HP WORLD ‘96
Paper #3020

Using IMAGE/SQL with TPI Keys

Rich Trapp
OMNIDEX Product Manager - IMAGE/SQL

Dynamic Information Systems Corporation
5733 Central Avenue
Boulder, Colorado 80301
303-444-4000

Introduction

This paper discusses using TPI keys with IMAGE/SQL. First, it will define the various components that are relevant to the discussion, which includes, IMAGE/SQL and TPI. Next, it explains how to make this combination work, and how to verify that TPI and IMAGE/SQL are working together correctly.

IMAGE/SQL: What is it?

TURBOIMAGE

IMAGE/SQL is made up of two main components. The first component is what most MPE users know as IMAGE or TurboIMAGE. The second component is ALLBASE, or more precisely, parts of ALLBASE. For this discussion "IMAGE" refers to the TurboIMAGE part of IMAGE/SQL. This paper will leave it to the academics to define what type of database IMAGE is. If you are not familiar with TurboIMAGE, then refer to HP's TurboIMAGE Manual.

ALLBASE/SQL

ALLBASE/SQL is HP's fully relational database.

IMAGESQL

The program IMAGESQL.PUB.SYS attaches a TurboIMAGE database to an ALLBASE/SQL database. It reads the structures of the TurboIMAGE database and create mapping structures in the ALLBASE DBEnvironment file. It creates SQL INDEXES which map to TurboIMAGE keys and TPI GENERIC or SORTED keys. NOTE: It does not support MULTIPLE or KEYWORD third party indexes. This attachment allows a TurboIMAGE database to be read to and written from ALLBASE/SQL as if it were a set of SQL tables.

HP ALLBASE/SQL PC API

This allows PC users to access ALLBASE/SQL or IMAGE/SQL data via a client-server interface.

TPI: What is it?

TPI stands for Third Party Indexing. It is a standard intrinsic level interface between IMAGE and third party indexing products. It allows programmers to write applications that call IMAGE intrinsics to take advantage of Omnidex or Superdex without having to code any vendor-specific routines.

DBFIND

TPI modifies the action of DBFIND mode 1 and also adds several other modes that allow various types of indexed retrievals. This makes the code somewhat portable and easier to maintain rather than using vendor-specific routines. It also gives tools like QUERY or ALLBASE access to these indexes with a standard interface.

DBGET

Calling DBFIND mode 1 sets a pointer for a chained read of a detail set using DBGET mode 5. Under TPI, you can also use DBGET to search a third party index instead of an IMAGE search item. You can now do a DBFIND mode 1 followed by a DBGET mode 5 on a MASTER set if the key is a third party index. There are also other DBGET modes that correspond to other DBFIND modes as well. See your third party vendor documentation for more detailed information.

Omnidex or Superdex Sorted keys

Currently, the SQL optimizer only uses SORTED third party indexes, such as Level 1 of Superdex from Bradmark, or IMSAM from DISC. Other types of third party indexes can coexist, but only SORTED keys are supported for use in retrievals.

TPI

Enabling

First, decide which fields to index, then install the third party indexes and finally, enable the database for third party indexing. For IMSAM/Omnidex, you should use either DISC's DBINSTAL or OMNIUTIL utility programs when enabling or disabling the database. We do not recommend using DBUTIL to ENABLE/DISABLE the database for indexing for IMSAM/Omnidex users.

Verifying

The DBUTIL "SHOW <dbname> FLAGS" command shows you if TPI is enabled and which third party product is in use. For Omnidex, DBUTIL shows "Indexing is enabled for Omnidex". For Superdex, it shows "Indexing is enabled for Superdex". If TPI is disabled, it shows "Indexing is disabled".

Attaching the database to ALLBASE

To use third party keys from SQL, you must attach the IMAGE database to an ALLBASE DBE file. If you do not already have a DBE file created, the following procedure will create one for you.

```
HPIX3> RATDEV/IMAGESQL:IMAGESQL

HP36385B X.G1.03               IMAGE/SQL Utility   TUE, MAY 23, 1995,  9:26 AM
(C) COPYRIGHT HEWLETT-PACKARD COMPANY 1993

>>SET TURBODB SALES
>>SET SQLDBE SALESDBE
DBE does not exist, do you want to create one? [Y/N] : Y
Creating DBE now ...
>>ATTACH
Split 2 compound source field(s) (ATCWARN 32063).
Mapped 36 source table/source field name(s) (ATCWARN 32062).
>>EXIT
HPIX3> RATDEV/IMAGESQL:
```

Verifying indexes

Once you have attached the IMAGE database to the ALLBASE DBEFILE, you can verify that it has registered the indexes by following the procedure below:

```
:ISQL

                                       TUE, MAY 23, 1995,  9:38 AM
HP36216-02A.G1.08           Interactive SQL/3000            ALLBASE/SQL
(C)COPYRIGHT HEWLETT-PACKARD CO. 1982,1983,1984,1985,1986,1987,1988,
1989,1990,1991,1992,1993,1994. ALL RIGHTS RESERVED.

isql=> CONNECT TO 'SALESDBE';
isql=> SELECT * FROM SYSTEM.TPINDEX;

SELECT * FROM SYSTEM.TPINDEX;
--------------------+--------------------+--------------------+------+-----
INDEXNAME           |TABLENAME           |OWNER               |UNIQUE|CLUST
--------------------+--------------------+--------------------+------+-----
CUSTOMER_NAME_T1    |CUSTOMERS           |SALES               |     0|
```

```
LAST_PUR_DATE_T2    |CUSTOMERS           |SALES               |     0|
S_D_T3              |CUSTOMERS           |SALES               |     0|
CONTACT_T4          |CUSTOMERS           |SALES               |     0|
TITLE_T5            |CUSTOMERS           |SALES               |     0|
CITY_T6             |CUSTOMERS           |SALES               |     0|
STATE_T7            |CUSTOMERS           |SALES               |     0|
ZIP_T8              |CUSTOMERS           |SALES               |     0|
TERRITORY_T9        |CUSTOMERS           |SALES               |     0|
SALESMAN_T10        |CUSTOMERS           |SALES               |     0|
CUSTOMER_BAL_T11    |CUSTOMERS           |SALES               |     0|
COMMENTS_T12        |CUSTOMERS           |SALES               |     0|
PRODUCT_NO_T13      |PRODUCTS            |SALES               |     0|
PRODUCT_CLASS_T14   |PRODUCTS            |SALES               |     0|
PRODUCT_NAME_T15    |PRODUCTS            |SALES               |     0|
PRODUCT_DESC_T16    |PRODUCTS            |SALES               |     0|
--------------------------------------------------------------------------
First 16 rows have been selected.
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd] >
```

You will notice that every third party key, including MULTIPLE and composite keys (keys made up of more than one field or parts of fields), are registered. Each key has an "_T#" appended to it. The "T" stands for Third Party Index and the number is the order it appears. You cannot reference these keys directly. You may only use SORTED indexes which correspond to actual fields. MULTIPLE keys and composite or concatenated keys are not supported for this release.

Update statistics

To tell the SQL optimizer to use the indexes installed on this table, use the update statistics command:

```
:ISQL

                                        TUE, MAY 23, 1995,  9:41 AM
HP36216-02A.G1.08           Interactive SQL/3000            ALLBASE/SQL
(C)COPYRIGHT HEWLETT-PACKARD CO. 1982,1983,1984,1985,1986,1987,1988,
1989,1990,1991,1992,1993,1994. ALL RIGHTS RESERVED.

isql=> CONNECT TO 'SALESDBE';
isql=> UPDATE STATISTICS FOR TABLE SALES.CUSTOMERS;
isql=> EXIT;
```

Do this for every dataset that has indexes that you want to access through ALLBASE/SQL.

Retrievals

ISQL

Any time a SELECT statement is parsed, the SQL optimizer selects the most efficient method based on the information gathered during UPDATE STATISTICS. Any phrase involving a field that has a SORTED key on it causes SQL to consider using the SORTED key. Here are some examples:

```
isql=> SELECT CUSTOMER_NAME, CONTACT
         FROM SALES.CUSTOMERS C
         WHERE C.CUSTOMER_NAME LIKE 'ABC%';

SELECT CUSTOMER_NAME, CONTACT  FROM SALES.CUSTOMERS C  WHERE C.CUSTOMER_NAM
------------------------------+-------------------------
CUSTOMER_NAME                 |CONTACT
------------------------------+-------------------------
ABC Data Services Inc.        |Arlene D. Brandt
ABC Diskette                  |Dr. Mike Hubbard
ABC Electronic Sales Co.      |Thomas M. Garrettson
ABC Management Systems, Inc.  |Vernon W. Ruskin
.
.
.
--------------------------------------------------------------------------
Number of rows selected is 4
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd] >
```

```
isql=> SELECT CUSTOMER_NAME, CONTACT
         FROM SALES.CUSTOMERS ORDER BY CUSTOMER_NAME;

SELECT CUSTOMER_NAME, CONTACT FROM SALES.CUSTOMERS ORDER BY CUSTOMER_NAME;
------------------------------+-------------------------
CUSTOMER_NAME                 |CONTACT
------------------------------+-------------------------
1st Solutions                 |Donald A. Shifris
202 Data Systems, Inc.        |Kevin Galbraith
2M Corp.                      |Michael Canu
3CI                           |Rajiv P. Mehta
3Com Corp.                    |William Krause
3G Company, Inc.              |Maris Graube
3M                            |Tom Stark
3M Media Services, Inc.       |Mary Staples
4G Data Systems, Inc.         |Gregory Maugeri
A Computer Store              |Skip Robbins
.
.
.
--------------------------------------------------------------------------------
First 16 rows have been selected.
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd] >
```

ODBC

Because ODBC passes a SELECT to ALLBASE/SQL, it acts the same as ALLBASE/SQL. If you have a problem with an ODBC retrieval, display the SQL statement being executed and test the retrieval in ISQL.

Is ALLBASE/SQL using my indexes?

Using GENPLAN

To see if the optimizer will use an index, you can use the GENPLAN command in ISQL.
To use the GENPLAN command, insert “GENPLAN FOR” in front of the desired SELECT statement. This will cause ISQL to place the retrieval information in the SYSTEM.PLAN table. You can retrieve this table to see how the optimizer has chosen to access your data.

```
:ISQL

                                        TUE, MAY 23, 1995,  9:55 AM
HP36216-02A.G1.08           Interactive SQL/3000            ALLBASE/SQL
(C)COPYRIGHT HEWLETT-PACKARD CO. 1982,1983,1984,1985,1986,1987,1988,
1989,1990,1991,1992,1993,1994. ALL RIGHTS RESERVED.

isql=> CONNECT TO 'SALESDBE';
isql=> GENPLAN WITH (CUSTOMER_NAME CHAR(40)) FOR
         SELECT CUSTOMER_NAME, CONTACT
         FROM SALES.CUSTOMERS C
         WHERE C.CUSTOMER_NAME LIKE :CUSTOMER_NAME;
isql=> SELECT * FROM SYSTEM.PLAN;

SELECT * FROM SYSTEM.PLAN;
-----------+-----------+-----------+--------------------+-----------------
QUERYBLOCK |STEP       |LEVEL      |OPERATION           |TABLENAME
-----------+-----------+-----------+--------------------+-----------------
          1|          1|          1|index scan          |CUSTOMERS
.
.
.
--------------------------------------------------------------------------------
Number of rows selected is 1
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd] >
```

Notice the words “index scan” under ‘OPERATION’. This means that SQL is going to use an ‘index scan’ for this retrieval. If SQL didn’t have an index, or the optimizer thought it would be more efficient to not use it, the display would say ‘parallel scan’.

Here is an example where the optimizer determines that it will not use the indexes:

```
isql=> CONNECT TO 'SALESDBE';
isql=> GENPLAN FOR
       SELECT DELIVERY_DATE, CUSTOMER_NAME
       FROM SALES.ORDER_LINES, SALES.CUSTOMERS
       WHERE SALES.CUSTOMERS.CUSTOMER_NO = SALES.ORDER_LINES.CUSTOMER_NO;
isql=> SELECT * FROM SYSTEM.PLAN;

SELECT * FROM SYSTEM.PLAN;
-----------+-----------+-----------+--------------------+------------------
QUERYBLOCK |STEP       |LEVEL      |OPERATION           |TABLENAME
-----------+-----------+-----------+--------------------+------------------
          1|          1|          2|parallel scan       |ORDER_LINES
          1|          2|          2|hash scan           |CUSTOMERS
          1|          3|          1|nestedloop join     |
.
.
.
-------------------------------------------------------------------------------
Number of rows selected is 3
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd] >
```

Forcing an index scan

If GENPLAN shows that the optimizer is using a ‘serial scan’ instead of an ‘index scan’, you can force (most of the time, sometimes it doesn’t work) ISQL to use indexes by doing this:

```
:ISQL

                                       TUE, MAY 23, 1995,  9:44 AM
HP36216-02A.G1.08           Interactive SQL/3000            ALLBASE/SQL
(C)COPYRIGHT HEWLETT-PACKARD CO. 1982,1983,1984,1985,1986,1987,1988,
1989,1990,1991,1992,1993,1994. ALL RIGHTS RESERVED.

isql=> CONNECT TO 'SALESDBE';
isql=> SETOPT GENERAL INDEXSCAN;
```

Next, do the retrieval you wish to test and ALLBASE/SQL will use the indexes regardless of the decision of the SQL optimizer. Unfortunately, this command is not available except in ISQL. Any other access (like via ODBC) will have to rely on the optimizer’s decision.

If you change this option and do a ‘select * from system.plan;’ to check it, make sure you do a GENPLAN FOR SELECT... *BEFORE* you do the SELECT. If you don’t, you’ll see your old plan as it was before the SETOPT command.

Using DEBUG

To verify what is actually going on, you can use DEBUG to trap the DBFIND and DBGET calls which are taking place within ISQL. For this to work, you must be logged on as a user with PM (PRIV MODE) capability on the ACCOUNT and on the USER since you need to set a breakpoint in XL.PUB.SYS.

The file RATMAC contains the following DEBUG macros that display the arguments and the return status from DBFIND and DBGET:

```
macro bdbfind{
wl 'base   @ ', r26; dv r26,2,b;
wl 'set    @ ', r25; dv r25,4,b;
w  'mode   @ ', r24; dv r24,1,#,,2;
wl 'item   @ ', sp-#52; dv [sp-#52],4,b;
wl 'arg    @ ', sp-#56; dv [sp-#56],C,b;
var image_status:u32 r23;
var odx_status:u32 r23+#20;
C
}

macro bdbfind8 {
w 'image status @ ', image_status; dv image_status,1,#,,2;
w 'odx status   @ ', odx_status; dv odx_status,1,#,,2;
}

macro bdbget{
wl 'base   @ ', r26; dv r26,2,b;
wl 'set    @ ', r25; dv r25,4,b;
w 'mode   @ ', r24; dv r24,1,#,,2;
wl 'list   @ ', sp-#52; dv [sp-#52],4,b;
wl 'arg    @ ', sp-#56; dv [sp-#60],C,b;
var image_status:u32 r23;
var get_buffer:u32 [sp-#56];
C
}

macro bdbget8 {
w 'image status @ ', image_status; dv image_status,1,#,,2;
}

b dbfind,,,{bdbfind};
b ?dbfind+$8,,,{bdbfind8};
b dbget,,,{bdbget};
b ?dbget+$8,,,{bdbget8};
```

Start ISQL with DEBUG and use the macros:

```
HPIX3> RATDEV/IMAGESQL:RUN ISQL.PUB.SYS;DEBUG

DEBUG/iX B.79.06

DEBUG Intrinsic at: 7dc.0005b310 ?PROGRAM
$1 ($5a) nmdebug >USE RATMAC
added: NM    [1] PUB 205.01025cc0 dbfind
added: NM    [2] PUB 205.01025c64 ?dbfind+$8
added: NM    [3] PUB 205.01026c80 dbget
added: NM    [4] PUB 205.01026bec ?dbget+$8
$b ($5a) nmdebug >C
```

Connect and do the retrieval:

```
HP36216-02A.G1.08          Interactive SQL/3000          ALLBASE/SQL
(C)COPYRIGHT HEWLETT-PACKARD CO. 1982,1983,1984,1985,1986,1987,1988,
1989,1990,1991,1992,1993,1994. ALL RIGHTS RESERVED.

isql=> connect to 'salesdbe';
isql=> select customer_name, contact
```

```
       from sales.customers c
       where c.customer_name like 'ABC%';

Break at: NM    [1] PUB 205.01025cc0 dbfind
base   @ $418c5828
VIRT $2b4.418c5828 $ 00015341 4c45532e  ..SA LES.
set    @ $d6b32024
VIRT $a.d6b32024 $ 00014d00 01000101 00010001 00000005  ..M. .... .... ....
mode   @ $418c558c VIRT $2b4.418c558c #      1      0
item   @ $418c597c
VIRT $a.d6b32034 $ 00080001 00000001 41424340 3b202020  .... .... ABC@ ;
arg    @ $418c5978
VIRT $a.d6b3203c $ 41424340 3b202020 20202020 20202020  ABC@ ;
VIRT $a.d6b3204c $ 20202020 20202020 20202020 20200000                  ..
VIRT $a.d6b3205c $ 00000000 00000000 00000000 00000000  .... .... .... ....
Break at: NM    [2] PUB 205.01025c64 ?dbfind+$8
image status @ $418c5848 VIRT $2b4.418c5848 #      0      0
odx status   @ $418c585c VIRT $2b4.418c585c #    256      0
$b ($62) nmdebug >
```

The relevant parts above are underlined. You can see that the base-id at address 418c5828 is 0001; the dataset at address d6b32024 is 0001; the mode is 1; the item at address 418c597c is number 0008 (customer-name); and the argument at address 418c5978 is "ABC@;". So our SELECT statement has been turned into a DBFIND mode 1 on CUSTOMERS with CUSTOMER-NAME = "ABC@;". This sets up a TPI retrieval that allows DBGET mode 5 to retrieve the qualified records from this set. It's a MASTER set, but since we're using a Third Party Index, DBFIND is allowed.

Here's what the DBGET output looks like:

```
$b ($62) nmdebug > c
Break at: NM    [3] PUB 205.01026c80 dbget
base   @ $418c5828
VIRT $2b4.418c5828 $ 00015341 4c45532e  ..SA LES.
set    @ $d6b32024
VIRT $a.d6b32024 $ 00014d00 01000101 00010001 00000005  ..M. .... .... ....
mode   @ $d6b32032 VIRT $a.d6b32030 #      0      5
list   @ $418c597c
VIRT $a.d6b35064 $ 00000002 00080006 00000000 00000000  .... .... .... ....
arg    @ $418c5978
VIRT $2b4.418c20f8 $ 00000000 00000000 00000000 00000000  .... .... .... ....
VIRT $2b4.418c2108 $ 00000000 00000000 00000000 00000000  .... .... .... ....
VIRT $2b4.418c2118 $ 00000000 00000000 00000000 00000000  .... .... .... ....
Break at: NM    [4] PUB 205.01026bec ?dbget+$8
image status @ $418c5848 VIRT $2b4.418c5848 #      0     28
```

The base-id and dataset are the same as for DBFIND. The mode is 5, which retrieves the records selected by searching the SORTED index. The list contains 2 fields, field 0008 (CUSTOMER-NAME) and field 0006 (CONTACT).

Summary

There are 6 things that must be done use Third Party Indexes from the SQL side of IMAGE/SQL.

1. Install the Third Party Index on the IMAGE database
2. Enable that database for TPI.
3. Create an ALLBASE DBE FILE.
4. ATTACH the IMAGE database to the ALLBASE DBEFILE.
5. UPDATE STATISTICS for each TABLE with Third Party Indexes.
6. Use SELECT statements that reference these indexes when possible.

Conclusions

IMAGE/SQL has a lot to offer. Being able to access your IMAGE data via SQL, and all the doors that opens, is usually more than worth the effort involved. Third Party Indexing, although relatively new technology, has been put to good use on the IMAGE side of things. With these enhancements to ALLBASE to use these indexes where they exist, the benefit may be shared and hopefully, will make relational access even faster than previously possible.

CURRENT VERSIONS and PATCHES

If you need a patch for any part of IMAGE/SQL, you probably should get the current patches for all the pieces. Here is a list (as of 3/27/96) current patches for MPE/iX 5.0:

TURBOIMAGE version C.06.12 is patch:	**TIXHXX9.**
IMAGESQL version B.G1.10 is patch:	**ATCHXY3.**
ALLBASE/SQL version A.G1.15 is patch:	**SQLHXW0**
HP PC API ODBC version A.G1.05 is patch:	**SQLHXM2**