Converting IMS Logically Related Database DBDs to HALDB

Rich Lewis
IMS Advanced Technical Support
IBM Americas

May 2006

Converting IMS Logically Related Database DBDs to HALDB

When you convert databases to HALDB (High Availability Large Database) you must change their DBDs. This includes changing the ACCESS parameter on the DBD statement to specify a HALDB type and deleting the DATASET statements. If the databases have logical relationships additional changes may be required. This paper explains the requirements for converting DBDs for databases with logical relationships to HALDB.

Additional information on migrating databases to HALDB is available in *Migrating to IMS HALDB (High Availability Large Database)* at

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS693.

The *IMS Utilities Reference: System* manuals contain detailed information on coding the parameters on the DBD statements. The manuals are:

IMS Version 8 Utilities Reference: System, SC27-1309
IMS Version 9 Utilities Reference: System, SC18-7834

The information on coding the PARENT parameter on the SEGM statement is incomplete and misleading in these manuals. An expanded and corrected explanation of the PARENT parameter is included under "Addendum to the IMS V8 and V9 Utilities Reference: System manuals" on page 12.

HALDB Logical Relationship Support

HALDB has two restrictions for logical relationships that do not exist for non-HALDB databases. First, it does not support virtual pairing for bidirectional logical relationships. Physical pairing is required for bidirectional logical relationships. Second, HALDB does not support symbolic pointers. Direct pointers are required.

Types of Logical Relationships

HALDB does not support virtual pairing for bidirectional logical relationships. Bidirectional logical relationships must use physical pairing. HALDB supports unidirectional logical relationships and bidirectional logical relationships with physical pairing.

When you migrate databases to HALDB you should make the following migrations of your logical relationships:

- Unidirectional to unidirectional (no change)
- Bidirectional with physical pairing to bidirectional with physical pairing (no change)
- Bidirectional with virtual pairing to bidirectional with physical pairing

Virtual Pairing

When virtual pairing is used one of the two logical children does not physically exist. Instead, there is a logical child pointer in the parent of the virtual logical child to a physical logical child.

Figure 1 illustrates virtual pairing. The SKILNAME and NAMESKIL segments are paired.

NAMESKIL is virtual. It does not actually exist, but programs can access the relationship from NAMEMAST to SKILMAST. IMS does this by using the logical child first (LCF) pointer from NAMEMAST to SKILNAME. It may then follow logical twin (LT) pointers to find the correct SKILNAME segment. These segments and the LT pointers are not shown. IMS then uses the physical parent (PP) pointer from SKILNAME to SKILMAST.

SKILNAME is physical. When programs access the relationship from SKILMAST to NAMEMAST, IMS uses the physical child first (PCF) pointer and physical twin (PT) pointers to find the correct instance of SKILNAME. The PT pointers and the other instances of SKILNAME are not shown. It then follows the logical parent (LP) pointer from SKILNAME to NAMEMAST.

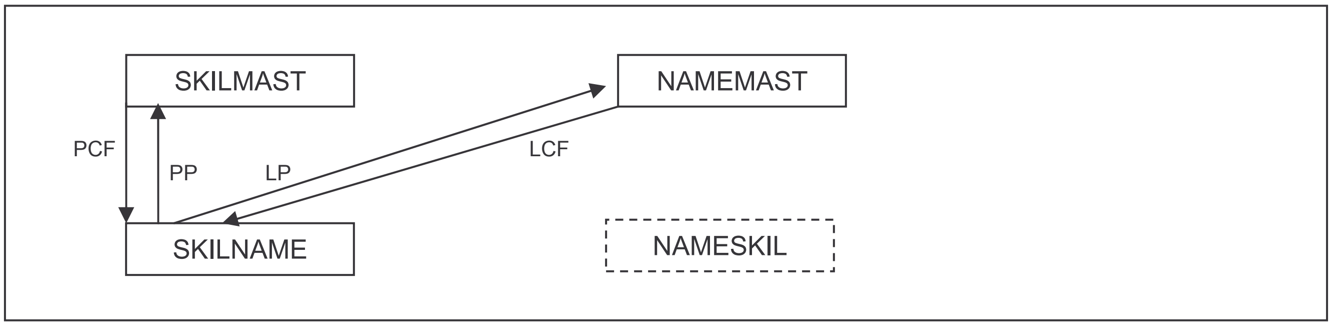

Figure 1: Bidirectional Logical Relationship with Virtual Pairing

Physical Pairing

When physical pairing is used two logical children physically exist.

Figure 2 illustrates physical pairing. The SKILNAME and NAMESKIL segments are paired.

NAMESKIL and SKILNAME are physical. When programs access the relationship from SKILMAST to NAMEMAST, IMS uses the physical child first (PCF) pointer and physical twin (PT) pointers to find the correct instance of SKILNAME. The PT pointers and the other instances of SKILNAME are not shown. It then follows the logical parent (LP) pointer from SKILNAME to NAMEMAST. Similarly, when programs access the relationship from NAMEMAST to SKILMAST, IMS uses the physical child first (PCF) pointer and physical twin (PT) pointers to find the correct instance of NAMESKIL. The PT pointers and the other instances of NAMESKIL are not shown. It then follows the logical parent (LP) pointer from NAMESKIL to SKILMAST.

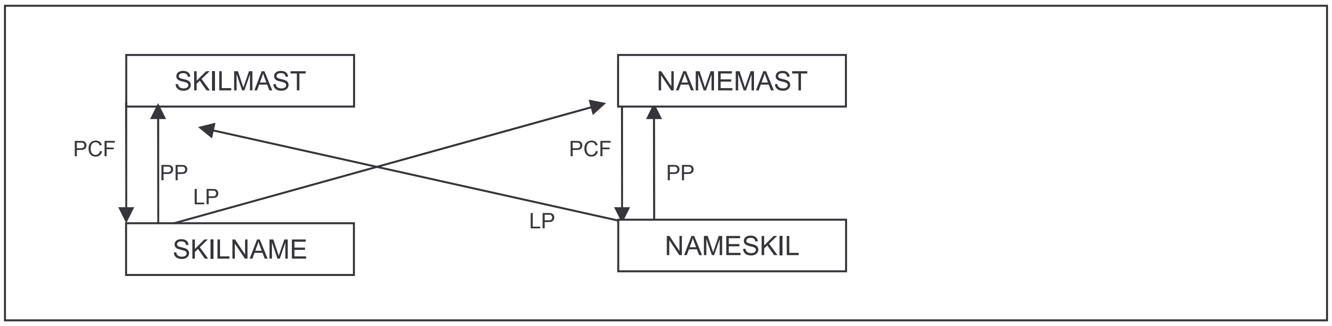

Figure 2: Bidirectional Logical Relationship with Physical Pairing

Virtual Pairing vs. Physical Pairing

When virtual pairing is used, one of the logical children does not physically exist. IMS uses logical child pointers and logical twin pointers when navigating through the relationship that has the virtual logical child. The physical parent pointer is used to navigate from the logical child to its parent. When physical pairing is used, both logical children exist. There are no logical child pointers or logical twin pointers. IMS always uses physical child pointers and physical twin pointers when navigating logical relationships. The logical parent pointer is used to navigate to the logical parent.

Changing Virtual Pairing to Physical Pairing in DBDs

When you migrate a databases with virtual pairing to HALDB the logical relationship must be converted to physical pairing. The following provides guidance and instructions for making the conversion.

Recognizing Virtual Pairing

You may determine if you have virtual pairing by examining your DBDs. If your HDAM or HIDAM database contains a SEGM statement with a SOURCE parameter, this segment is a virtual logical child. The first subparameter of SOURCE identifies the real logical child. The third subparameter identifies the database in which the real logical child resides. In Example 1 the NAMESKIL segment in the DBD for the PAYROLDB database contains a SOURCE parameter. It is a virtual logical child. The source parameter references the SKILNAME segment in the SKILLINV database. SKILNAME is a real logical child.

```
         DBD      NAME=PAYROLDB,ACCESS=(HDAM,OSAM),                       X
                  RMNAME=(DFSHDC40,8,50000,1000)
         DATASET  DD1=PAYHDAM,BLOCK=4096,SCAN=0
         SEGM     NAME=NAMEMAST,PTR=TWINBWD,RULES=(VVV),                  X
                  BYTES=150
         LCHILD   NAME=(SKILNAME,SKILLINV),PAIR=NAMESKIL,PTR=DBLE
         FIELD    NAME=(EMPLOYEE,SEQ,U),BYTES=60,START=1
         FIELD    NAME=MANNBR,BYTES=15,START=61
         FIELD    NAME=ADDR,BYTES=75,START=76
         SEGM     NAME=NAMESKIL,                                          X
                  PARENT=NAMEMAST,                                        X
                  PTR=PAIRED,                                             X
                  SOURCE=((SKILNAME,DATA,SKILLINV))
         FIELD    NAME=TYPE,BYTES=21,START=1
         FIELD    NAME=STDLEVL,BYTES=20,START=22
         DBDGEN
         FINISH
         END
```

Example 1: PAYROLDB HDAM Database Which Contains the Virtual Logical Child

Converting Virtual Pairing to Physical Pairing

The following contains instructions for converting bidirectional logical relationships with virtual pairing to bidirectional logical relationships with physical pairing. For illustration it uses the HDAM DBDs shown in examples 2 and 3 and the corresponding PHDAM DBDs shown in examples 4 and 5.

To convert a virtually paired logical relationship to physical pairing you must make changes in four DBD macros. You must add an LCHILD statement that references the new real logical child. You must modify the LCHILD statement that references the old real logical child. Your must modify SEGM statements for both logical children. The following explains these changes.

LCHILD for New Real Logical Child

Add an LCHILD statement under the parent of the segment which was the old real logical child. In example 3 this is the SEGM statement with NAME=SKILMAST.

1. The new LCHILD statement must have a NAME parameter which specifies the segment which had the SOURCE parameter. In the example this is the NAMESKIL segment in the PAYROLDB database.

2. The PAIR parameter must specify the segment which was specified in the SOURCE parameter. This was the real logical child.

After making these changes the LCHILD statement in example 5 is:

```
LCHILD  NAME=(NAMESKIL,PAYROLDB),PAIR=SKILNAME
```

LCHILD Statement for Old Real Logical Child

Make the following changes in the LCHILD statement that references the segment which was the real logical child. In example 2 this is the LCHILD statement in the PAYROLDB database which includes NAME=(SKILNAME,SKILLINV).

1. Delete the PTR parameter or specify PTR=NONE. The PTR parameter is used to specify the logical child pointers. They are not used with physical pairing.

2. If there is a RULES parameter, delete it. RULES is used on the LCHILD statement only with virtual pairing. It is used to control logical twin sequencing when there is no sequence field or nonunique sequence fields. Physical pairing does not use logical twin pointers; however, you may want to specify a corresponding RULES value in the SEGM statement for the old virtual logical child in order to sequence the new physical logical child segments.

After making these changes the LCHILD segment in example 4 is:

```
LCHILD  NAME=(SKILNAME,SKILLINV),PAIR=NAMESKIL
```

SEGM Statement for the Old Virtual Logical Child

Make the following changes in the segment which was the virtual logical child. In example 2 this is the SEGM statement with NAME=NAMESKIL.

1. Add a BYTES parameter. The BYTES parameter defines the size of the segment. It includes the logical parent's concatenated key (LPCK) and the fixed intersection data. The length of the fixed intersection data in the paired logical children must be the same. The BYTES value for this segment may be calculated by taking the BYTES value from the paired logical child's SEGM statement, adding the LPCK size for this segment and subtracting the LPCK size for the paired logical child. In examples 2 and 3, the paired logical child for the NAMESKIL segment is the SKILNAME segment. Its size is 80 bytes. The LPCK for the NAMESKIL segment is the SKTYPE field in the SKILMAST segment. Its size is 21 bytes. The LPCK for the SKILNAME segment is the EMPLOYEE field in the NAMEMAST segment. Its size is 60 bytes. The BYTES parameter for the NAMESKIL segment should be 41 which is 80 + 21 - 60.

2. Delete the SOURCE parameter. The SOURCE parameter is used to define a virtual logical child by specifying its paired logical segment. This is not used for physical pairing.

3. Change the PARENT parameter so that it also references its logical parent. In example 3 this is the SKILMAST segment. The PARENT parameter should also include the PHYSICAL (P) keyword. PHYSICAL specifies that the logical parent's concatenated key (LPCK) is physically stored in the segment. The VIRTUAL (V) keyword specifies that the LPCK is not stored in the segment. HALDB always stores the LPCK in the segment. If you specify VIRTUAL or V, a warning message is issued and P is used. To avoid the warning message, you should specify PHYSICAL or P.

4. Change the PTR or POINTER parameter specifications. Add a specification for physical twin pointers. This should be TWIN (T), TWINBWD (TB), or NOTWIN (NT). Do not use NOTWIN unless you will have a maximum of only one instance of this segment type under a parent. LPARNT and PAIRED are defaults for HALDB logical child segments. LPARNT is used to indicate a logical parent pointer is used. PAIRED is used to indicate that a bidirectional logical relationship is used. You may either specify LPARNT and PAIRED or allow them to default.

5. Consider adding a RULES parameter. The first three values are the insert, delete, and replace rules for this segment. The final subparameter is either FIRST, LAST, or HERE. It is used only for segments which do not have a sequence field or which have a non-unique sequence field. It determines the placement of inserted segments in the twin chain. The example SEGM does not have a unique sequence field. A value of FIRST is specified for the final subparameter.

After making these changes the SEGM statement in example 4 is:

```
SEGM    NAME=NAMESKIL,                                   X
        PARENT=((NAMEMAST),(SKILMAST,P,SKILLINV)),       X
        BYTES=41,                                        X
        PTR=(TWIN,LPARNT,PAIRED),                        X
        RULES=(VVV,FIRST)
```

SEGM Statement for the Old Real Logical Child

Make the following changes in the segment which was the real logical child. In example 3 this is the SEGM statement with NAME=SKILNAME.

1. If VIRTUAL (V) was specified after the logical parent name in the PARENT parameter, change this to PHYSICAL (P). P specifies that the logical parent's concatenated key (LPCK) is stored in the segment. Even if V is specified, HALDB will use P. If you specify V, a warning message is issued and P is used. To avoid the warning message, you should specify P. The V in example 3 is changed to P in example 5.

2. Change the PTR parameter specifications. Delete LTWIN (LT) or LTWINBWD (LTB) keywords, if they exist. These keywords are used to specify logical twin pointers. Logical twin pointers are not used with physical pairing. LPARNT and PAIRED are required, but they are default values with HALDB. You may either specify LPARNT and PAIRED or allow them to default.

After making these changes the SEGM statement in example 5 is:

```
SEGM    NAME=SKILNAME,                                   X
        PARENT=((SKILMAST,DBLE),(NAMEMAST,P,PAYROLDB)),  X
        BYTES=80,                                        X
        PTR=(TWINBWD,LPARNT,PAIRED),                     X
        RULES=(VVV)
```

Non-HALDB Database DBD Examples

These are the example DBDs before they are converted to HALDB.

```
         DBD      NAME=PAYROLDB,ACCESS=(HDAM,OSAM),                        X
                  RMNAME=(DFSHDC40,8,50000,1000)
         DATASET  DD1=PAYHDAM,BLOCK=4096,SCAN=0
         SEGM     NAME=NAMEMAST,PTR=TWINBWD,RULES=(VVV),                   X
                  BYTES=150
         LCHILD   NAME=(SKILNAME,SKILLINV),PAIR=NAMESKIL,PTR=DBLE
         FIELD    NAME=(EMPLOYEE,SEQ,U),BYTES=60,START=1
         FIELD    NAME=MANNBR,BYTES=15,START=61
         FIELD    NAME=ADDR,BYTES=75,START=76
         SEGM     NAME=NAMESKIL,                                           X
                  PARENT=NAMEMAST,                                         X
                  PTR=PAIRED,                                              X
                  SOURCE=((SKILNAME,DATA,SKILLINV))
         FIELD    NAME=TYPE,BYTES=21,START=1
         FIELD    NAME=STDLEVL,BYTES=20,START=22
         DBDGEN
         FINISH
         END
```

Example 2: PAYROLDB HDAM Database Which Contains the Virtual Logical Child

```
         DBD      NAME=SKILLINV,ACCESS=(HDAM,OSAM),                        X
                  RMNAME=(DFSHDC40,12,50000,824)
         DATASET  DD1=SKILHDAM,BLOCK=4096,SCAN=0
         SEGM     NAME=SKILMAST,BYTES=31,PTR=TWINBWD
         FIELD    NAME=(SKTYPE,SEQ,U),BYTES=21,START=1
         FIELD    NAME=STDCODE,BYTES=10,START=22
         SEGM     NAME=SKILNAME,                                           X
                  PARENT=((SKILMAST,DBLE),(NAMEMAST,V,PAYROLDB)),          X
                  BYTES=80,                                                X
                  PTR=(LPARNT,LTWINBWD,TWINBWD),                           X
                  RULES=(VVV,LAST)
         FIELD    NAME=(STDLEVL,SEQ),BYTES=20,START=61
         FIELD    NAME=EMPLOYEE,START=1,BYTES=60
         DBDGEN
         FINISH
         END
```

Example 3: SKILLINV HDAM Database Which Contains the Real Logical Child

HALDB Database DBD Examples

These are the example DBDs after they have been converted to HALDB.

```
        DBD     NAME=PAYROLDB,ACCESS=(PHDAM,OSAM),                      X
                RMNAME=(DFSHDC40,8,10000,1000)
        SEGM    NAME=NAMEMAST,PTR=TWINBWD,RULES=(VVV),                  X
                BYTES=150
        LCHILD  NAME=(SKILNAME,SKILLINV),PAIR=NAMESKIL
        FIELD   NAME=(EMPLOYEE,SEQ,U),BYTES=60,START=1
        FIELD   NAME=MANNBR,BYTES=15,START=61
        FIELD   NAME=ADDR,BYTES=75,START=76
        SEGM    NAME=NAMESKIL,                                          X
                PARENT=((NAMEMAST),(SKILMAST,P,SKILLINV)),              X
                BYTES=41,                                               X
                PTR=(TWIN,LPARNT,PAIRED),                               X
                RULES=(VVV,FIRST)
        FIELD   NAME=TYPE,BYTES=21,START=1
        FIELD   NAME=STDLEVL,BYTES=20,START=22
        DBDGEN
        FINISH
        END
```

Example 4: PAYROLDB Database After Conversion to PHDAM

```
        DBD     NAME=SKILLINV,ACCESS=(PHDAM,OSAM),                      X
                RMNAME=(DFSHDC40,12,10000,824)
        SEGM    NAME=SKILMAST,BYTES=31,PTR=TWINBWD
        LCHILD  NAME=(NAMESKIL,PAYROLDB),PAIR=SKILNAME
        FIELD   NAME=(TYPE,SEQ,U),BYTES=21,START=1
        FIELD   NAME=STDCODE,BYTES=10,START=22
        SEGM    NAME=SKILNAME,                                          X
                PARENT=((SKILMAST,DBLE),(NAMEMAST,P,PAYROLDB)),         X
                BYTES=80,                                               X
                PTR=(TWINBWD,LPARNT,PAIRED),                            X
                RULES=(VVV,LAST)
        FIELD   NAME=(STDLEVL,SEQ),BYTES=20,START=61
        FIELD   NAME=(EMPLOYEE),START=1,BYTES=60
        DBDGEN
        FINISH
        END
```

Example 5: SKILLINV Database After Conversion to PHDAM

Symbolic Pointing vs. Direct Pointing

HALDB does not support symbolic pointers. All logical relationship pointers must be direct. The following provides guidance and instructions for making the conversion of logical relationship pointers from symbolic to direct.

Recognizing Symbolic Pointers

Symbolic pointers are specified in HDAM and HIDAM databases by two specifications on the SEGM statement for the logical child. First, the POINTER or PTR parameter does not have an LPARNT or LP keyword. LPARNT is used to include the direct pointer to the logical parent in the logical child. Second, the PARENT parameter includes PHYSICAL or P in the specification of the logical parent. This includes the logical parent's concatenated key (LPCK) in the logical child segment. The LPCK is the symbolic pointer. Example 6 shows a database with a symbolic pointer.

```
DBD      NAME=WIDGET,ACCESS=(HDAM,OSAM),                           X
         RMNAME=(DFSHDC40,60,50000,100)
DATASET  DD1=WIDDS1,BLOCK=4096,SCAN=0
SEGM     NAME=WIDNAME,BYTES=36,PTR=TWINBWD
LCHILD   NAME=(WIDGETN,WHAREHOU),PAIR=WHAREH
FIELD    NAME=(WIDTYPE,SEQ,U),BYTES=20,START=1
FIELD    NAME=WIDCODE,BYTES=6,START=21
SEGM     NAME=WHAREH,                                               X
         PARENT=((WIDNAME),(WHNAME,P,WHAREHOU)),                    X
         BYTES=26,                                                  X
         PTR=(TWIN,PAIRED),                                         X
         RULES=(VVV)
FIELD    NAME=(WHCODE,SEQ,U),BYTES=1,START=20
FIELD    NAME=QTY,START=21,BYTES=6
DBDGEN
FINISH
END
```

Example 6: WIDGET HDAM Database Which Contains a Symbolic Pointer

The SEGM statement with NAME=WHAREH is a logical child segment with a symbolic pointer to the WHNAME segment in the WHAREHOU database. First, the PTR parameter does not contain an LPARNT or LP keyword. LPARNT or LP is used to specify a direct pointer. Second, the PARENT parameter contains a P keyword in the specification of the logical parent. The P indicates that the LPCK is physically stored in the segment. The LPCK is the symbolic pointer.

Converting Symbolic Pointers to Direct Pointers

The following contains instructions for converting logical relationship symbolic pointers to direct pointers.

When a database is converted to HALDB there are no requirements to change the parameters in the SEGM statement for the logical child. You may specify the LPARNT or LP keyword in the PTR parameter but it is not required. LPARNT is a default value for the PTR parameter in HALDB logical children SEGM statements. Logical children with symbolic pointers include the PHYSICAL or P keyword in the specification of the logical parent in the PARENT parameter on the SEGM statement. P is always used with HALDB. There are no differences in LCHILD statements for direct pointers versus those for symbolic pointers. This means that the SEGM and LCHILD statements do have to be changed. If you wish, you may add the LPARNT to the PTR parameter on the SEGM statement.

Example 7 shows the DBD for an HDAM WIDGET database with a symbolic pointer to a logical parent in the WHAREHOU database. Example 8 shows the DBD for the database after it has been converted to PHDAM. LPARNT is included in the PTR parameter of the PHDAM WHAREH SEGM statement in example 8, but it is not required.

```
         DBD      NAME=WIDGET,ACCESS=(HDAM,OSAM),                      X
                  RMNAME=(DFSHDC40,60,50000,100)
         DATASET DD1=WIDDS1,BLOCK=4096,SCAN=0
         SEGM     NAME=WIDNAME,BYTES=36,PTR=TWINBWD
         LCHILD   NAME=(WIDGETN,WHAREHOU),PAIR=WHAREH
         FIELD    NAME=(WIDTYPE,SEQ,U),BYTES=20,START=1
         FIELD    NAME=WIDCODE,BYTES=6,START=21
         SEGM     NAME=WHAREH,                                         X
                  PARENT=((WIDNAME),(WHNAME,P,WHAREHOU)),              X
                  BYTES=26,                                            X
                  PTR=(TWIN,PAIRED),                                   X
                  RULES=(VVV)
         FIELD    NAME=(WHCODE,SEQ,U),BYTES=1,START=20
         FIELD    NAME=QTY,START=21,BYTES=6
         DBDGEN
         FINISH
         END
```
Example 7: WIDGET HDAM Database Which Contains a Symbolic Pointer

```
         DBD      NAME=WIDGET,ACCESS=(PHDAM,OSAM),                     X
                  RMNAME=(DFSHDC40,60,10000,100)
         SEGM     NAME=WIDNAME,BYTES=36,PTR=TWINBWD
         LCHILD   NAME=(WIDGETN,WHAREHOU),PAIR=WHAREH
         FIELD    NAME=(WIDTYPE,SEQ,U),BYTES=20,START=1
         FIELD    NAME=WIDCODE,BYTES=6,START=21
         SEGM     NAME=WHAREH,                                         X
                  PARENT=((WIDNAME),(WHNAME,P,WHAREHOU)),              X
                  BYTES=26,                                            X
                  PTR=(TWIN,PAIRED,LPARNT),                            X
                  RULES=(VVV)
         FIELD    NAME=(WHCODE,SEQ,U),BYTES=1,START=20
         FIELD    NAME=QTY,START=21,BYTES=6
         DBDGEN
         FINISH
         END
```

Example 8: WIDGET PHDAM Database Which Contains a Direct Pointer

Addendum to the IMS V8 and V9 Utilities Reference: System manuals

IMS Version 8 and Version 9 *Utilities Reference: System* manuals have incomplete descriptions of the PARENT parameter on the SEGM statement in DBDGEN. The following repeats the syntax of this parameter and provides a more complete description.

SEGM Statement PARENT Parameter for HDAM, PHDAM, HIDAM, and PHIDAM

Syntax:

```
                      ┌─,SNGL─┐
PARENT=──┬─(segname2─┼───────┼──┬──────────────────────────────────────────────┬─)─┬──────>
         │            └─,DBLE─┘  │                  ┌─,VIRTUAL──┐              │    │
         │                       └─(──lpsegname──┼───────────┼──,dbname1──)───┘    │
         │                                          └─,PHYSICAL─┘                   │
         └─ 0 ──────────────────────────────────────────────────────────────────────┘
```

Parameter Description:

PARENT=

Specifies the names of the physical and logical parents of the segment type being defined, if any.

0

For root segment types, the PARENT= keyword must be omitted or PARENT=0 specified.

segname2

For dependent segment types, specifies the name of this segment's physical parent.

SNGL or DBLE

Specifies the type of physical child pointers to be placed in all occurrences of the physical parent of the segment type being defined. SNGL/DBLE can be specified only for segments in HDAM, HIDAM, PHDAM, PHIDAM, or DEDB databases and are ignored if the physical parent specifies hierarchic pointers (PTR=HIER or HIERBWD).

SNGL causes a 4-byte physical child first pointer to be placed in all occurrences of the physical parent of the segment type being defined. SNGL is the default.

DBLE causes a 4-byte physical child first pointer and a 4-byte child last pointer to be placed in all occurrences of the physical parent of the segment type being defined.

lpsegname

Specifies the name of the logical parent of the segment type being defined, if any. This operand is used only during DBDGEN of a physical database, and it must be specified on SEGM statements that define real logical child segment types.

VIRTUAL or PHYSICAL
Specifies whether or not the logical parent's concatenated key (LPCK) is to be stored as a part of the logical child segment. It is specified for logical child segments only. If PHYSICAL is specified, the LPCK is stored with each logical child segment. If VIRTUAL is specified, the LPCK is not stored in the logical child segment. PHYSICAL must be specified for a logical child segment whose logical parent is in a HISAM database, or for a logical child segment that is sequenced on its physical twin chain through use of any part of the logical parent's concatenated key. Symbolic pointers in HDAM and HIDAM databases use the LPCK and require the PHYSICAL specification. VIRTUAL is the default for HDAM and HIDAM. PHYSICAL is the default for PHDAM and PHIDAM. If VIRTUAL is specified for PHDAM or PHIDAM it is ignored and PHYSICAL is used. VIRTUAL may be abbreviated as V. PHYSICAL may be abbreviated as P.

dbname1
Specifies the name of the database in which the logical parent is defined. If the logical parent is in the same database as the logical child, dbname1 can be omitted.