IMAGE Internals and Performance - a Robelle tutorial

Bob Green

Robelle Solutions Technology Inc.
Telephone: 604.582.1700
Fax: 604.582.1799
Email: bgreen@robelle.com
HPWorld, Chicago: August 2001

1

IMAGE/SQL databases are inherently messy, chaotic places. Their tidy exteriors sometimes conceal nasty surprises that can hamper performance and reduce the speed of retrievals and updates. But armed with the right tools and know-how, you can quickly bring order to the chaos.

The HowMessy program from Robelle and the Dbloadng program from Allegro can report on the internal inefficiency of your IMAGE database to help you determine why performance is not what it should be.

Then you can use Adager or DBGeneral to apply the changes that will reorganize the data along the most efficient lines. This how-to session will use the HowMessy report to teach you the theory of IMAGE database efficiency, including blocks, paging, disc I/O, and new features such as B-trees. This dynamic information session is a must for MPE system managers and IMAGE database administrators.

For Techies

References

What's Inside

2

This tutorial is hosted by Bob Green, President and founder of Robelle. Bob thanks François Desrochers and Neil Armstrong of Robelle for their work on the tutorial materials.

Bob Green is a software developer, award winning author and speaker, and member of the Interex Hall of Fame. Bob created Robelle to develop irresistible software. You can visit the Robelle web site at

`www.robelle.com`

With 22 years of success, 14,000 product installations world-wide and over 10% of the Fortune 500 as Robelle customers, they must be doing something right. Over the years, Bob and his small staff have published dozens of technical papers, so many that strangers often imagined that Robelle must be a huge company.

Bob welcomes your comments at: `bgreen@robelle.com`

For Techies

References

How slow is your database?

- A database is slow if it takes more I/O than it should
- Unnecessary I/O is still a major limiting factor, even on N-Class machines
- Databases are slow by nature
- Run HowMessy or DBGeneral against your database
  - HowMessy is a bonus program for Robelle and Adager customers
  - DBGeneral is a product offered by Bradmark, Inc.

3

The high-end N-class HP 3000 servers can now do up to 400 I/Os per second (see http://www.robelle.com/library/papers/ios400.pdf). But, databases and demands grow even faster, so disc I/O can still have a significant impact on performance. An inefficient database consumes more disc I/O than a database that is optimized.

Two programs can help identify messy databases. One, a bonus program distributed to Robelle customers, is called HowMessy. Adager has made special arrangements with Robelle to distribute HowMessy to all Adager customers. The program is located in the Library group of the Rego account.

Another is the report feature of DBGeneral. DBGeneral's section 2 provides a full suite of diagnostics and repair functions ranging from dataset-level detailed analysis (sections 2.1 to 2.4) to high-speed global diagnostics (section 2.6).

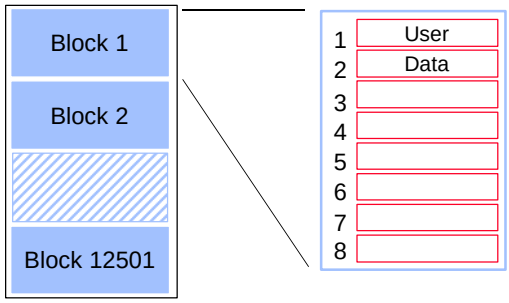
Both programs produce a report that shows the database's internal efficiency. Databases are inefficient by nature. Unless you never add new entries or delete existing entries (e.g., use your database for archival purposes), your database becomes inefficient over time. In this tutorial, we will use the HowMessy report style in most of our examples.

For Techies

References

Blocks Vs Pages

- IMAGE/SQL organizes records into fixed-size Blocks, but does all I/O operations in 4,096-byte pages
- A page may contain many user records
- More entries per page means fewer I/Os
- Fewer I/Os means better performance



Blocking factor = 8

Capacity: 100001

4

Internally, all IMAGE/SQL datasets are organized into blocks. The size of these blocks is controlled by the Blockmax parameter of the $control statement in the schema. The default block size is 1,024 bytes ($control blockmax=512, because the schema uses 16-bit words). Each block contains one or more user records. The actual number of records per block is known as the blocking factor.

The block organization is a hangover from the original design of IMAGE on the Classic HP 3000 computers (MPE V). On MPE/iX, IMAGE/SQL actually performs page-level I/O. A page has 4,096 bytes. This means that one or more IMAGE/SQL blocks are read or written as part of each I/O operation.

Because HowMessy reports statistics by block, not by page, we will mostly talk about blocks throughout the tutorial. When the block statistics may not be a good approximation of MPE/iX page I/O, we will point it out. Whether dealing with blocks or pages, the goal is to minimize the number of I/Os required to access the database.

HowMessy and DBGeneral quantify many of the I/O operations that take place in your database.

For Techies

References

Record location in masters

- Search item values must be unique
- Location of entries is determined by a hashing algorithm or a primary address calculation
- Calculation is done on search item value to transform it into a record number between one and the capacity
- Different calculation depending on the search item type
  - X, U, Z, and P give random results
  - I, J, K, R, and E give predictable results

5

For master datasets, each search item must have unique values. IMAGE/SQL determines the location of each entry by performing a calculation on the search item value. This is known as a hashing algorithm or a primary address calculation. The result of this calculation is a record number between one and the capacity of the dataset. The capacity is the maximum number of records you can have.

IMAGE/SQL uses a different algorithm depending on the search item data-type.

The goal for master datasets is to generate one disc access for a hashed DBGET (mode-7), and three disc I/Os for a DBPUT: one to read the block, one to write it back, and one to update the file label.

Update:

MPE/iX 6.5 Express 2, Image C.09.02, has changed the internal record pointer format in datasets from **entryname** format to **entrynumber** format. Previously, Image internal record numbers were in **entryname** format: 32-bits, the first 24 bits of which contain the block number and the last eight bits contain the position of the record within that block. Now the pointer is in **entrynumber** format: the relative record position in the entire dataset.

With the new limit of 2 billion records and the entry size maximum being 4756 bytes, the dataset size limit has gone from 80GB to over 10 Terrabytes.

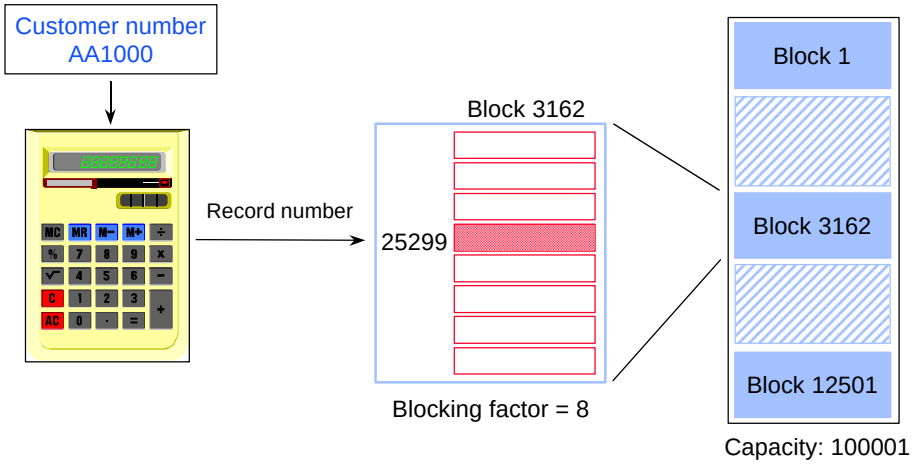
For Techies

References

A summary of the hashing algorithm is in Chapter 10 of the *TurboIMAGE Reference Manual*.

Hashing algorithm

- Customer number AA1000 is transformed into a record number



6

Let us assume you have an empty dataset, and you are gradually populating it with customer information.

When adding a record for customer AA1000, IMAGE/SQL performs the hashing algorithm and comes up with record number 25299. This is the first record that is added to this block. This example requires the minimum number of disc accesses to add the entry and read it back later.

More Information:

Read Alfredo Rego's paper, "*Do migrating secondaries give you migraines?*"

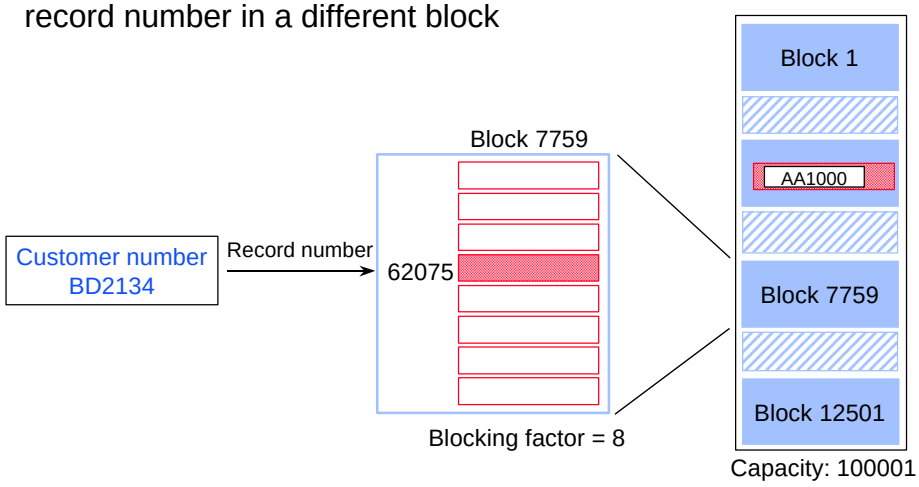
Located on the web at
http://www.adager.com/TechnicalPapersHTML/Migraines.html

For Techies

References

Hashing algorithm (no collision)

- Customer number BD2134 gives a different record number in a different block



7

The second customer number BD2134 also hashes to an empty block. Again, this is a perfect situation in which only the minimum number of disc accesses is used.

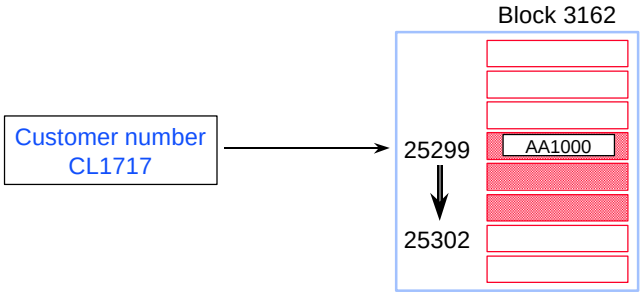
Primary: any record, like these two, which is stored directly at its hash location, is called a primary.

For Techies

References

Hashing algorithm (collision - same block)

- Customer number CL1717 hashes to the same record number as AA1000 location.
- IMAGE/SQL tries to find an empty location in the same block. If it finds one, no additional I/O is required.
- CL1717 becomes a secondary entry. Primary and secondary entries are linked using pointers that form a chain.



8

Although hashing algorithms are designed to give unique results, sometimes IMAGE/SQL calculates the same record number for two different search item values. In this case a new entry may want to occupy the same space as an existing **Primary** entry. This is known as a collision, and the new entry is called a **Secondary**.

Most master datasets have secondaries. If you have a lot of secondaries, or if they are located in different blocks, you should be concerned about generating more I/Os than needed.

In this example, customer number CL1717 collides with AA1000, and CL1717 becomes a secondary. IMAGE/SQL then tries to find the closest free record location. In this case, the next empty record number is 25302, which is in the same block. No additional I/O is needed.

IMAGE/SQL uses pointers to link the primary entry with its secondary. Record 25299 for AA1000 has a pointer to record number 25302 and vice versa. This is known as a **synonym chain**.

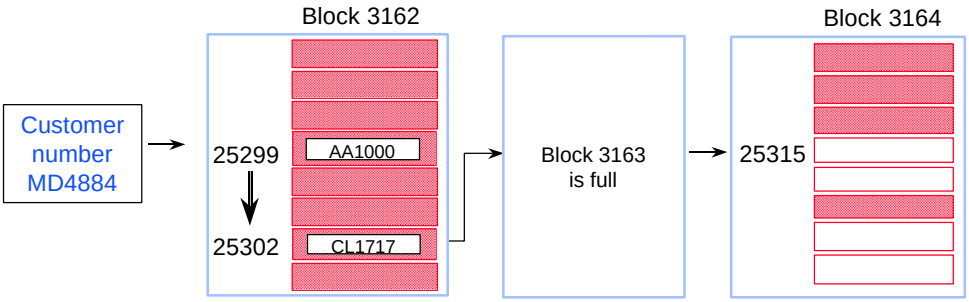
For Techies

Define Primary and Secondary

References

Hashing algorithm (collision - different block)

- Customer number MD4884 collides with AA1000
- No more room in this block. IMAGE/SQL reads the following blocks until it finds a free record location.
- In this case, MD4884 is placed two blocks away, which requires two additional I/Os.



9

If a customer number collides with a primary entry, and there are no free locations in the same block, IMAGE/SQL must do at least one more read to find an empty location. It does this by reading the following blocks in serial fashion. Once it finds a location, it requires one more write to update the synonym chain.

In the example shown on the slide above, adding the new “25315” entry requires two extra I/Os because the following block is full. However, once the record is created, it only takes one extra I/O to find the “25315” record on a retrieval. That is because IMAGE/SQL follows the synonym chain, which points to “25302” in the same block, then directly to the target “25315”, skipping over the full block.

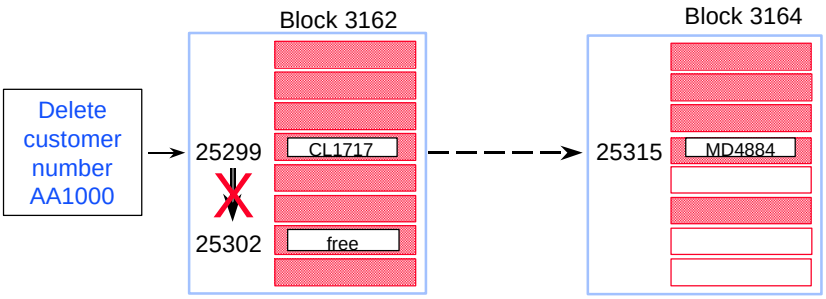
Of course, as a synonym chain gets longer, retrievals on the chain may take longer!

For Techies

References

Migrating secondary (delete primary entry)

- A primary entry location must contain a record
- AA1000 is deleted, CL1717 must be moved into its place
- Synonym chain now has only 2 entries



10

Because the primary entry is also the **head** of the synonym chain, the primary location must always contain a record that is a member of the synonym chain (if any). If IMAGE/SQL deletes the existing primary entry, it moves the first secondary in the chain to the primary position. And adjusts the pointers in the chain accordingly. If it didn't do this, it would not be able to find the synonyms on retrievals, since everything starts by hashing to the primary location.

In the example above, customer AA1000 is deleted. CL1717 must be moved into AA1000's location, and CL1717's old location is released. Because both entries are in the same block, IMAGE/SQL requires only two I/O operations (one read and one write). If CL1717 had been stored in a different block, the operation would have required four I/Os.

In either case, the pointers in the other secondary, customer MD4884, must be updated. This requires two more I/O operations.

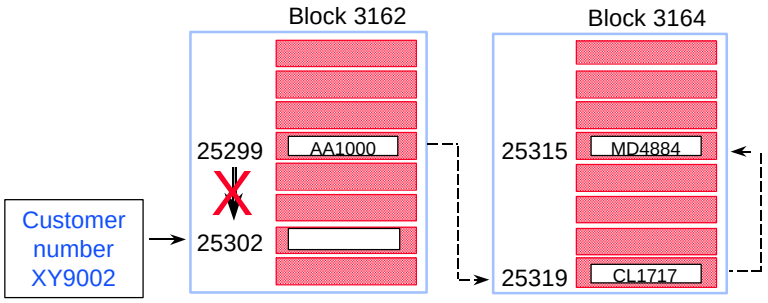
For Techies

Define 'Migrating Secondary'

References

Migrating secondary (occupying a primary location)

- A secondary cannot occupy a primary entry location
- Customer number XY9002 collides with CL1717
- CL1717 must be moved to a new empty location



11

A secondary, however, cannot occupy the location of a primary entry for a different synonym chain. If a new primary entry collides with an existing secondary, IMAGE/SQL must move the secondary to a different location.

In the example above, customer XY9002 is added. The calculated primary location is already occupied by customer CL1717, which is a secondary to AA1000. IMAGE/SQL must find an empty location for CL1717. Using the same logic that was used to add the first entry, it reads forward serially until it finds an empty location for the migrating secondary.

Notice in the example, the two secondaries are now in the same block. This is only chance. Under different circumstances, the entry for customer CL1717 could have moved to a totally different block, much farther away.

Once IMAGE/SQL has moved the existing secondary out of the desired primary location, it updates the pointers in the synonym chain (i.e., in AA1000 and MD4884). Again, the number of additional I/O operations will vary depending on the new location for CL1717.

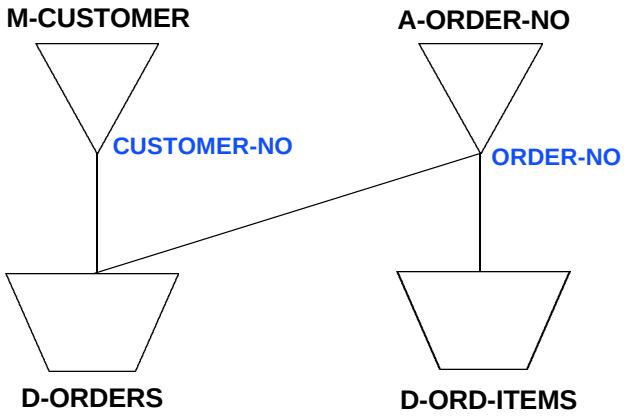
All this extra work has to be done before the new record can be inserted.

For Techies

Why can't a secondary occupy the location of a primary?

References

A sample IMAGE/SQL database



12

This is a very simple database. We have two master datasets (m-customer and a-order-no) and two detail datasets (d-orders and d-ord-items).

M-customer is a manual master that contains the customer information (name, address, etc.).

A-order-no is an automatic master that contains only order numbers.

D-orders is a detail dataset that contains order header information. It is linked to m-customer and a-order-no. Multiple records can exist for a particular customer, but each order number can have only one record.

D-ord-items is a detail dataset that contains the items on an order. It is only linked to a-order-no. In this dataset, a specific order number can have multiple records.

For Techies

References

HowMessy sample report

HowMessy/XL (Version 2.6) for IMAGE/3000 databases
Data Base: STORE.DATA.INVENT By Robelle Solutions Technology Inc.
Run on: SUN, JAN 9, 2000, 11:48 AM Page: 1

Data Set	Type	Capacity	Entries	Load Factor	Secon-daries (Highwater)	Max Blks	Blk Fact
M-Customer	Man	248113	178018	71.7%	30.5%	1496	11
A-Order-No	Ato	1266783	768556	60.7%	25.7%	1	70
D-Orders	Det	1000000	768558	76.9%	(851445)		32
D-Ord-Items	Det	4000000	3458511	86.5%	(3470097)		23

Search Field	Max Chain	Ave Chain	Std Dev	Expd Blocks	Avg Blocks	Ineff Ptrs	Elong-ation
Customer-No	32	1.92	0.32	1.00	1.90	90.5%	1.90
Order-No	10	1.35	0.62	1.00	1.00	0.0%	1.00
!Order-No	1	1.00	0	1.00	1.00	0.0%	1.00
S Customer-No	80	14.34	17.76	1.75	9.20	57.2%	5.25
S !Order-No	1604	8.06	35.75	1.36	11.32	72.5%	8.34

13

The heading on the report shows the program name and version, the fully-qualified database name, and the date and time the report was produced.

The *Data Set* column shows the dataset name. The *Type* column shows the dataset type: “Man” for a manual master, “Ato” for an automatic master, “MDX” for a Master configured with dynamic dataset expansion, “Det” for a detail dataset, or “DDX” for a detail dataset configured with dynamic dataset expansion.

The *Capacity* column shows the capacity of the dataset. For MDX and DDX datasets, this represents the maximum capacity allowed, not the current capacity. It is followed by the *Entries* column, which shows the current number of entries. The *Load Factor* column is the percentage obtained when you divide Entries by Capacity.

The remaining columns have slightly different meanings depending on the dataset type. We will discuss these on the following pages.

For Techies

References

HowMessy sample report (master dataset)

HowMessy/XL (Version 2.6) for IMAGE/3000 databases
Data Base: STORE.DATA.INVENT
By Robelle Solutions Technology Inc.
Run on: SUN, JAN 9, 2000, 11:48 AM
Page: 1

Data Set	Type	Capacity	Entries	Load Factor	Secon- daries	Max Blks (Highwater)	Blk Fact
M-Customer	Man	248113	178018	71.7%	30.5%	1496	11
A-Order-No	Ato	1266783	768556	60.7%	25.7%	1	70
D-Orders	Det	1000000	768558	76.9%	(	851445)	32
D-Ord-Items	Det	4000000	3458511	86.5%	(	3470097)	23

Search Field	Max Chain	Ave Chain	Std Dev	Expd Blocks	Avg Blocks	Ineff Ptrs	Elong- ation
Customer-No	32	1.92	0.32	1.00	1.90	90.5%	1.90
Order-No	10	1.35	0.62	1.00	1.00	0.0%	1.00
!Order-No	1	1.00	0	1.00	1.00	0.0%	1.00
S Customer-No	80	14.34	17.76	1.75	9.20	57.2%	5.25
S !Order-No	1604	8.06	35.75	1.36	11.32	72.5%	8.34

14

Secondaries	Percentage of entries that are secondaries
Max Blks	Maximum number of contiguous blocks IMAGE/SQL has to read before finding a free location for a secondary. This is the worst case scenario, not the actual number.
Blk Fact	Blocking Factor column indicates the number of records in each block (not an MPE/iX page).
Search Field	Key item name
Max Chain	Maximum number of entries in the longest synonym chain
Ave Chain	Average number of entries in the synonym chain
Std Dev	Standard deviation shows the deviation from the average that is required to encompass 90% of the cases. If the deviation is small, then most chains are close to the average.
Expd Blocks	Number of blocks we expect to read in the best case scenario (usually 1.00 for master datasets) to get a whole synonym chain
Avg Blocks	Number of blocks needed to hold the average chain
Ineff Ptrs	Percentage of secondary entries that span a block boundary
Elongation	Average number of blocks divided by the expected number of blocks

For Techies

References

Interpreting master dataset lines

- Pay attention to the following statistics:
  - High percentage of Secondaries (inefficient hashing)
  - High Maximum Blocks (clustering)
  - High Maximum and Average Chains (inefficient hashing)
  - High Inefficient Pointers (when secondaries exist)
  - High Elongation (when secondaries exist)

15

A high percentage of secondaries might indicate that the hashing is not working properly. The dataset may be getting full. A master dataset load factor should be kept below 70%.

A high number of Maximum Blocks might indicate a clustering effect. Some sections are really full, while others are empty. Entries are not being spread evenly throughout the dataset. The clustering increases as the numbers in Max Blks and the Blocking Factor get larger.

High values in Maximum and Average Chains are a side effect of a high proportion of secondary entries. High values in these fields indicate that many secondaries are colliding with a few of the same primary entries. If there are many secondaries, low values in these fields indicate that collisions are random. In the latter case, you can often solve the problem if you increase the capacity.

When a dataset has secondary entries with high values in Inefficient Pointers, or Elongation, or both, there may be a problem with the hashing algorithm. When you see these factors together, you will likely also see clustering.

For Techies

References

Report on m-customer

- The number of secondaries is not unusually high
- However, there may be problems
  - Records are clustering (high Max Blks)
  - Long synonym chain
  - High percentage of Inefficient Pointers

Data Set	Type	Capacity	Entries	Load Factor	Secon- daries (Highwater)	Max Blks	Blk Fact
M-Customer	Man	248113	178018	71.7%	30.5%	1496	11

Search Field	Max Chain	Ave Chain	Std Dev	Expd Blocks	Avg Blocks	Ineff Ptrs	Elong- ation
Customer-No	22	1.92	0.32	1.00	1.90	90.5%	1.90

16

This is not a tidy dataset. The number of secondaries is fairly high (30.5%, or around 54,000). The Maximum Chain length is 22. This is unusual for a master dataset.

The high Max Blks value (1,496) suggests some clustering. All of this indicates that the hashing is not producing random distribution. If you add a new record whose primary address happens to be the first block of the contiguously-occupied 1,496 blocks, IMAGE/SQL will have to do many extra disc I/Os just to find the first free entry for the new record (assuming that MDX has not been enabled on this dataset).

For Techies

References

Report on a-order-no

- Very tidy dataset
  - Number of secondaries is acceptable
  - Max Blks, Ineff Ptrs and Elongation are at the minimum values, even if the Maximum Chain length is a bit high

Data Set	Type	Capacity	Entries	Load Factor	Secon-daries (Highwater)	Max Blks	Blk Fact
A-Order-No	Ato	1266783	768556	60.7%	25.7%	1	70

Search Field	Max Chain	Ave Chain	Std Dev	Expd Blocks	Avg Blocks	Ineff Ptrs	Elong-ation
Order-No	10	1.35	0.62	1.00	1.00	0.0%	1.00

17

This dataset is very tidy. Although the percentage of secondaries is similar to the one for m-customer (30.5% versus 25.7%), the longest synonym chain has ten records with a 1.35 average. The Inefficient Pointers and Elongation are at their lowest values. Max Blks of 1 means IMAGE/SQL can find a free record location within the same block every time it searches for one.

These figures can be explained in part by the high Blocking Factor. It also means the hashing algorithm is working correctly, dispersing records evenly across the dataset.

For Techies

References

DBGeneral master dataset report

- "Master Dataset Summary" section
- Very similar to the HowMessy report
- Possible problems due to synonyms and inefficient pointers

	dataset		k t	blk fct	% full	% kv cap	% synonyms % now	% exp
3	USERMAST	M	X	3	50	--	27*	21

Synonym chains		block distribution			pointer inef	
avg len	max len	%blk full	%blk cont	avg pages	% <1P	% inef
1.3	3	18	4	.1	50	50

18

The HowMessy and DBGeneral efficiency reports are similar. This is an example of the DBGeneral master dataset report. The *%full*, *%synonyms*, *avg len* (average chain length), and *max len* (maximum chain length) are the same as in the HowMessy report. The *%blk full* is the percentage of blocks in the dataset that are already full.

The DBGeneral report provides some extra information. In addition to the percentage of full blocks (%blk full), we are also shown the percentage of blocks that are contiguous (%blk cont). For this dataset, 4% of all blocks are contiguous (making it more likely that IMAGE/SQL will require extra disc I/O to find the next free entry).

The DBGeneral report gives the final statistics in 4,096-byte pages instead of blocks. On average, the synonym chains occupy 0.1 pages. One half (50%) of the synonym chains span a page, thus making 50% of the synonym retrievals inefficient.

For Techies

References

DBGeneral master dataset report

- "Master Dataset Distribution" section
- Another way of viewing the data

```
USERMAST          full  |                                        s    ss     s     s|
#3                      |                                        s    ss     s     s|
                        |                    P  P                s    ss     s     s|
primaries   :    72.7%  |   s    ss s s  P  P     s  s  ss   Ps   s s    ss|
secondaries :    27.3%  |   s    ss s s  Ps P     s  ss ss   Ps   s s    ss|
average     :     2.2   |  PP   PPP P PPPPs PP   PP PPs PP  PPPP PP P   PPs|
std dev     :     1.5   |  PP   PPP P PPPPssPP  sPPsPPs PPs PPPPsPP Ps  PPP|
%full       :    50.0%  |P PPPP PPP PPPPPPsPPP  PPPPPPsPPPP PPPPPPP PPPPPPP|
capacity    : 220       |P PPPP PPP PPPPPPPPPP  PPPPPPPPPPP PPPPPPP PPPPPPP|
entry count : 110       |P PPPP PPPsPPPPPPPPP  PPPPPPPPPPP PPPPPPPsPPPPPPP|
                  empty 0---10---20---30---40---50---60---70---80---90---100
```

19

DBGeneral also provides a graphical view of the distribution of primary and secondary entries in a master dataset. On the left-hand side, you see the same statistics as in the previous report. The *average* and *std dev* are for the secondary chains in the master dataset.

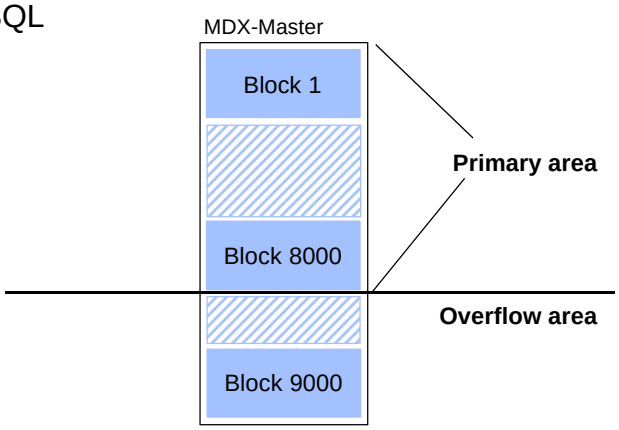
In the histogram, the horizontal axis represents a percentile of the dataset from 0 to 100. The vertical axis represents the percent fullness, incremented by tenths. A vertical bar that is 10 letters high indicates a section of the dataset that is completely full. If it were 8 letters high, it would be 80% full. The letters *P* and *s* indicate the portion of the percent fullness that is occupied by either primaries or secondaries in each part of the dataset.

For Techies

References

Dynamic dataset expansion (master - MDX)

- Primary area used by hashing algorithm
- Overflow area used when primary area usage reaches a threshold set in IMAGE/SQL



20

A "regular" master dataset uses all existing blocks for address calculation. As the master is filled up, there might be a point where the number of secondaries will cause performance degradation. Increasing the capacity requires exclusive access to the database. Even though this is probably the best thing to do, it might not be easy to schedule.

In version C.07.14 of IMAGE/SQL, HP introduced dynamic dataset expansion for masters. Physically, disc space is fully allocated as with "regular" masters. The difference here is that you tell IMAGE/SQL to use only part of that space as a primary area. The hashing algorithm uses blocks in the primary area for address calculation.

If there is a collision, IMAGE/SQL searches serially for an empty location. It will search in a maximum of 40 blocks or 10% of the primary area, whichever is smallest. If it cannot find an empty location, the secondary entry is added to an overflow area reserved at the end of the file. Entry allocation within the overflow area works like a detail dataset (to be discussed later) with DDX enabled.

This should remove some of the performance penalties associated with a large number of secondaries. Note, this approach has other limitations and should not be seen as a permanent solution. It's just a way to buy you some time until you can fix things later.

For Techies

References

Master dataset solutions

- Increase capacity to a higher odd number
- Increase the Blocking Factor
  - Increase block size
  - Reduce record size
- Change binary keys to type X, U, Z, or P
- Check your database early in the design
- Use HowMessy on test databases

21

Because the capacity is used in the hashing algorithm, you can reduce secondaries if you increase capacity. In addition, it is generally agreed that a capacity with an odd or prime number provides better distribution of records. An capacity with an even number usually gives poor results; avoid a capacity that is a power of two (2).

Because all I/O operations are done in one or more blocks, you can improve performance if you put more records in each block. You can increase the block size with the Blockmax option in DBSCHEMA or with one of the various database tools available. The other option is to reduce the size of each record. To do this, use different data-types (I, J, or P instead of Z for storing numeric values) or assign the correct maximum number of characters in each field. In other words, do not use a 50-character field if 30 characters is sufficient. The idea is to save space.

Changing the search item data-type can also help the hashing algorithm. This algorithm varies depending on the search item type. Data-types X, U, Z, and P usually produce a better distribution of records. When allocated randomly, record numbers tend to be distributed more evenly. Record numbers produced from binary data-types, such as I, J, and K, are both easily predictable and more likely to produce clustering.

Keep in mind that it is easier to change a database structure when it is still in the design stage. Check your structure as early as possible. Ideally, you should try to simulate the production environment, then use HowMessy to look at the results.

For Techies

References

Analyzing the report

HowMessy/XL (Version 2.6) for IMAGE/3000 databases
Data Base: SOMEDB.DATA.SOMEACCT
By Robelle Solutions Technology Inc.
Run on: SUN, JAN 9, 2000, 11:48 AM
Page: 1

Data Set	Type	Capacity	Entries	Load Factor	Secon- daries	Max Blks (Highwater)	Blk Fact
Part-Master	Ato	10000	4305	43.0%	14.2%	16	78
Part-Loc-Master	Ato	606010	303005	50.0%	86.4%	612	67
Parts-Detail	Det	303030	303005	100.0%	(	303005)	63

Search Field	Max Chain	Ave Chain	Std Dev	Expd Blocks	Avg Blocks	Ineff Ptrs	Elong- ation
Part-No	3	1.17	0.40	1.00	1.67	63.3%	1.67
Part-Loc-No	4305	7.38	73.09	1.07	45.84	66.4%	42.90
Part-No	41065	70.38	666.46	1.86	3.20	3.1%	1.72
!Part-Loc-No	1	1.00	0.00	1.00	1.00	0.0%	1.00

22

This example is based on an actual database. We have a detail dataset linked to two master datasets. Part-master and parts-detail are fine. Part-loc-master, however, is not doing as well as the others. Notice the high numbers in most of its columns.

The numbers in Secondaries, Max Chain, and Avg Blocks are extremely high. Max Blks and Avg Chain are also fairly high. The hashing algorithm is having difficulty distributing the entries.

Notice the key part-loc-no is an 8-byte integer.

For Techies

References

Solution #1: Change key type to byte and retain data values

HowMessy/XL (Version 2.6) for IMAGE/3000 databases — Data Base: SOMEDB.DATA.SOMEACCT — By Robelle Solutions Technology Inc. — Run on: SUN, JAN 9, 2000, 11:48 AM — Page: 1

Data Set	Type	Capacity	Entries	Load Factor	Secondaries	Max Blks (Highwater)	Blk Fact
Part-Master	Ato	10000	4305	43.0%	14.2%	16	78
Part-Loc-Master	Ato	606010	303005	50.0%	27.7%	95	67
Parts-Detail	Det	303030	303005	100.0%	(303005)		63

Search Field	Max Chain	Ave Chain	Std Dev	Expd Blocks	Avg Blocks	Ineff Ptrs	Elong-ation
Part-No	3	1.17	0.40	1.00	1.67	63.3%	1.67
Part-Loc-No	13	1.38	0.77	1.00	1.42	29.3%	1.42
Part-No	41065	70.38	666.46	1.86	3.20	3.1%	1.72
!Part-Loc-No	1	1.00	0.00	1.00	1.00	0.0%	1.00

23

You could change the field data-type from 8-byte integer (I4) to 8-byte character (Z8). The field would still take up the same amount of space. The key values would remain the same, and they would be stored as binary data. To do this, you could use Adager's Redefine Item command or DBGeneral's Data Item Rename option in section 5.1.

With this approach, application programs do not have to be changed at all. Their definition is still valid. Utility tools that rely on root file information (like Suprtool), however, will not know about this change. As a result, they will have difficulty working with the modified field, unless there is a way to redefine field characteristics (e.g., Suprtool's Define command).

However, this approach does get IMAGE/SQL to use the character-type hashing algorithm. The values go through a different calculation sequence that assigns them different record locations.

The figures are dramatically different:

Secondaries	Max Blocks	Max Chain	Avg Chain	Avg Blocks
86.4%	612	4305	7.38	45.84
27.7%	95	13	1.38	1.42

For Techies

References

Solution #2: Change key type to byte and convert data values

HowMessy/XL (Version 2.6) for IMAGE/3000 databases — Data Base: SOMEDB.DATA.SOMEACCT — By Robelle Solutions Technology Inc. — Run on: SUN, JAN 9, 2000, 11:48 AM — Page: 1

Data Set	Type	Capacity	Entries	Load Factor	Secondaries	Max Blks (Highwater)	Blk Fact
Part-Master	Ato	10000	4305	43.0%	14.2%	16	78
Part-Loc-Master	Ato	606010	303005	50.0%	21.4%	0	56
Parts-Detail	Det	303054	303005	100.0%	(303005)		53

Search Field	Max Chain	Ave Chain	Std Dev	Expd Blocks	Avg Blocks	Ineff Ptrs	Elong-ation
Part-No	3	1.17	0.40	1.00	1.67	63.3%	1.67
Part-Loc-No	6	1.27	0.54	1.00	1.00	0.0%	1.00
Part-No	41065	70.38	666.46	2.07	3.43	3.4%	1.66
!Part-Loc-No	1	1.00	0.00	1.00	1.00	0.0%	1.00

24

You could change the data-type to character (Z18) and convert the data from binary format to a string of numeric digits. The "new" field occupies more space because an 8-byte integer can contain a very large value. This also explains why the Blocking Factor is now 56 instead of 67.

To accomplish this, you could use Adager's Change Item command or DBGeneral's Change Item option in section 5.1.

Application programs have to be changed and recompiled to match the new field characteristics (type and length). This obviously requires much more planning and work from the development group.

Even though there are less records in each block, the report figures are looking a bit better. This improvement is probably due to increased field length, which gives the hashing algorithm more bits to work with.

The differences between the three reports are now:

Secondaries	Max Blocks	Max Chain	Avg Chain	Avg Blocks
86.4%	612	4305	7.38	45.84
27.7%	95	13	1.38	1.42
21.4%	0	6	1.27	1.00

For Techies

References

HowMessy sample report (detail dataset)

HowMessy/XL (Version 2.6) for IMAGE/3000 databases

Data Base: STORE.DATA.INVENT
By Robelle Solutions Technology Inc.

Run on: SUN, JAN 9, 2000, 11:48 AM
Page: 1

Data Set	Type	Capacity	Entries	Load Factor	Secon- daries Blks (Highwater)	Max Blk Fact
M-Customer	Man	248113	178018	71.7%	30.5% 1496	1
A-Order-No	Ato	126673	768556	60.7%	25.7% 1	70
D-Orders	Det	1000000	768556	76.9%	(851445)	12
D-Ord-Items	Det	4000000	3458511	86.5%	(3470097)	23

Search Field	Max Chain	Ave Chain	Std Dev	Expd Blocks	Avg Blocks	Ineff Ptrs	Elong- ation
Customer-No	22	1.92	0.32	1.00	1.90	90.5%	1.90
Order-No	10	1.35	0.62	1.00	1.00	0.0%	1.00
!Order-No	1	1.00	0	1.00	1.00	0.0%	1.00
S Customer-No	80	14.34	17.76	1.75	9.20	57.2%	5.25
S !Order-No	1604	8.06	35.75	1.36	11.32	72.5%	8.34

25

For detail datasets, the *Highwater* column indicates the highest record location used so far.

The *Blocking Factor* column is the same as for master datasets. It indicates the number of records stored in each block.

The *Search Field* column indicates the name of each item linked to a master dataset. This is also known as a path in a dataset. HowMessy prints one line for each path, including its corresponding statistics. A search item is sometimes preceded by an exclamation mark (!), the letter *S*, or both. An exclamation mark indicates the primary path. The letter *S* indicates a sorted path. We will discuss these two indicators on a later slide.

Maximum Chain and *Average Chain* pertain to the number of records that have the same search item value. They indicate the longest chain and the average chain length, respectively.

The *Standard Deviation*, *Expected Blocks*, and *Average Blocks* columns have the same meaning as for master datasets.

Similarly, *Inefficient Pointers* indicate the percentage of chains that span block boundaries.

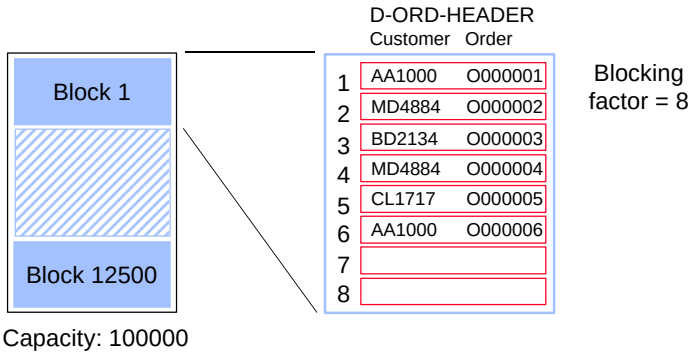
Elongation has the same meaning for detail datasets as for master datasets: the average number of blocks divided by the expected number of blocks.

For Techies

References

Empty detail dataset

- Records are stored in the order they are created, starting from record 1
- Records for the same customer are linked together using pointers to form a chain
- Chains are linked to the corresponding master entry



26

When a detail dataset is empty, IMAGE/SQL adds records in the order that they are created, starting from record location one.

Records are stored in chronological sequence. That means records for the same customer are not necessarily stored consecutively, unless they happened to be added at the same time.

However, assuming that customer id is a key, then records with the same customer id are linked together with pointers to form a chain. The first and last records in a chain are also linked from the corresponding master record for that customer id.

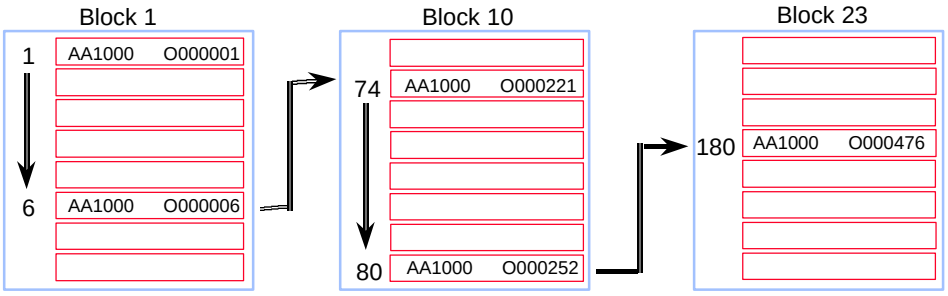
These chains by customer id are used when you do retrievals by customer id. IMAGE/SQL finds the matching entry in the master dataset, perhaps following the synonym chain (explained much earlier!), then follows the detail chain for this **path**.

For Techies

References

Detail chains get scattered

- Over time, records for the same customer are scattered over multiple blocks



27

As IMAGE/SQL adds more records to the dataset, the chains for any given key value get longer and span more blocks. The more active the dataset is, in terms of additions and deletions, the more likely that adjacent records on a chain will be in different blocks.

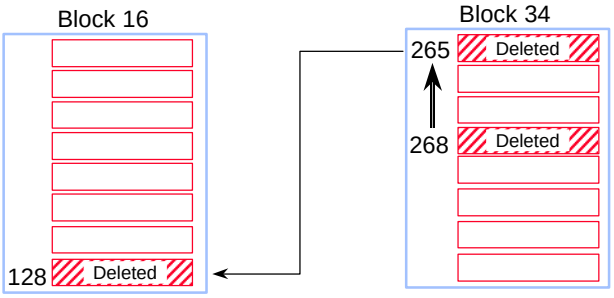
As a result, IMAGE/SQL has to perform a lot of extra disc I/Os to read a whole chain for a given customer. In the worst case scenario (but a common one for active datasets), IMAGE/SQL does one disc I/O for each record in the chain.

For Techies

References

Delete chain

- Deleted records are linked together
- IMAGE/SQL reuses the locations in the delete chain, if there are any



28

IMAGE/SQL adds records to an empty dataset in a sequential fashion. However, when IMAGE/SQL deletes a record, that location is opened up for reuse by a new record. IMAGE/SQL manages this reusable space by linking all deleted locations together to form what is known as the **delete chain.**

To conserve space, IMAGE/SQL reuses locations on the delete chain before assigning a new location that has never been used. It does so by going backward along the delete chain. In other words, the last entry that IMAGE/SQL deletes is the first entry it reuses.

Only when the delete chain is empty does IMAGE/SQL allocate a new location from the available space at the end of the dataset.

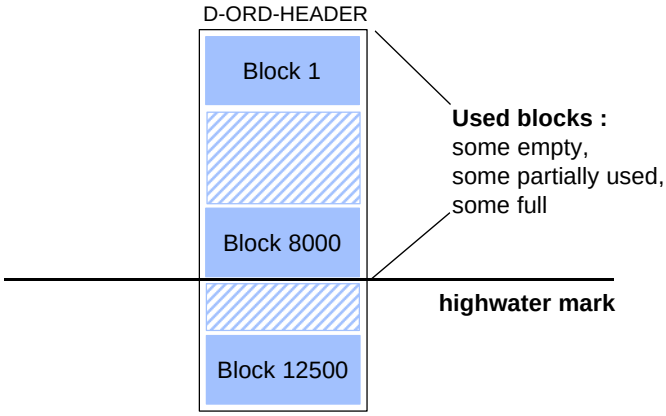
The delete chain does not differ from any other chain. If there are many deleted records, they are dispersed throughout the dataset. When you add or delete a record, IMAGE/SQL needs to do a few additional disc I/Os to maintain the delete chain.

For Techies

References

Highwater mark

- Indicates highest record location used so far
- Serial reads scan the dataset up to the highwater mark



29

IMAGE/SQL keeps track of the highest record number ever used. This record number is known as the *highwater mark*. The record may no longer be there (e.g., it might have been deleted). For IMAGE/SQL, the important thing is that record number has been used once.

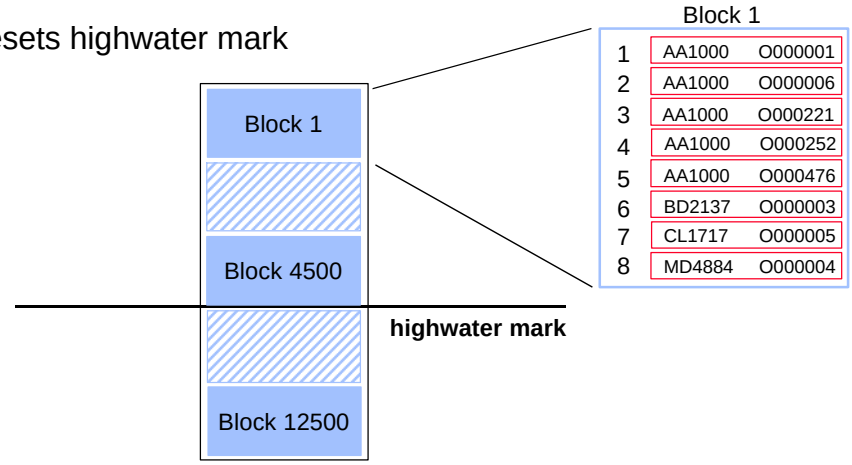
In an active dataset, many records are deleted. As a result, the current number of records is often a lot less than what can be stored between the beginning of the dataset and the highwater mark. Nevertheless, serial scans still read up to the highwater mark, which means IMAGE/SQL ends up performing a lot more I/Os than if all the records occupied consecutive locations at the beginning of the dataset.

For Techies

References

Repacking a detail dataset

- Groups records along primary path
- Removes delete chain (no holes)
- Resets highwater mark



30

You can often improve performance by repacking a detail dataset. The repacking process usually groups all records along their **primary path** values. In other words, all the records for a customer are put into consecutive locations (if customer id is the primary path). Depending on the chain length, a single disc I/O may read all the appropriate records.

The repacking default in Adager's Detpack and DBGeneral's dataset reorganization feature (section 3.6) is along the primary path. For various reasons, you may not want or be able to change the primary path. If that is the case, both products give you the option to use any other path in the dataset.

Repacking removes the delete chain. It groups all the records at the beginning of the dataset without leaving empty locations in the middle of blocks. Repacking also resets the highwater mark. The last record now becomes the new highwater mark, and serial scans no longer have to do extra disc I/Os.

If you are tight on time, you can condense the dataset instead. Although this function does compact the delete chain, it does not reorganize individual chains. This should improve serial scans without affecting chained reads. To condense a dataset, select the Serial Repack option in Adager or the Condense option in DBGeneral.

For Techies

References

Interpreting detail dataset lines

- Pay attention to the following statistics:
  - Load Factor approaching 100% (dataset full)
  - Primary path (high Average Chain; accessed often)
  - High Average Chain and low Standard Deviation, especially with a sorted path (Is path really needed?)
  - High Inefficient Pointers (entries in chain not consecutive)
  - High Elongation (entries in chain not consecutive)

31

Unlike in master datasets, the Load Factor in detail datasets does not have an impact on performance. It should only be a concern if the Load Factor is getting close to 100%.

Because most tools repack a dataset along the primary path, it is essential to pick the right one. A primary path should be the one with the longest Average Chain that is accessed the most frequently. There is nothing to gain by assigning it to a path with an Average Chain of 1. Repacking only improves statistics for the specific path. Other paths will probably show poor numbers.

Keep an eye on paths with a high Average Chain length and a low Standard Deviation (which means that most chains are close to the average). When IMAGE/SQL tries to read these records, their chains require a lot of disc I/Os. Each path adds overhead during create and delete operations. Ask yourself if the overhead is worth it. Remember, sorted paths require more I/Os. It might be as fast, or faster, to do a serial scan to retrieve these records.

A high percentage of Inefficient Pointers or a high Elongation value indicate that records in the chain are spanning block boundaries.

For Techies

References

Report on d-orders

- Primary path should be on customer-no, not on order-no
- Highwater mark is high
- Repack along new primary path regularly

Data Set	Type	Capacity	Entries	Load Factor	Secon-Max daries Blks (Highwater)	Blk Fact
D-Orders	Det	1000000	768556	76.9%	(851445)	12

Search Field	Max Chain	Ave Chain	Std Dev	Expd Blocks	Avg Blocks	Ineff Ptrs	Elong-ation
!Order-No	1	1.00	0	1.00	1.00	0.0%	1.00
S Customer-No	80	14.34	17.76	1.75	9.20	57.2%	5.25

32

The primary path is on order-no, which has an Average and Maximum Chain length of 1. The primary path should actually be on customer-no, even though order-no is accessed most often.

The highwater mark is high. IMAGE/SQL is reading more than 6,900 extra blocks on serial scans.

The path on customer-no can also be improved. Average Blocks are much higher than expected. Inefficient Pointers and Elongation are also fairly high.

This dataset would benefit from a change in the primary path and a repacking.

Use sorted paths with caution because they can easily degrade performance.

For Techies

References

Report on d-ord-items

- Inefficient Pointers and Elongation are high
- Highwater mark is fairly high
- Repack the dataset regularly
- Is the sorted path really needed?

Data Set	Type	Capacity	Entries	Load Factor	Secon- daries	Max Blks (Highwater)	Blk Fact
D-Ord-Items	Det	4000000	3458511	86.5%	(	3470097)	23

Search Field	Max Chain	Ave Chain	Std Dev	Expd Blocks	Avg Blocks	Ineff Ptrs	Elong- ation
S !Order-No	1604	8.06	35.75	1.36	11.32	72.5	8.34

33

The highwater mark is a bit high. Inefficient Pointers and Elongation are really high. This dataset needs to be repacked on a regular basis. Again, evaluate the usefulness of the sorted path.

For Techies

References

DBGeneral detail dataset report

- "Detail Dataset Path Information" section
- More examples of inefficient paths

				data chains			chain pointers			
	dataset/master path	blk fct	S ?	number chains	avg length	max length	std dev	avg pages	% <1P	% inef
39	SPLITDET	85								
	1 !REMAST		N	1832	2	84	.1	1.3	28	34
	2 REMAST		N	2704	2	84	.1	1.2	41	77
	delete chain is 48.5% of entry count					1961		5.8	51	49
40	ADJHIST	51								
	1 !PARCAUTO		Y	1342	3	93	.2	2.4	56	140*
	2 PARCAUTO		Y	1067	3	93	.2	2.2	36	78
	delete chain is 6.4% of entry count					302		10.4	48	52

34

We once again see how the DBGeneral report is similar to the HowMessy report. The *number of chains*, *avg length*, *max length*, and *std dev* correspond to the similar columns in the HowMessy report. DBGeneral also provides the number of chains for each path.

While HowMessy reports efficiency in blocks, DBGeneral does so in 4,096-byte pages. The *avg pages* is the average number of pages that each chain in the path occupies. The *%<1P* is the percentage of chains that occupy a single page (retrieving these chains does not cause any extra disc I/O). The *%inef* is the total percentage of chain pointers that span a page (i.e., are inefficient).

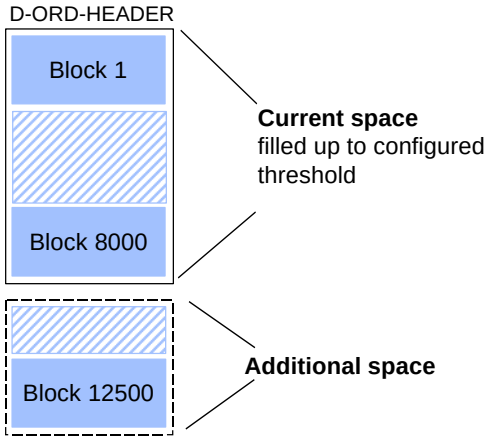
DBGeneral also provides information on the delete chain.

For Techies

References

Dynamic dataset expansion (details - DDX)

- Allocates more disc space as dataset fills up



35

With “regular” detail datasets, you can add entries and consume all the available blocks. Once the dataset is full, you cannot add new entries until you increase the capacity, which requires exclusive access. It might be hard to schedule downtime to change the capacity.

Use dynamic dataset expansion for detail datasets to help manage disc space. In this case, IMAGE/SQL allocates additional disc space whenever the dataset reaches a predefined threshold. This is identical to changing the capacity. It does this dynamically without any intervention and without shutting down the database.

When defining a DDX dataset, you can specify a number of parameters:

- the threshold (e.g., load factor greater than 85%)
- the absolute maximum capacity
- the amount of space allocated during an expansion

Caveats: Once new space is allocated, it is never returned even if it becomes empty in the future. Don’t make the threshold too low, or additional space too small, to avoid frequent expansion. Don’t make the maximum capacity too high or a runaway applications may eat up all the available disc space.

For Techies

References

Detail dataset solutions

- Assign the primary path correctly; select a search item with an Average Chain length > 1 that is accessed most often
- Repack datasets along the primary path regularly
- Increase the Blocking Factor
  - Increase block size
  - Reduce record size
- Understand sorted paths
- Check your databases early in the design; use HowMessy on test databases

36

As mentioned earlier, the primary path should have an Average Chain length greater than 1, and it should be the most frequently accessed path. If there is no explicitly defined primary path, DBSCHEMA uses the first path in a dataset.

Repack active datasets on a regular basis. How often you repack them varies from one dataset to another.

As for master datasets, try to increase the Blocking Factor. You can increase the block size by using the Blockmax option in DBSCHEMA or by using one of the various database tools available. The other option is to reduce the size of each record. To do this, use different data-types (I, J, or P instead of Z for storing numeric values) or assign the correct maximum number of characters in each field. That is, do not use a 50-character field when a 30-character field is sufficient. The idea is to save space.

Sorted paths can adversely affect performance. You should try to understand how they work. Use them with caution.

It is easier to change a database structure when it is still in the design stage. Check your structure as early as possible. Ideally, you should try to simulate the production environment, then use HowMessy to look at the results.

For Techies

References

B-Trees

- Indexed-sequential keys for IMAGE datasets
- Adds a .idx KSAM file with filecode -412
- Performance may decline as trees are heavily populated
- No performance levers known to us, except buy more RAM!

37

HP added B-Trees in the C.07.10 version of IMAGE (MPE/iX 5.5 Express 4 release). This provides indexed sequential indexing for IMAGE datasets, with the indices stored in a KSAM/iX file. The index file is kept in the Posix name space with the same name as the dataset, plus a `.idx` extension. The KSAM file is privileged with a filecode of -412. Therefore, the size limit for this B-tree index file is 4 GB.

IMAGE incurs the overhead of maintaining the various index records during the DBPUT and DBDELETE calls on the dataset. DBFIND also incurs overhead for reading the KSAM file to search down the B-tree.

The type of indexing is determined by the Mode parameter on DBFIND. A mode 1 DBFIND does a normal primary path lookup via the master dataset, as if there were no B-trees (Unless BTREEMODE is set to ON in DBUTIL or by calling DBCONTROL, then mode 1 means a B-tree search). Other modes are 4, which allows a B-tree index search on numeric fields, 10 which is a standard IMAGE DBFIND regardless of the BTREEMODE setting, 21 and 24 which are the same as 1 and 4 but do not return the number of entries (so are faster).

KSAM/iX has been known to slow down as a particular index tree becomes heavily populated. Results will, however, depend upon the key data in individual cases. We are not aware of any levers you can turn to improve the performance of B-trees, except perhaps to add more RAM to your system!

For Techies

References

Minimum number of disc I/Os

Intrinsic	Disc I/Os
DBGET	1
DBFIND	1
DBBEGIN	1
DBEND	1
DBUPDATE	1 (non-critical item)
DBUPDATE	13 (critical item)
DBPUT	3 [+ (4 x #paths, if detail)]
DBDELETE	2 [+ (4 x #paths, if detail)]
Serial reads:	
Master	Capacity / Blocking factor
Detail	# entries / Blocking factor

38

This table shows how much I/O would be required on an ideal database.

DBUPDATE on a critical item (search item or sort field), DBPUT, and DBDELETE require a lot of I/Os because pointers must be updated in the corresponding master dataset, as well as the previous and the next records in the chain.

Serial scans on master datasets must read the whole file because records are added anywhere in the allocated blocks, in other words randomly.

IMAGE/SQL reads to the highwater mark on all detail datasets. On detail datasets that have been packed, the number of current records is the same as the highwater mark. Thus serial scans only read the minimum number of blocks.

For Techies

References

Automating HowMessy analysis

- Recent version of HowMessy creates a self-describing file with these statistics
- Process the file with generic tools (Suprtool) or custom programs (COBOL, 4GL), and produce custom reports
- Send messages to database administrators
- Write “smart” job to fix databases without user intervention

39

Recent versions of HowMessy create a self-describing file with all the statistics shown on the report. A self-describing file is an ordinary MPE file with information about the fields in the user labels. DBGeneral’s diagnostics functions (section 2) also write the information to a file. The only difference is the file is not self-describing.

That means you can write a program using your favorite programming language (e.g., COBOL, PowerHouse, etc.) to analyze the data and to report only the lines that interest you. You can use utility programs like Suprtool and AskPlus to do similar tasks. These statistics files also allow you to keep historical data so that you can analyze trends.

You can even write smart job streams that automatically fix the database (e.g., repack detail datasets) and send messages to the database administrator about any changes. For an example job that selects HowMessy items that may require attention and actually emails them to the Data Base Administrator, see this article:

`http://www.robelle.com/tips/remote-dba.html`

For Techies

References

Contact Information

- Robelle. — 1-604-582-1700, support@robelle.com, www.robelle.com
- Adager — 1-208-726-9100, support@adager.com, www.adager.com
- Bradmark — 1-800-621-2808, techsupp@bradmark.com, www.bradmark.com

40

Robelle Solutions Technology Inc.
Suite 201, 15399 - 102A Avenue
Surrey, B.C. Canada V3R 7K1
Telephone: 604.582.1700
Fax: 604.582.1799
E-mail: support@robelle.com
Web: www.robelle.com

Adager Corporation
Sun Valley, Idaho 83353-3000
U.S.A.
Telephone: (208) 726-9100

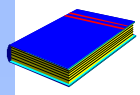
support@adager.com
www.adager.com

Bradmark, Inc.

4265 San Felipe, Suite 800
Houston, Texas 77027
U.S.A.
Telephone: 713-621-1639
Fax: (713) 621-1639
E-mail: techsupp@bradmark.com
Web: www.bradmark.com

For Techies

References



Summary

- IMAGE/SQL databases become messy over time, especially if they are active
- HowMessy (via Robelle or Adager) and DBGeneral let you analyze a database's efficiency
- You should have some knowledge of the internal workings of IMAGE/SQL
- Monitor your databases regularly

41

Unless they are static, all IMAGE/SQL databases eventually become inefficient. HowMessy (whether you got it from Robelle or Adager) and DBGeneral provide reports that let you analyze the efficiency of your databases. Understanding the report is easy, once you understand some of the internals of IMAGE/SQL. Then by monitoring your databases regularly, you can make sure that you get the best performance from your IMAGE/SQL applications.

For Techies

References