



# Database Update

Vikram Kumar B T

CSY

Aug 2001

Enhancements  
included in  
6.5 Exp. 2  
&  
7.0 Exp. 1

- Increase limits of a database
- Support of dataset size > 80GB
- Support find by record number in QUERY
- Native QUERY allows runtime PARM=%777
- Support ANSI SQL AS clause in select statement

## Increase limits of a database

- Increase number of datasets per database to 240
- Increase number of items per database to 1200
- Increase number of paths (from master to detail) to 64

- New version of DBSCHEMA allows higher limits
  - Rootfile version will be 'C'5 if datasets/items/paths exceed old limits
  - Applications need to increase buffer size for DBINFO mode 103, 203, 204 and 301
  - For debugging purpose, DBUTIL flag available  
>>ENABLE basename FOR OLDINFOLIMITS
  - DBCONTROL mode 20 allows applications to negate the DBUTIL setting in the specific DBOPEN
- 
- Old apps, DB has old limit : continue to work
  - Old apps, DB has higher limits :data MAY overrun the buffer, apps MAY abort
  - Old apps, DB has higher limits, flag set :DBINFO returns error -270
- <<new apps : increase buffer, call DBCONTROL 20>>
- New apps, DB has old limit : continue to work
  - New apps, DB has higher limits : DBINFO returns larger buffer
  - New apps. DB has higher limits, flag set : DBINFO returns lager buffer

## Support of dataset size > 80 GB

Using 24bit/8bit record name format as the internal pointer limits the maximum dataset size to 80GB. Changing record name format to record number format allows dataset size greater than 80GB.

- Use record number instead of record name as the internal pointers
- Need to specify \$CONTROL LARGESET in DBSCHEMA to create a database in record number format
- Rootfile version will be 'C'6 to denote the database is in record number format
- Only one format allowed for a given database
- Use DBBIGSET.PUB.SYS to migrate databases
- DBINFO mode 406 tells what are the features used
- DBUTIL >>SHOW basename ALL will display

DBINFO mode 406, 17th element:  
bit 0..15 where 15 is the rightmost bit  
bit 9 : higher limit  
bit 10: record number format  
bit 11: not use  
bit 12: MDX  
bit 13: B-tree  
bit 14: Jumbo  
bit 15: DDX

Rootfile version in Rootfile record 0  
'C'2 -- base --- C.05.XX and before  
'C'3 -- jumbo --- C.06.XX  
'C'4 -- b-tree --- C.07.XX  
'C'5 -- limit --- C.09.XX  
'C'6 -- record number --- C.09.XX

## Find by Record number in QUERY

- Record number is preceded by a # sign
- Default record number is a decimal number
- Record number can be an octal (preceded by %) or a hexadecimal value (preceded by \$)
- Examples: to read the fifteenth record in the dataset INVOICES
  - FIND INVOICES.#15
  - FIND INVOICES.##%17
  - FIND INVOICES.##\$f

User can find a specific record by giving a record number after the dataset name  
>FIND datasetname.#recordnumber

QUELXD6 Oct 2000 D.03.17

Also Support for new IMAGE Limits

## Display percent completion in QUERY

- New command VERBOSE to enable the report
- New command TERSE to disable the report (this is default)
- Enhance command SHOW to show VERBOSE and SHOW ALL to display all options
- Use SETVAR HP\_QUERY\_PROGRESS\_INTERVAL nnn to set the time interval. The value nnn is 1 to 65000 in seconds, with the default is 30 seconds
- Patch QUELXJ0 released

Query/iX will give a progress report during long database retrievals for commands such as FIND, SUBSET or MULTIFIND:

```
aaa ENTRIES AFTER bbb RECORDS OUT OF ccc IN  
STEP ddd OF eee
```

## Other QUERY Patches

- QUELXE9 Dec 2000 D.03.18  
Business Basic Floating Decimal Fix

- QUELXJ0 Feb 2001 D.03.19  
Progress Reporting Enhancement  
VERBOSE/TERSE Commands  
SHOW Improved - All option  
Performance enhancements

- QUELXP5 Apr 2001 D.03.20

Report Reals trailing zero fix

QUERYCM is no longer being updated (still 3.17) and will be deleted in the future.

**Query has done a FIND of 16,700,000 records (the FIND Limit) on a 979-100 under MPE/iX 6.5 with Jamaica discs in under 14 minutes. (MUSIC database)**

## Enhancements ready to test/ submit

- Large file dataset
- TurboIMAGE scalability II
- Increase Allbase limits
- Allow one store procedure to call another store procedure
- IMAGE/SQL NOAUTOs
  
- (Allbase Auto-increment)

## Large file data set

- \$CONTROL LFDS in DBSCHEMA to create large file dataset, which is default
- May need \$CONTROL LARGESET if the capacity is too big
- The maximum large file dataset size is 128GB, if exceeds, need to make the dataset jumbo
- Large file dataset and jumbo dataset cannot co-exist in the same database
- Rootfile version will be 'C'7 and bit 8 will be true if database has at least one large file dataset
- Basic migration tool : DBLOAD/DBUNLOAD to disk

For a dataset larger than 4GB, use MPE large file feature instead of jumbo dataset.

# TurboIMAGE scalability II

## Put/Delete semaphore

- Used to serialize DBPUT/DBDELETE/DBUPDATE activities
- One semaphore per database

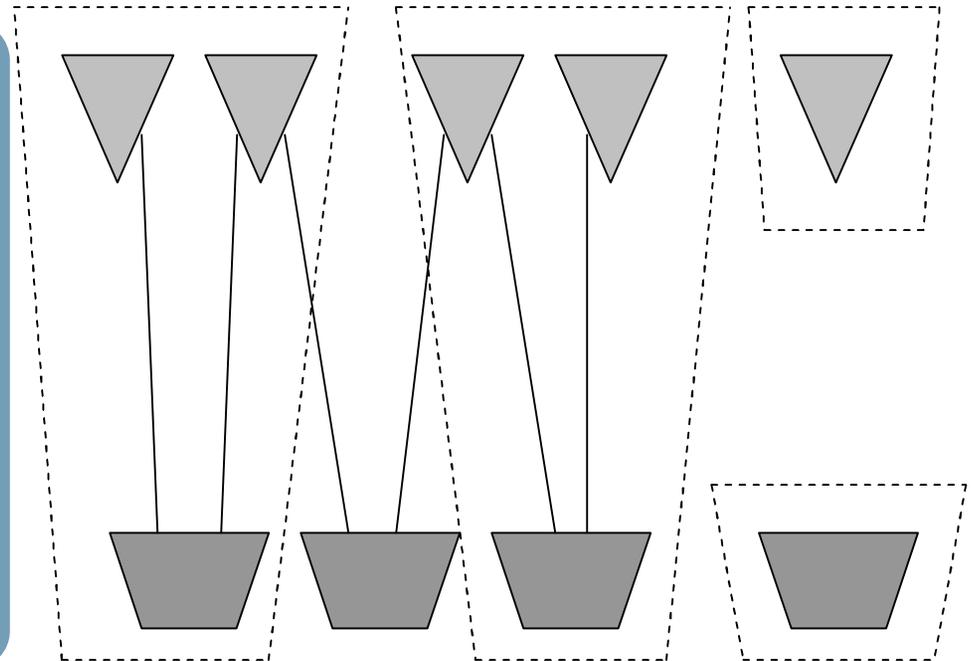
## Usage of put/delete semaphore

- Control the modification to the dataset file label
- Cover XM rollback at intrinsic level
- Avoid deadlock between data block locks
- Manage dynamic dataset expansion

## DSEM

- Group related datasets together
- Different group can be modified concurrently

Divide Put/Delete semaphore down to block level



# TurboIMAGE scalability II

- Lock dependency semaphores (one for each dataset) one at a time
  - lock all the necessary data blocks
  - release the dependency semaphore
- Release all the data block locks at end of intrinsic
- EHWM introduced for detail dataset
  - EHWM resides in dataset user label, starts from 15th double word (1 base)
  - each EHWM has 21 entries, each entry represents a block, with the 1st entry as a header
  - each entry occupies 8 bytes
  - allocated when the first DBPUT (if EHWM enabled) access the dataset
  - deallocate when user disable EHWM
- DBUTIL
  - >> ENABLE basename FOR EHWM

## 0th entry

1st	2nd	3rd	4th	5th	6th	7th	8th
E	H	W	M	Blocking Factor		No. of Entries	

·  
·  
·

## Nth entry

1st	2nd	3rd	4th	5th	6th	7th	8th
<b>Block number</b>				<b>PIN</b>		<b>No. of records used in block</b>	

## Increase Allbase limits

- Increase number of pages for runtime control block from 2000 to 6000
- Increase number of concurrent transactions from 250 to 750

***Allbase/SQL H0***

***Under beta testing***

# SP Calling SP

```
CREATE PROCEDURE [Owner.]ProcedureName [LANG = languagename]
```

```
[(ParameterName ParameterDataType
```

```
CREATE PROCEDURE [Owner.]ProcedureName [LANG = ProcLangName]
```

```
[(ParameterDeclaration [, ParameterDeclaration] [...])]
```

```
[WITH RESULT ResultDeclaration [, ResultDeclaration] [...]]
```

```
AS BEGIN
```

```
    ProcedureStatement;
```

```
EXECUTE PROCEDURE [Owner.]ProcedureName
```

```
[(ActualParameter) [,...]]
```

```
where ActualParameter =
```

```
    [ParameterName = ] ParamaterValue [OUTPUT [ONLY]]
```

```
[...] END ;
```

```
[IN DBEFileSetName]
```

## SP Calling SP

where ParameterDeclaration =

```
ParameterName ParameterType [LANG = ParameterLanguage]  
[DEFAULT DefaultValue] [NOT NULL] [OUTPUT]
```

where ResultDeclaration =

```
ResultType [LANG = ResultLanguage] [NOT NULL]
```

Example:

```
CREATE PROCEDURE ReportMonitor (PartNumber CHAR(20) ) AS  
BEGIN  
    EXECUTE PROCEDURE RemoveParts(:PartNumber);  
    RETURN ::sqlcode ;  
END ;
```

Constraints : Return Value from Execute Procedure within the Create Procedure can't be assigned to variable.

## Extra enhancement

- For auditing from the log file
- Users use the same logon with different session name
- No userident being passed through password/userident parameter of DBOPEN call
- DBUTIL flag  
>>ENABLE basename FOR FORCESESSION

Use session name as user identifier in the log file to help the auditors to differentiate users with the same logon.

# IMAGE/SQL NOAUTOs

- ATTACH [WITH OWNER] [noauto/ auto] [noautosplit/ autosplit]
- Automatic masters will not be attached in IMAGE/SQL 'ATTACH' command
- Automatic split of compound items will not be done

- Lab testing complete
- Call for beta testing
- Release vehicle to be identified
- NO AUTOVIEWS investigated
- Require Allbase DBCORE changes
- Not addressed now

Investigation  
for  
making TurboIMAGE  
thread safe

(TOP SIB ITEM)

Three requests included:

- make TurboIMAGE thread aware and thread safe
- make TurboIMAGE forkable
- pass 'base ID' between processes

## Thread Characteristics:

- Thread has its own PIN
- Thread has its own stack
- Thread shares SR5 space

## Problems for not being thread safe:

- Runtime control block
- Global variables
- Open files

### • Runtime Control Block (DBUX/DBU)

- One per DBOPEN
- transaction information
- locking information
- logging information
- current record pointers
- opened files
- trailer area

### • Global Variables

- qllock\_trace
- bti\_global
- ccu\_global
- chunk\_control

### • Open Files

- except rootfile, not using file system intrinsics
- maintain own current record pointers
- use SMCB

## **Fork() Characteristics:**

- **Forked process inherits characters from its parent**
- **Forked process has its own SR5**
- **Duplicate some file system data structures and share some**

## **Problems for forking:**

- **fork() will fail if DB opened**
- **Runtime control block**
- **Global variables**
- **Open files**

### • **Runtime Control Block (DBUX/DBU)**

- Not able to duplicate file system data structure of DBUX and DBU
- Many fields not suitable for sharing
- State of the process while forking

### • **Global Variables**

- qllock\_trace
- bti\_global
- ccu\_global
- chunk\_control

### • **Open Files**

- except rootfile, not using file system intrinsics
- maintain own current record pointers
- use SMCB

## Pass 'base ID' between processes:

- threads
- forking/forked processes
- father/son processes
- unrelated processes

## •Pass 'base ID' means share DBU

- What if the receiving process already has the same DB opened?
- Who is the owner?
- What if process terminated?
- How to handle user logging?

## What is 'base ID'?

- An index into an array of DBUs this process has opened
- Array resides in DBUX

## Options for Runtime Control Block

- Copying DBUX and DBU
- Sharing DBUX and DBU

### •Pros for Copying

- Separate current record pointers
- Distinct XM related data structures
- 'accessor entry' in DBG matches with no. of DBU
- Separate DBU trailer area
- Same process termination procedure

### •Cons for Copying

- Time consuming to duplicate DBUs
- May have space problem in SR5 for threads
- May conflict the "inherit" concept

## Options for Runtime Control Block

- Copying DBUX and DBU
- Sharing DBUX and DBU

### •Pros for Sharing

- The key concept for thread is sharing, so does DBU
- Share DBU will share open files
- Share DBU will share current record pointers

### •Cons for Sharing

- Need mechanism to control the currency of XM and locking in DBU
- Need space control procedures to handle DBU trailer area
- Need extra space for log record belongs to different PIN
- Break the coherency among the DBOPEN, "accessor entry" and DBU

## **Feedback:**

- If share, it shares current record pointer too. It is programmer's responsibility to handle it correctly
- For thread, concurrency is the key. Lock out other thread for the duration of a whole intrinsic is not acceptable
- To make fork() work is very important too

## **Recommendation:**

- Passing "base ID" between any two processes is too vague and ambitious
- If we do decide to implement, may implement in phases
- Thread should share DBUX/DBU
- Fork() should duplicate the DBUX/DBU