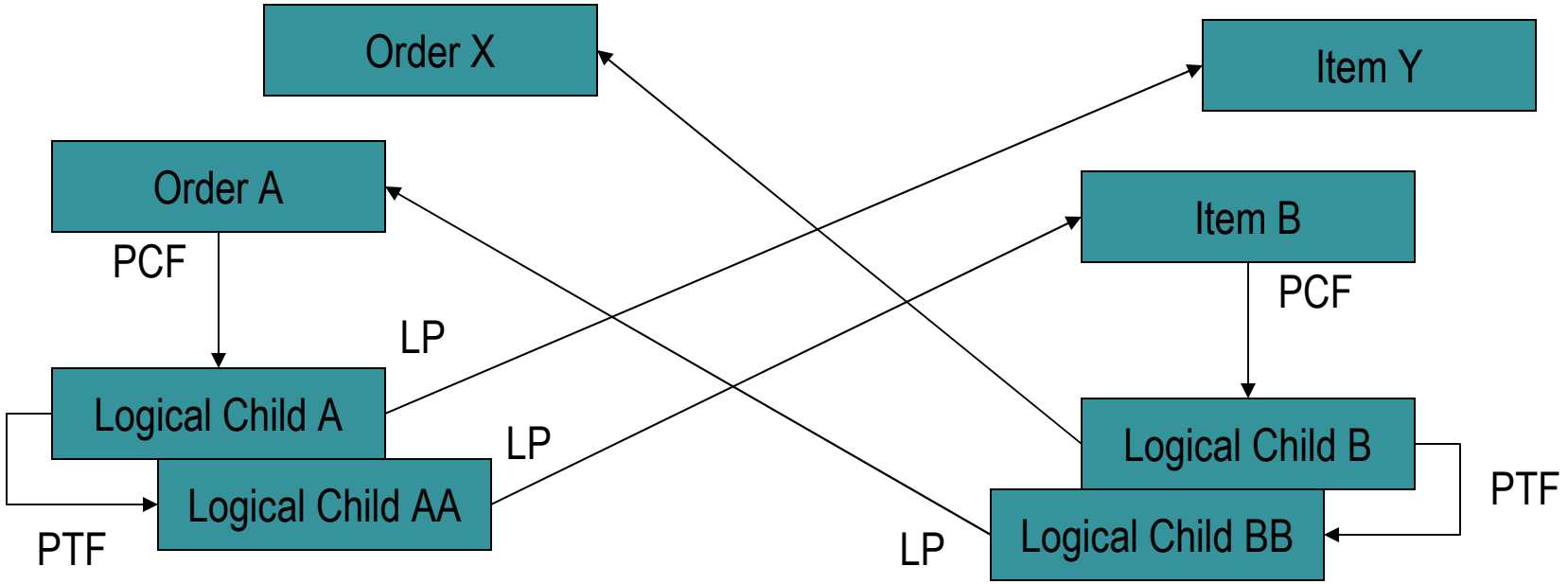
Physical Child and Physical Twin or Hierarchic pointers link PP to LC

Logical Parent pointer links the LC to the LP

Requires a physical segment on each side of the relation

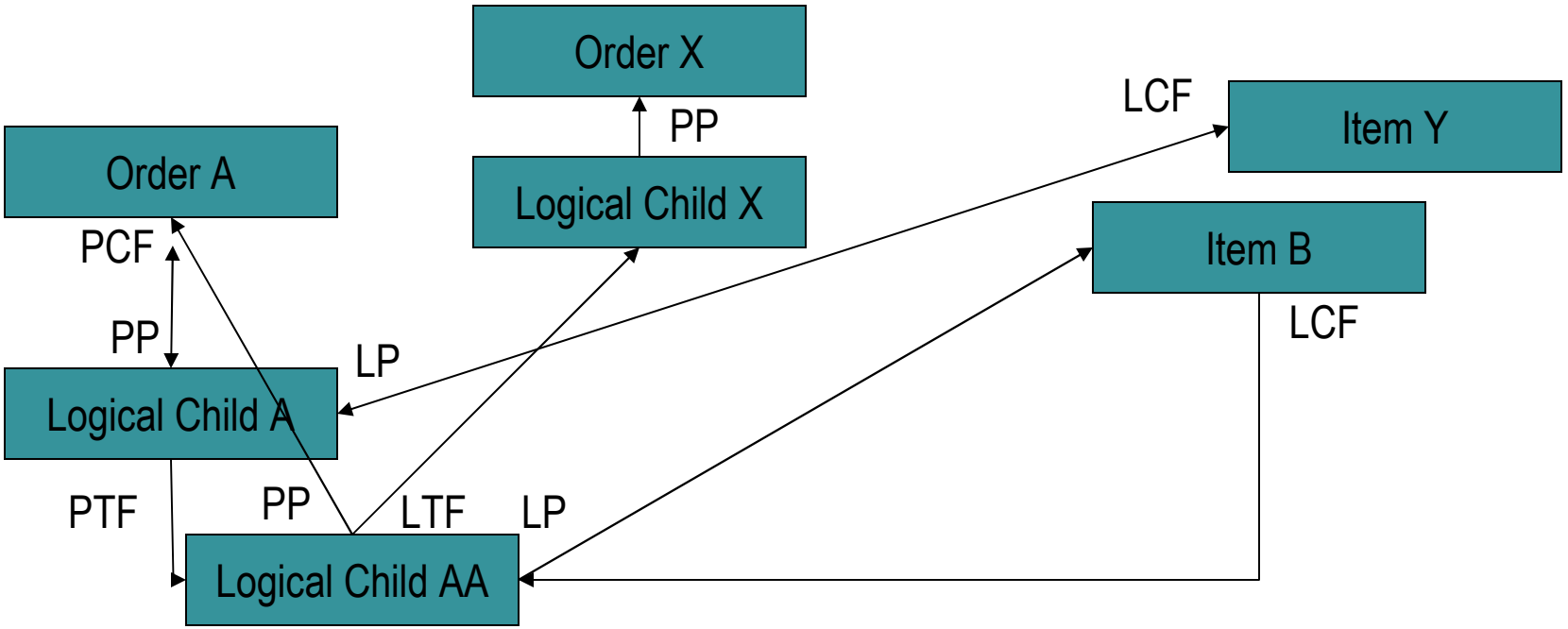
When one side is inserted, deleted or replaced, IMS does the same on the other side

The only way to implement a logical relationship with PHDAM or PHIDAM



Bidirectional Virtual Pairing

- Logical Child First (LCF) pointer replaces PCF on the virtual side
- Logical Twin Forward (LTF) replaces PTF and links Logical Twins in different records
- Physical Parent (PP) pointer replaces LP for access to parent from Logical Child
- Physical segment only exists on one side of the relation
- Real Logical Child must be in an HD database
- Not allowed with HALDB databases



Prefixes with Logical Relationshsips

Logical Child Prefix

PP, LTF, and LTB are only present if virtual pairing

SC	DB	HF	HB	PP	LTF	LTB	LP

OR

SC	DB	PTF	PTB	PP	LTF	LTB	LP	PCF	PCL
							PCF	PCL	EPS

Logical Parent Prefix

PP pointer is only present if a lower level segment is a logical parent

SC	DB	HF	HB	PP	LCF	LCL

OR

SC	DB	PTF	PTB	PP	PCF	PCL	LCF	LCL

The Logical Child Segment

SC	DB	Pointer Area	Logical Parent's Concatenated Key	Fixed Intersection Data

- Logical Parent's Concatenated Key
  - Sequence field of all segments from the root to the logical parent
  - Always appears to the application program
  - May or may not be physically stored in the Logical Child
    - If not stored, then IMS will generate it when it retrieves the LP
- Logical Parent Pointer
  - The LPCK if it is physically stored
    - Must be used if the LP is in an HISAM database
    - This is called a symbolic pointer
  - A pointer in the LC segment prefix
    - May only be used if the LP is in an HD database
    - It is the only kind of pointer that can exist in an HISAM prefix
- Fixed Intersection Data
  - Data that is dependent on the relationship between the segments
  - Maintained on both sides of a bidirectional relationship
  - Variable Intersection data is in the dependents of the LC

The Concatenated Segment

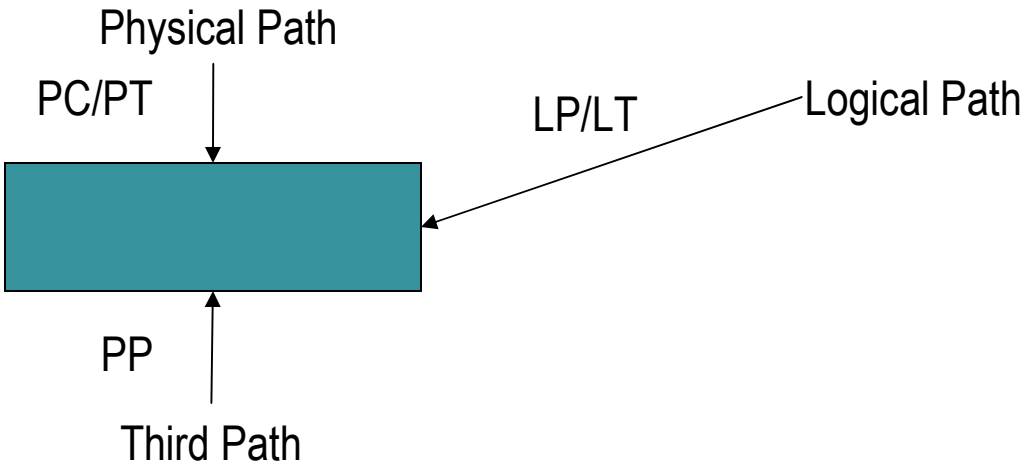
- Physical Database
  - Separate definitions for PP, LC and LP
  - Pointer usage is defined
- Logical Database (ACCESS=LOGICAL)
  - Used by applications tp process with the logical relationships
  - Concatenated segment is what the application sees
    - Must be defined in the Logical Database

The dependent structure under the concatenated segment is a secondary data structure just like what would be created when the concatenated segment Is the target of a secondary index.

The Rules

- Logical Children
  - Must have both a physical and a logical parent
  - Only allowed one of each
  - It must be a dependent so the root cannot be a logical child
  - May not have an immediate dependent that is a logical child
  - It may have physical children. If physical pairing, only one of the paired segments may have children
- Logical Parents
  - A logical parent may be at any level, including the root
  - It may have one or more logical children
  - It may not also be a logical child
- Physical Parents
  - May not be logical children

Three Paths



- Delete Problems
  - Cannot delete a segment while there are still logical children pointing to it
    - If we have LCF pointers, cannot delete until all those pointers are zero
    - If a segment does not have LCF pointers, then IMS puts a counter in the prefix
  - A segment has to be both physically deleted and logically deleted
    - Physical deletion means that it cannot be accessed via physical pointers
    - Logical deletion means that it cannot be accessed via logical pointers
    - Can still be reached by the third path

Rod Murchison

BMC Software, Inc.

rod_murchison@bmc.com

(208) 462-2599