IMAGE/SQL and TPI

Gerry Johnson
M. B. Foster Associates
82 Main St
Chesterville Ontario

613-448-2333
613-448-2588
gerryj@mbfoster.com

Presentation #: 315

Tuesday, September 12, 2000

Introduction

The purpose of this presentation is to put forward the benefits of careful planning when creating an Allbase interface to Image. In particular, the proper use of third party indexes (TPI) can have a major impact on the effectiveness of your implementation.

Once you have connected an image database to Allbase through Image/SQL, queries will be checked by the optimizer to set up the best possible method for processing that particular SQL: statement. The optimizer utilizes several pieces of information including the various key information. The Image/SQL manual says,

"If the database is enabled for third-party indices (TPI), ATTACH enters definitions for all TPIs, excluding keyword indices and the ones for which the third-party does not provide information. The TPIs are entered as unique or non-unique based on the index configuration and information provided by the third-party for this index. While keys, search items, and B-Tree indices are registered in the specified DBEnvironment, third-party indices are registered in all attached DBEnvironments. Multiple index definitions on the same column can coexist and the SQL optimizer derives the optimal access plan based on the statistics present in the system catalog. In other words, the key or search item of the set can have a maximum of three index definitions. One will be a hash index definition entered automatically at ATTACH time, another can be a B-Tree index definition, and the third can be a third-party index definition. It is recommended that both B-Tree index and third-party index be not created on the same item as it will have an unnecessary impact on the performance (Optimizer calculates cost for each index). The Optimizer decides which index to use and the proper order of operations to ensure that the most efficient path is used. In the following example, SALES is attached to PARTSDBE. The accompanying message summarizes the mapping that took place during the attach.

Some history

Image is the major database component on most HPe3000s. It has been around since the first MPE operating systems were developed. It is a very solid database that can allow a programmer to develop high quality applications. The strength of Image along with the highly resilient operating system and hardware has made the HPe3000 a great success. Allbase/SQL was Hewlett Packard's entry into the SQL world. Allbase is a sophisticated SQL standard DBMS that can support industry standard transaction and query processing utilizing standard SQL.

HP needed a method to make the power of SQL available to Image users, so they could take advantage of SQL standard products such as the ODBC API. Originally a version of image was created that tried to add SQL processing to Image. This proved to be very slow and was discarded. Image/SQL was created to allow an Image database to be attached and handled by the powerful SQL processing provided by Allbase/SQL.

When you run Image/SQL and attach an image database to an Allbase DBE many things happen. A table created for each data set in Image will have the image name as the owner. You can connect several Image databases to a DBE. Each will be a unique database with its own owner name. Indexes are created in system tables that reflect Image keys, B-trees and Third party indexes. It is the presence of these indexes that allow for fast retrieval of data when criteria are used in select statements.

Third party indexing

Image was created to allow keyed reads on indexes with exact matches. You can read a master data set by its key and you can chain through Image details by following the lists of entries attached to a master. Until the creation of B-tree indexes, you could not do any partial key retrieval. Even then you could only do the partial retrieval on an Image key. Folks tried many ways to improve this process, such as creating MPE keyed files, KSAM to get around this problem. Dynamic Information Systems Corporation release their Omnidex product early on to address this limitation of Image. They were very successful but you had to use non-standard Image calls to make it work. It produced very fast access to sorted and key-worded keys. Bradmark Technologies released Superdex to address the same problem.

Customers were asked for standard methods to access their Image data and still obtain the power of this third party indexing systems without having to use the special calls. Hewlett Packard introduced the interface to third party indexing as a result. This enhancement allowed applications that did not have knowledge of the third party indexing system to use and update Image data without problem. It also allowed Image programmers to write code that would work no matter which vendor's indexing system was installed.

The popularity of client server applications and web-based applications that expose Image databases to users who may have no experience with the HPe3000 has made a profound difference. It becomes even more vital that the most efficient methods of retrieving data be utilized. In the old days when a user running QUERY.PUB.SYS, would issue a query, it became apparent very quickly that he was not using keys. The message 'USING SERIAL READ' was quite obvious. Now the user may not even be aware if an item is an index or even that they are communicating with an HPe3000. Tools like Impromptu, from Cognos or Crystal Reports, make producing reports very easy. Producing a well design index structure for your data becomes very critical to the success of your client server application. It is not enough to index every item, but the database administrator must pay careful attention to the types of keys he is installing. My favorite bad choice for an index is item sex in a human resources system. Since the item can have only two values and since they will be spread evenly in the data, this is a very bad choice. Careful thought is key to success.

Not all TPI indexes are exported to Image/SQL. The DBA can view all indexes associated with a database by examining SYSTEM.TPINDEX. Other users can view the TPIs to which they have access by examining CATALOG.TPINDEX. The manual states that only non-keyword indexes are exported. These indexes are stored in the system table SYSTEM.TPINDEX. This implies that you need to carefully consider the types of keys that you install on an Image database that will be used with Image/SQL.

Indexing systems

Omnidex uses three types of keys. Sorted or generic keys, no-parse key-word keys and regular key word keys. Sorted keys are treated much like Ksam where the b-tree is returned in sorted order. You can do a read on an individual key, or you can do generic lookups including ranges. These keys are exported to Allbase. No-parse keys are not returned in sorted order and are handled like regular key-word keys, accept that the whole item in indexed, not broken up into key-words. Key-word items have all the values in the item stored as individual indexes. Only keyword keys are not exported.

Superdex also supports multiple key types. You can purchase Superdex model I or model II. Model I utilize sorted keys and they should be exported to Allbase. Superdex model II supports both non-keyword keys and key-word keys. Key-word keys are not exported.

Now in many cases where key wording is not important to the application, it is better to select a sorted or no parse key type. If for example you are looking up a code or a date, key wording is of no particular value.

I have used the image database, TOYDB, as an example. I have created indexes for both Superdex and Omnidex. In both cases the entire third party index have been exported when I attached the database to a DBE.

I have used ISQL to list the indexes in the DBE TOYDBE to view the indexes.

Select * from system.tpindex;

```
------------------+------------------+------------------+------+-----
INDEXNAME          |TABLENAME         |OWNER             |UNIQUE|CLUST
-------------------+------------------+------------------+------+-
SI_PNAME_PLINE_T1  |PRODUCTS          |TOYDB             |     0|
SI_PNAME_T2        |PRODUCTS          |TOYDB             |     0|
SI_NAME_CITY_ST_T3 |CUSTOMERS         |TOYDB             |     0|
SI_NAME_T4         |CUSTOMERS         |TOYDB             |     0|
SI_CITY_T5         |CUSTOMERS         |TOYDB             |     0|
SI_STATE_T6        |CUSTOMERS         |TOYDB             |     0|
SI_INV_AMT_T7      |INVOICES          |TOYDB             |     0|
```

Below is the SIMAINT output from the Image base TOYDB showing the keys that have been created.

```
Dataset
  Path#  Path Name            Item Name          Typ Length/Offset
PRODUCTS
  10001  SI-PNAME-PLINE/B     PRODUCT-NAME         X      8
                              PRODUCT-LINE         X      1
  10002  SI-PNAME/B           PRODUCT-NAME         X      8
CUSTOMERS
  10003  SI-NAME-CITY-ST/B    CUSTOMER-NAME        X     15
                              CITY                 X     15
                              STATE                X     15
  10004  SI-NAME/B            CUSTOMER-NAME        X     15
  10005  SI-CITY/B            CITY                 X     15
  10006  SI-STATE/B           STATE                X     15
INVOICES
  10007  SI-INV-AMT/B         INVOICE-NO           X      3
                              PAID-AMOUNT          R      4
```

I have use ODBCLink/SE ODBCUTSE to show the tables that have keys and how they will appear to an application.

```
:odbcutse.odbcse.sys toydbe
MB Foster ODBC Utility Version E.58.03
Connecting to DBEnvironment toydbe...

0>show toydb.products
Field              ODBC-Datatype       HP-Datatype    Length  Offset
Nullable
PRODUCT_NO         CHAR(6)                            6       0      0
PRODUCT_NAME       CHAR(16)                           16      6      0
PRICE              DOUBLE                             8       24     0
PRODUCT_LINE       CHAR(2)                            2       32     0
QUANTITY           SMALLINT                           2       34     0
Indexed field                    Index Type          Index Name
```

```
PRODUCT_NAME+PRODUCT_LINE           TPI                  SI_PNAME_PLINE_T1
PRODUCT_NAME                        TPI                  SI_PNAME_T2
PRODUCT_NO                          IMAGE PRIMARY        PRODUCT_NO_M1
0>show customers
Field               ODBC-Datatype       HP-Datatype    Length  Offset
Nullable
CUSTOMER_NO         CHAR(6)                            6       0      0
CUSTOMER_NAME       CHAR(30)                           30      6      0
CITY                CHAR(30)                           30      36     0
STATE               CHAR(30)                           30      66     0
ZIPCODE             CHAR(6)                            6       96     0
ADDRESS             CHAR(30)                           30      102    0
SALES_AREA          CHAR(16)                           16      132    0
TURNOVER_1          DOUBLE                             8       152    0
TURNOVER_2          DOUBLE                             8       160    0
TURNOVER_3          DOUBLE                             8       168    0
TURNOVER_4          DOUBLE                             8       176    0
TURNOVER_5          DOUBLE                             8       184    0
TURNOVER_6          DOUBLE                             8       192    0
TURNOVER_7          DOUBLE                             8       200    0
TURNOVER_8          DOUBLE                             8       208    0
TURNOVER_9          DOUBLE                             8       216    0
TURNOVER_10         DOUBLE                             8       224    0
TURNOVER_11         DOUBLE                             8       232    0
TURNOVER_12         DOUBLE                             8       240    0
TURNOVER_PY         DOUBLE                             8       248    0
TURNOVER_MTD        DOUBLE                             8       256    0
Indexed field                       Index Type           Index Name
CUSTOMER_NAME+CITY+STATE            TPI                  SI_NAME_CITY_ST_T3
CUSTOMER_NAME                       TPI                  SI_NAME_T4
CITY                                TPI                  SI_CITY_T5
STATE                               TPI                  SI_STATE_T6
CUSTOMER_NO                         IMAGE PRIMARY        CUSTOMER_NO_M1
0>show invoices
Field               ODBC-Datatype       HP-Datatype    Length  Offset
Nullable
INVOICE_NO          CHAR(6)                            6       0      0
CUSTOMER_NO         CHAR(6)                            6       6      0
ORDER_NO            CHAR(8)                            8       12     0
AMOUNT              DOUBLE                             8       24     0
PAID_AMOUNT         DOUBLE                             8       32     0
INVOICE_DATE        CHAR(4)                            4       40     0
DUE_DATE            CHAR(4)                            4       44     0
Indexed field                       Index Type           Index Name
INVOICE_NO+PAID_AMOUNT              TPI                  SI_INV_AMT_T7
CUSTOMER_NO                         IMAGE                CUSTOMER_NO_D1
ORDER_NO                            IMAGE                ORDER_NO_D2
0>
```

I have used ISQL to list the indexes in the DBE TOYDBE to view the indexes.

```
SELECT * FROM SYSTEM.TPINDEX;
--------------------+--------------------+--------------------+------+-----
INDEXNAME           |TABLENAME           |OWNER               |UNIQUE|CLUST
--------------------+--------------------+--------------------+------+-----
PNAME_PLINE_T1      |PRODUCTS            |TOYDB               |     0|
PRODUCT_NAME_T2     |PRODUCTS            |TOYDB               |     0|
NAME_CITY_ST_T3     |CUSTOMERS           |TOYDB               |     0|
CUSTOMER_NAME_T4    |CUSTOMERS           |TOYDB               |     0|
CITY_T5             |CUSTOMERS           |TOYDB               |     0|
STATE_T6            |CUSTOMERS           |TOYDB               |     0|
INV_AMT_T7          |INVOICES            |TOYDB               |     0|
```

Below is the install script to add indexes to the Image base TOYDB showing the keys that have been created using Omnidex.

```
INSTALL TOYDB.OESC READONLY
;
TABLE PRODUCTS
CREATE PNAME-PLINE SORTED FROM PRODUCT-NAME PRODUCT-LINE
CREATE PRODUCT-NAME SORTED
;
TABLE CUSTOMERS
(24/46) Continue?
CREATE NAME-CITY-ST SORTED FROM CUSTOMER-NAME CITY &
     STATE
CREATE CUSTOMER-NAME SORTED
CREATE CITY SORTED
CREATE STATE SORTED
;
TABLE INVOICES
CREATE INV-AMT SORTED FROM INVOICE-NO PAID-AMOUNT OPTIONS NT
;
ACTIVATE
EXIT
```

I have use ODBCLink/SE ODBCUTSE to show the tables that have keys and how they will appear to an application.

```
ODBCUTSE.ODBCSE.SYS TOYDBE
```

```
MB Foster ODBC Utility Version E.58.03
Connecting to DBEnvironment TOYDBE...
0>SHOW PRODUCTS
Field               ODBC-Datatype       HP-Datatype    Length  Offset
Nullable
PRODUCT_NO          CHAR(6)                            6       0      0
PRODUCT_NAME        CHAR(16)                           16      6      0
PRICE               DOUBLE                             8       24     0
PRODUCT_LINE        CHAR(2)                            2       32     0
QUANTITY            SMALLINT                           2       34     0
Indexed field                       Index Type           Index Name
PRODUCT_NAME+PRODUCT_LINE           TPI                  PNAME_PLINE_T1
PRODUCT_NAME                        TPI                  PRODUCT_NAME_T2
PRODUCT_NO                          IMAGE PRIMARY        PRODUCT_NO_M1
0>SHOW CUSTOMERS
Field               ODBC-Datatype       HP-Datatype    Length  Offset
Nullable
CUSTOMER_NO         CHAR(6)                            6       0      0
CUSTOMER_NAME       CHAR(30)                           30      6      0
CITY                CHAR(30)                           30      36     0
STATE               CHAR(30)                           30      66     0
ZIPCODE             CHAR(6)                            6       96     0
ADDRESS             CHAR(30)                           30      102    0
SALES_AREA          CHAR(16)                           16      132    0
TURNOVER_1          DOUBLE                             8       152    0
TURNOVER_2          DOUBLE                             8       160    0
TURNOVER_3          DOUBLE                             8       168    0
TURNOVER_4          DOUBLE                             8       176    0
TURNOVER_5          DOUBLE                             8       184    0
TURNOVER_6          DOUBLE                             8       192    0
TURNOVER_7          DOUBLE                             8       200    0
TURNOVER_8          DOUBLE                             8       208    0
TURNOVER_9          DOUBLE                             8       216    0
TURNOVER_10         DOUBLE                             8       224    0
TURNOVER_11         DOUBLE                             8       232    0
TURNOVER_12         DOUBLE                             8       240    0
TURNOVER_PY         DOUBLE                             8       248    0
TURNOVER_MTD        DOUBLE                             8       256    0
Indexed field                       Index Type           Index Name
CUSTOMER_NAME+CITY+STATE            TPI                  NAME_CITY_ST_T3
CUSTOMER_NAME                       TPI                  CUSTOMER_NAME_T4
CITY                                TPI                  CITY_T5
STATE                               TPI                  STATE_T6
CUSTOMER_NO                         IMAGE PRIMARY        CUSTOMER_NO_M1
0>SHOW INVOICES
Field               ODBC-Datatype       HP-Datatype    Length  Offset
Nullable
INVOICE_NO          CHAR(6)                            6       0      0
CUSTOMER_NO         CHAR(6)                            6       6      0
ORDER_NO            CHAR(8)                            8       12     0
AMOUNT              DOUBLE                             8       24     0
PAID_AMOUNT         DOUBLE                             8       32     0
INVOICE_DATE        CHAR(4)                            4       40     0
DUE_DATE            CHAR(4)                            4       44     0
```

```
Indexed field                  Index Type          Index Name
INVOICE_NO+PAID_AMOUNT         TPI                 INV_AMT_T7
CUSTOMER_NO                    IMAGE               CUSTOMER_NO_D1
ORDER_NO                       IMAGE               ORDER_NO_D2
0>
```