W12

Performance Tuning Informix Database Engine

May 7–11, 2006

Tampa Convention Center

Tampa, Florida, USA

IDUG® 2006

North America

John F. Miller III
IBM

Sunday, May 7, 2006 • 8:30 a.m. – 12:00 a.m.

GoFurther

INTERNATIONAL DB2 USERS GROUP

Agenda

- Defining the problem
- Where to optimize
- Application Optimization Examples
- Understanding Database Server Optimization
- Understanding Fragmentation
- Q & A.

Factors Affecting Performance

- **Speed of the Application**
- **Speed of the Network**
- **Speed of the Database Server**
- **Speed of the Hardware**
- **Type of Application**
- **Human Perception**
- Performance is relative!

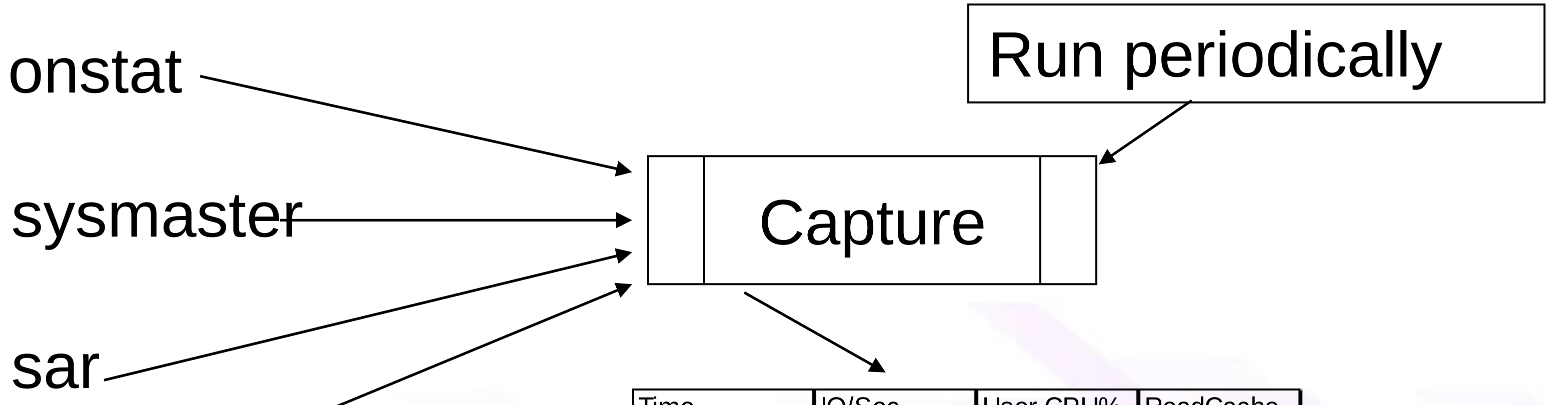
Two Major Types of Performance Issues

- Improve my current system?
- Why did my system suddenly slow down?

Create a Collection of Performance Information

- Understand your system characteristics
- Periodically capture key information
  - Include information from engine
    - onstat –p, onstat –g iof, etc.
    - Sysmaster database
  - Include information from the OS
    - sar, iostat, vmstat, mpstat, etc

Creating a Base Line



Time	IO/Sec	User CPU%	ReadCache
9/1/2002 8:00	41	0.23	99.6
9/1/2002 9:00	63.3	0.41	98.2
9/1/2002 10:00	92.34	0.61	98.3
9/1/2002 11:00	102.2	0.63	98.1
9/1/2002 12:00	50.32	0.55	97.3
9/1/2002 13:00	100.23	0.71	98.1
9/1/2002 14:00	104.2	0.72	98.5

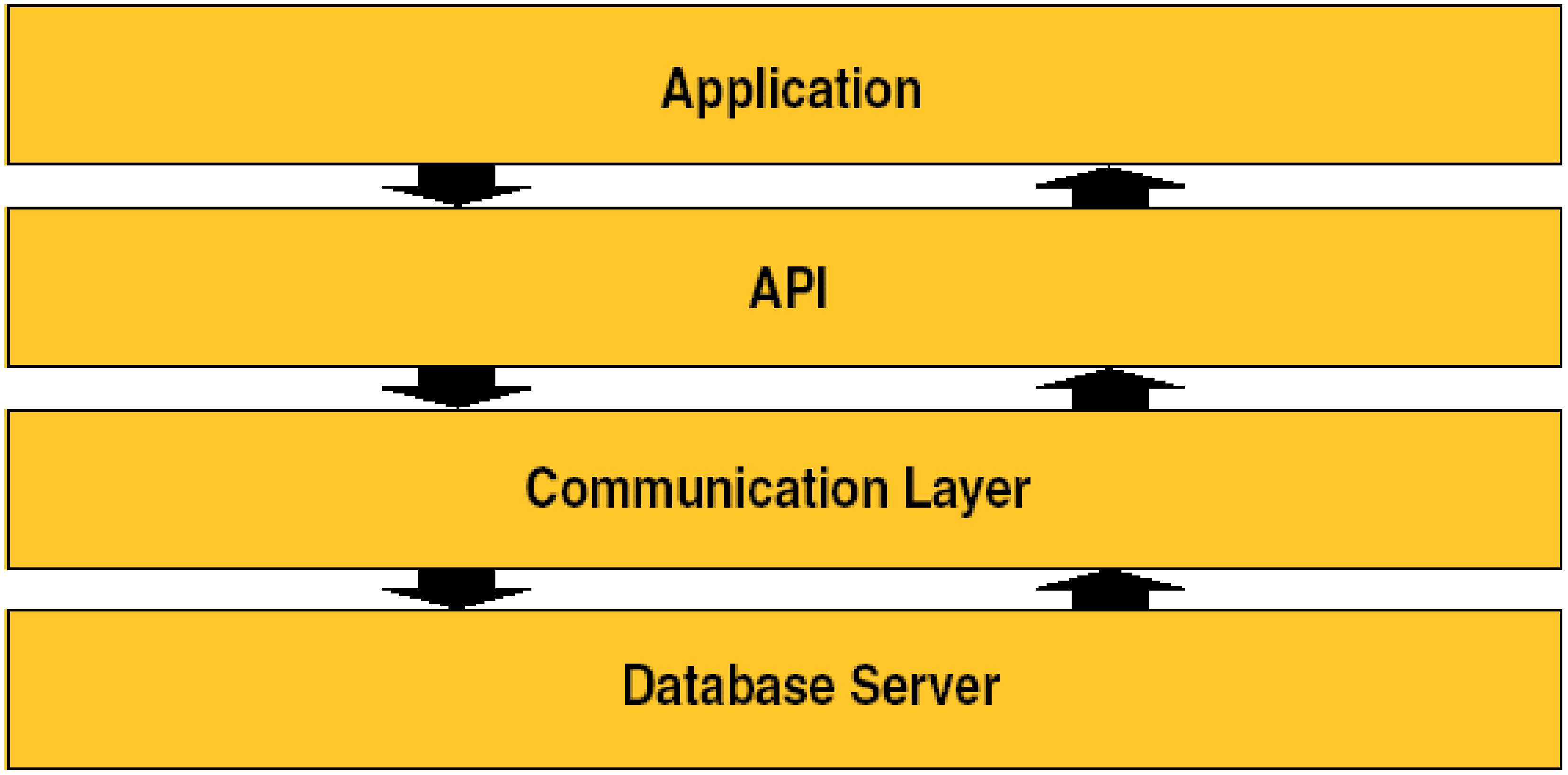
Analysis

Capture information now. Try to have several weeks of data before you need it so that trends and cycles can be understood.

OS Capture

- At a minimum you need information about
  - IO/sec
  - Processor utilization
  - Memory Utilization
- Consider using operating tools
  - sar –u
  - vmstat
  - iostat

Candidates for Optimization

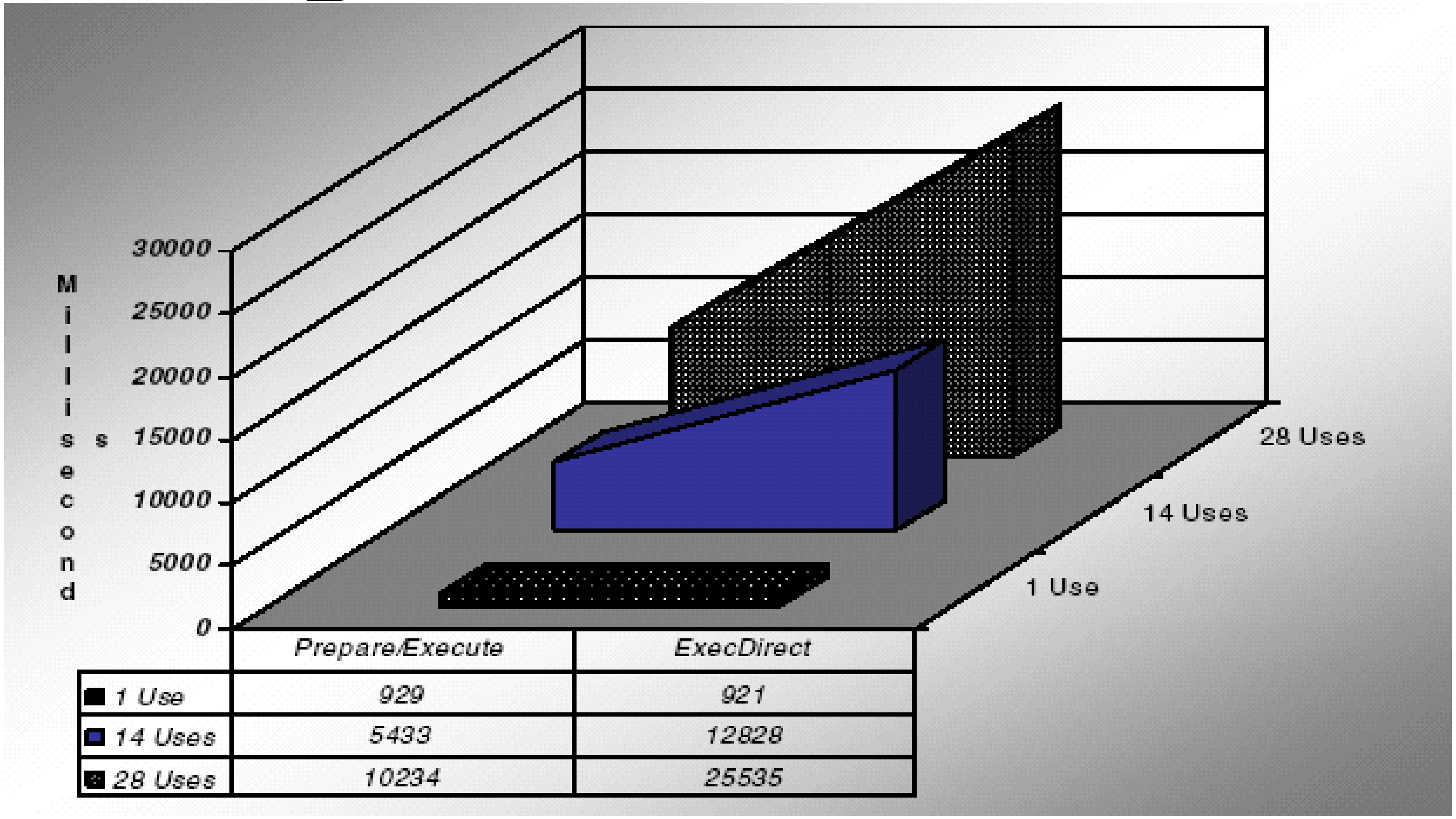


Application Optimizations

- **Structure the code so that the amount of information transmitted between the client and the server is minimized. Some examples:**
  1. Use joins instead of sub-queries (also consider IFX_FLAT_UCSQ)
  2. Perform aggregate functions in the query
  3. Prepare and reuse queries
  4. Reuse API objects rather than allocating new ones: statements and especially connections
  5. Consider using stored procedures
- **Reduce the number of client-server round trips**

Reusing Prepared Queries Example

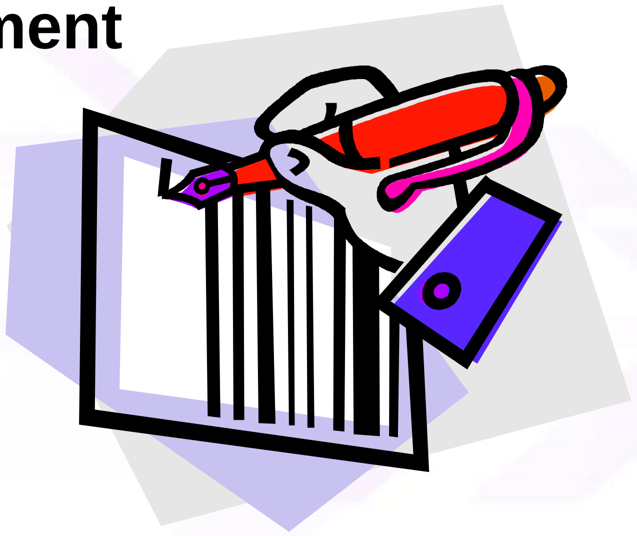
- **select fname, lname from customer where customer_num = ?**



	Prepare/Execute	ExecDirect
■ 1 Use	929	921
■ 14 Uses	5433	12828
▨ 28 Uses	10234	25535

Principle API Functions

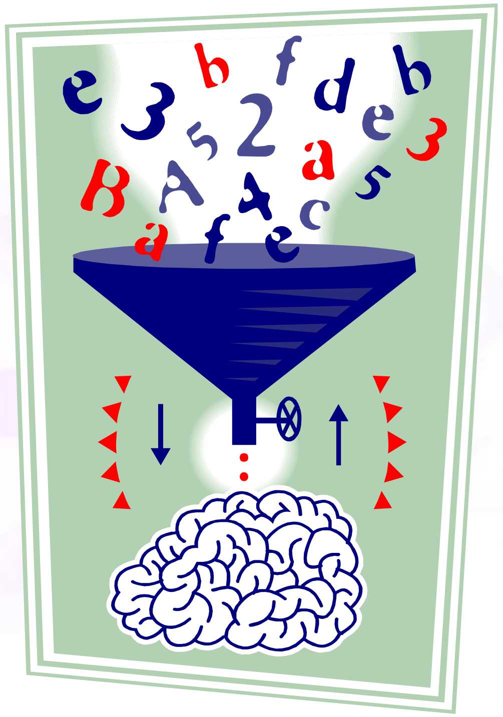
- **Internal statement management**
- **Internal connection management**
- **Code set conversion**
- **Data type conversion**
- **Data buffer management**
- **Message handling**



- All are candidates for optimization!

Data Type Conversion

- **The most commonly used APIs allow you to select data of one type into a variable of another type – avoid overusing this ability**
- **Keep database and application types as similar as possible - refer to API data type mapping tables**
- **Avoid using explicit casts where implicit casts will do**



Data Conversion Example

- **Required whenever data types differ between the application and the database**
- **Select customer_num from customer (100,000 rows)**
- **Variable binding - customer_num**
  1. To character - 6500 msec
  2. To integer - 5640 msec
- **Not something to worry about most of the time, but something to consider if developing a new application that processes a lot of data.**

Explicit Cast Example

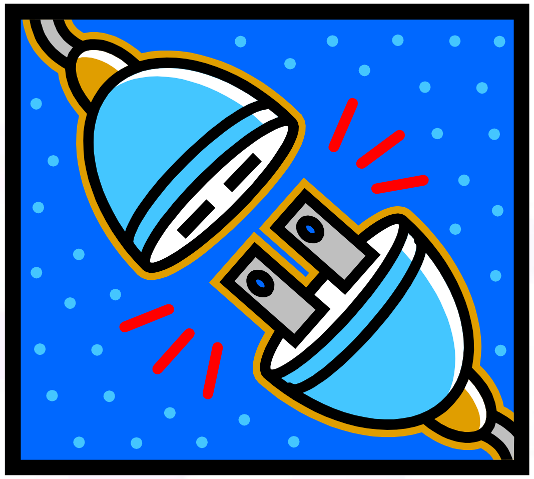
```
select  tab1.luid as col1, tab2.luid as col2
    from tab1
            inner join tab3 on tab3.luid = tab1.col3 and
               tab3.col6 <= 15134 and
               tab3.col4 > 15134
            left join tab4 on tab1.col5 = tab4.col5 and
               tab4.col6 <= 15134 and
               tab4.col4 > 15134
            left join tab5 on tab3.luid = tab5.col5 and
               tab4.luid is null and tab5.col6 <= 15134 and
               tab5.col4 > 15134
            inner join tab6 on tab6.luid = nvl(tab4.col3, tab5.col3) and
               tab6.duid = cast(tab3.luid as varchar(200)) and
               tab6.col6 <= 15134 and
               tab6.col4 > 15134
            left join tab2 on tab2.luid = tab6.col7 and
               tab2.col6 <= 15134 and
               tab2.col4 > 15134
    where (tab1.col6 <= 15134 and tab1.col4 > 15134) and
            (tab2.luid in (3470, 5248))
```

Explicit Cast Example

- UPDATE STATISTICS MEDIUM, explicit cast:
  - 55 minutes
- UPDATE STATISTICS HIGH, explicit cast:
  - 27 minutes
- UPDATE STATISTICS HIGH, implicit cast:
  - 3 seconds

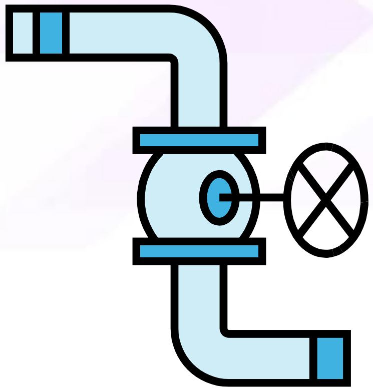
Connection Management

- **Connections are resources, internal structures are allocated and freed**
- **Closing and reopening is less expensive than freeing and reinitializing**
- **Some APIs can optimize around the connect-disconnect cycle (i.e. connection pooling)**

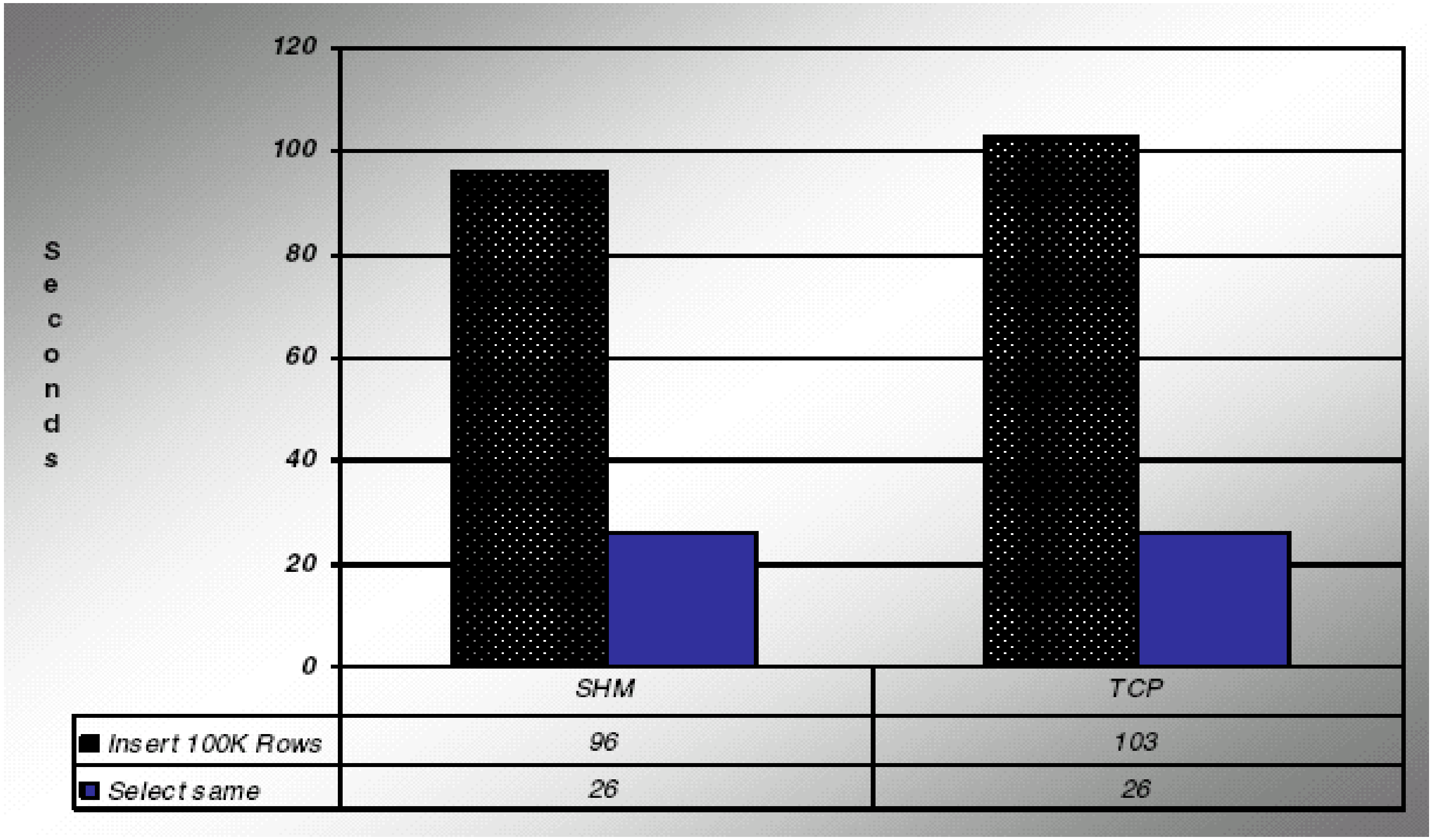


Communication Layer

- The “pipe” between the client and server processes
- OS-level connection establishment
- Message transfer
- Communication Options
  - Shared Memory
  - Pipe
  - TCP



Connection Type – Data Example

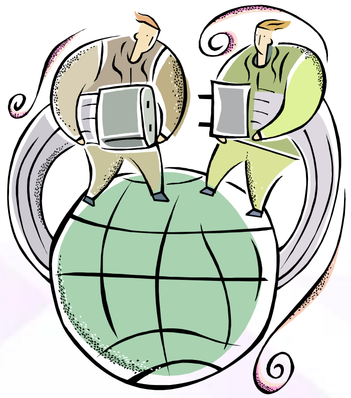


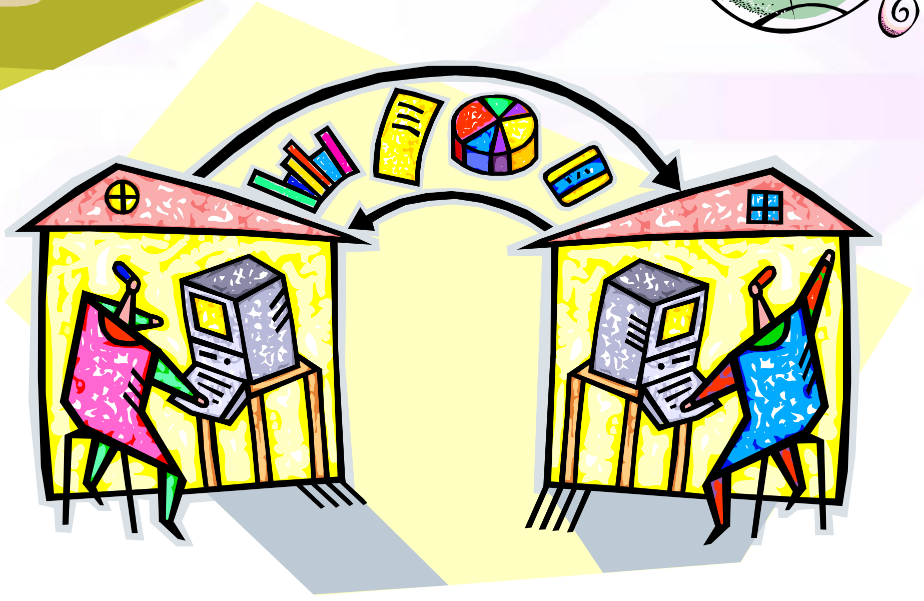
	SHM	TCP
■ Insert 100K Rows	96	103
■ Select same	26	26

Statement Handling

- **Statements are also resources; same rules apply**
- **Internal structures are allocated and freed**
- **Closing and reopening is less expensive than freeing and reinitializing**
- **Some APIs have some internal optimizations around the allocate-free cycle**

Network Messaging





Message Handling

- **SQLI messages carry information to and from the database server**
- **Informix products communicate using half duplex**
  - Client sends message to server
  - Server processes message
  - Server sends response
  - Client sends next message
- **Inherent delay in network and message processing adds to cost for each message**
- **Reduce the round trips by optimizing message transfers in your API**

Message Optimizations

- **Data Buffer Management**
  - Fetch Array Size
  - Fetch Buffer Size
- **Optimize Open-Fetch-Close (OPTOFC)**
- **Deferred Prepare**
- **Optimize Message Transfers (OPTMSG)**

Examining SQL Messages

- Network messages sent between the application and the database engine to accomplish SQL operations

```
PREPARE stmt FROM
          "select * from t where a > ?";
DECLARE cursor_1 from :stmnt;
OPEN cursor_1 USING var_1;
FETCH cursor_1 into results_1;
CLOSE cursor_1;
FREE cursor_1;
FREE stmt;
```

Normal Message Traffic

- 60KB of data returned by the select
- No blob data and rows smaller than 4KB

PREP/DEC	$\longleftrightarrow$	Small Sent - Medium Returned
OPEN	$\longleftrightarrow$	Small Sent - Medium Returned
FETCH	$\xleftrightarrow{16x}$	Small Sent - Medium Returned
CLOSE	$\longleftrightarrow$	Small Sent - Small Returned
FREE	$\longleftrightarrow$	Small Sent - Small Returned

Total number of messages sent = 40

FET_BUF_SIZE

- Set the buffer size for the fetch/insert buffers
- Size is specified in bytes up to 32KB
- Requires no application changes
- Yields similar performance as fetch array with NO application changes

FET_BUF_SIZE Message Traffic

- Example of the message traffic with the fetch buffer set at 16KB

PREP/DEC	⟷	Small Sent - Medium Returned
OPEN	⟷	Small Sent - Medium Returned
FETCH	4x ⟷	Small Sent - 16KB of data returned
CLOSE	⟷	Small Sent - Small Returned
FREE	⟷	Small Sent - Small Returned

Total number of messages sent = 16

OPTOFC Message Traffic

- Delays sending the *open* until the first fetch is requested by the application
- The engine will close the cursor when the last row is processed
- Requires only error checking changes
- Large performance improvements when selecting small sets of data

OPTOFC Message Traffic

- OPTOFC enabled
- Fetch buffer set at 16KB

PREP/DEC	⟷	Small Sent - Medium Returned
OPEN/FETCH	⟷	Medium Sent - 16KB Returned
FETCH	$\xleftrightarrow{2x}$	Small Sent - 16KB Returned
FETCH/CLOSE	⟷	Medium Sent - 16KB Returned
FREE	⟷	Small Sent - Small Returned

Total number of messages sent = 12

AutoFree Message Traffic

- Causes a cursor and its associated prepared statement to be freed upon closing the cursor

AutoFree Message Traffic

- Fetch buffer set at 16KB
- OPTOFC enabled

PREP/DEC	$\longleftrightarrow$	Small Sent - Medium Returned
OPEN/FETCH	$\longleftrightarrow$	Medium Sent - 16KB Returned
FETCH	$\xleftrightarrow{2x}$	Small Sent - 16KB Returned
FETCH/CLOSE/FREE	$\leftrightarrow$	Medium Sent - 16KB Returned

Total number of messages sent = 10

Result Summary

Options	Data Size	Network Messages
None	60 KB	40
FET_BUF_SIZE = 16,000	60KB	16

FET_BUF_SIZE=16000
OPTOFC=1
	60KB	12

FET_BUF_SIZE=16000
OPTOFC=1
IFX_AUTOFREE=1
	60KB	10

Small Select Statements Traffic Improvement

- Returning only one row

PREP/DEC	↔	Small Sent - Medium Returned
OPEN	↔	Small Sent - Medium Returned
FETCH	↔	Small Sent - Medium Returned
CLOSE	↔	Small Sent - Small Returned
FREE	↔	Small Sent - Small Returned

Total number of messages sent = 16

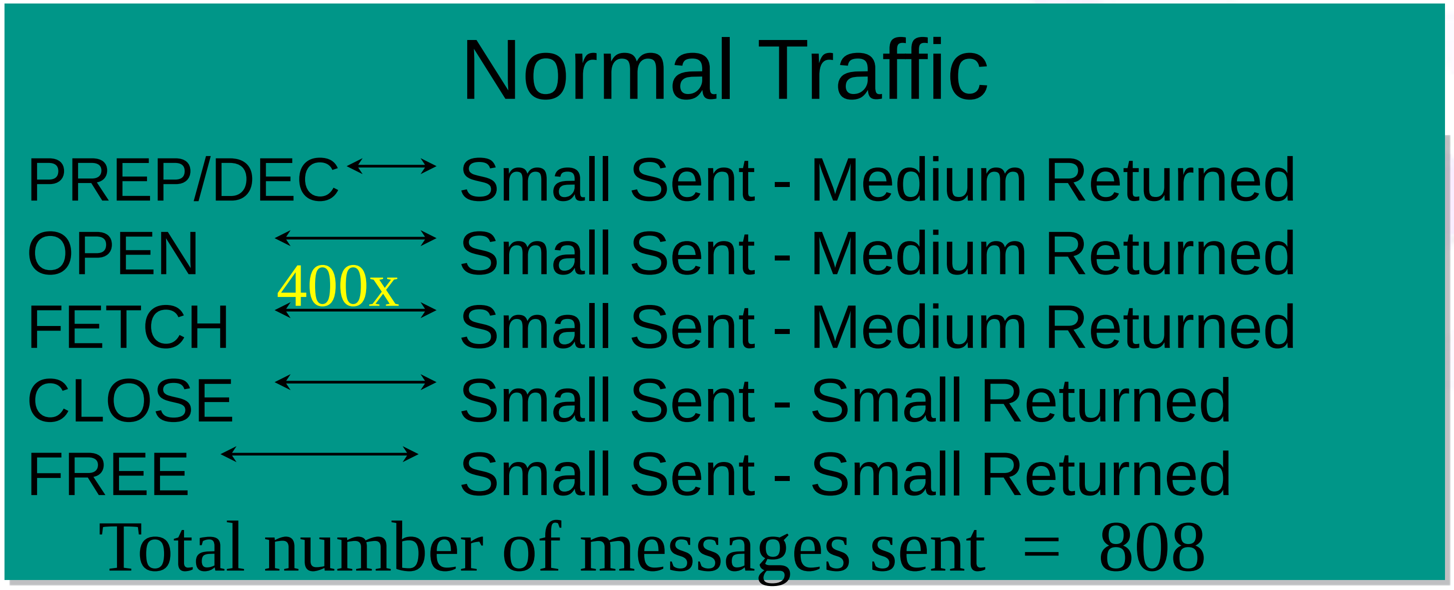
- OPTOFC & AutoFree enabled

PREP/DEC	↔	Small Sent - Medium Returned
OPEN/FETCH/CLOSE/FREE	↔	Medium Sent - 16KB Returned

Total number of messages sent = 2

Large Select Statements Traffic Improvement

- Select returning 1.6MB of data



Normal Traffic

PREP/DEC	$\longleftrightarrow$	Small Sent - Medium Returned
OPEN	$\longleftrightarrow$	Small Sent - Medium Returned
FETCH	$\xleftrightarrow{400x}$	Small Sent - Medium Returned
CLOSE	$\longleftrightarrow$	Small Sent - Small Returned
FREE	$\longleftrightarrow$	Small Sent - Small Returned

Total number of messages sent = 808

Large Select Statement Traffic Improvement

- Fetch buffer set at 32KB
- OPTOFC enabled
- AutoFree enabled

PREP/DEC	↔	Small Sent - Medium Returned
OPEN/FETCH	↔	Medium Sent - 32KB Returned
FETCH	↔ 48x	Small Sent - 32KB Returned
FETCH/CLOSE/FREE	↔	Medium Sent - 32KB Returned

Total number of messages sent = 102

Optimizing Data Sorting

- How can the DBA find out information about Sorting
- Examine ways to make sorts run faster

Information About Sorting

```
dbaccess sysmaster -
select * from sysprofile
    where name matches "*sort*";

 name                       value

 totalsorts                  2842
 memsorts                    2401
 disksorts                    441
 maxsortspace                1234
```

Number of times the database server has been asked to sort data

Number of times the database server has sort data using disk and memory

Number of times the database server has sort data solely in memory

The size in base pages of the larges sort

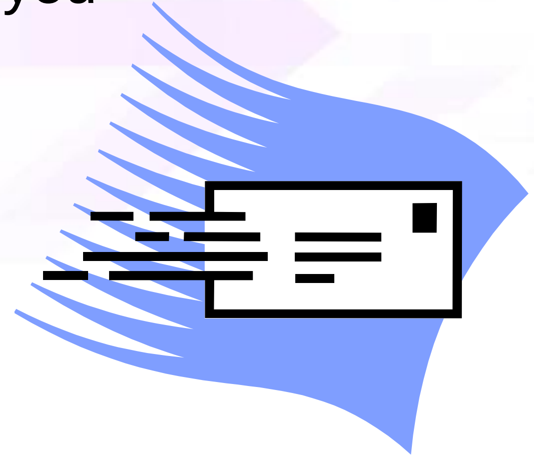
Sort Space and Type by User

```
dbaccess sysmaster -
select total_sorts, dsksorts,
max_sortdskspace from syssesprof;

sid   total_sorts    dsksorts    max_sortdiskspace


11              0           0                    0
13             23           0                    0
14           3819           0                    0
```

Indicates a specific user/application a DBA can examine for efficiency

Ways of Optimizing Sorts

- Avoiding Sorts
  - Improve indexes on tables
  - Sorts consume less resources then reading the index when the number of index keys to read compared to the index size is large
- Narrow columns require less work to sort
- Reduce the data to sort, only select the rows you need
  - SELECT FIRST N
  - SELECT FIRST N SKIP M



OLTP or Small Sort Task

- Tune DS_NONPDQ_QUERY_MEM
  - The amount of memory given to sorts who have a PDQ of 0
  - Minimum Value 128KB
  - Maximum Value 25% of DS_TOTAL MEMORY
- Modify the parameter while online
  - onmode -wf DS_NONPDQ_QUERY_MEM=500
- Example
  - A query with a sort would previously be given 128KB of memory to sort data.
  - If the sort exceeded 128KB of data to sort, it would go to disk
  - Now the sort only goes to disk if the data sorted exceeds DS_NONPDQ_QUERY_MEM

Customer Comment about Implementing DS_NONPDQ_QUERY_MEM

…CPU load average on the server dropped from 90% during peak to less than 50%. I also tracked the actual CPU seconds that oninit was using before and after the change. During peak times the oninit's for this instance was using 5 out of the 8 physical cpu's. After the change we consume less than 3 cpu's.

Large Sort Tasks

- PDQ
  - Enable PDQPRIORITY to allow more threads and resouces
  - Ensure DS_TOTAL_MEMORY in the onconfig file is set properly

If large sorting batch jobs run at night consider dynamically increasing this at the start of the evening and resetting it to its original value in the early morning with **onmode -M**

  - Make sure PSORT_NPROCES is set to the number of CPU VPs
  - DS_MAX_QUERIES is set

Temp Space for Sorts

Where is the best location for sort files when considering performance?

How to you control where sort files are placed?

Times for Index Builds

- Scenario
  - 200,00 rows
  - 1 CPU VP
  - Building an index on integer, char(20)

Using a temp dbspace
?? seconds

Using file system as temp space
?? seconds

First Rows Optimization

- Very fast to retrieve the first few rows
  - Avoids sorting at all cost
- To process the entire query can takes more resources and time
- Set in either: ONCONFIG parameter, Application Statement, Optimizer directive

	FIRST_ROWS	ALL_ROWS
Time for first row returned	1 second	12 seconds
Time for last row returned	62 seconds	15 seconds

First Rows Optimization

- First rows can be optimized by not requiring the entire result set be returned
  - SELECT FIRST N
    - Used to limit the number of rows returned from the query

```
SELECT FIRST 10 {+first_rows}  * FROM employees
```

  - SKIP & FIRST
    - Return the 7 to 17 row

```
SELECT SKIP 7 FIRST 10 {+first_rows} * FROM employees
```

OnLine’s I/O Subsystems

- Two different AIO Subsystems
- Kernel AIO
- Informix AIO

AIO Systems

- Kernel AIO (KAIO)
  - Utilizes kernel asynchronous I/O by running KAIO threads on each CPU VP
  - Informix uses it for only Raw IO
- Informix AIO VP's
  - Used for all cooked I/O
  - The I/O system used when Informix does not use the Kernel supplied KAIO package

Adjusting AIO VPs with KAIO

- AIO is used for file system I/O
  - file system sorts
  - stored procedures trace files
  - password lookups
- Small number of AIO VPs required
- **Common Monitoring Mistake**
  - **Do not use the process waitio time to measure disk activity**
  - **KAIO processes generally do not wait on disk I/O**

Adjusting AIO VPs without KAIO

- General rule “Check the inverted pyramid”
- Increase the number when
  - All the AIO VPs are in a sleeping state
  - All the AIO VPs consistently busy
- Decrease the number when total number of operations is 0 for one or more AIO VPs

Performance Tip

When initially setting AIO VPs error on the side of setting them too high

Watching The Inverted Pyramid
onstat –g iov

```
AIO I/O vps:
class/vp s   io/s totalops  dskread dskwrite  dskcopy  wakeups  io/wup  errors
  msc   0 i   0.0       26        0        0        0       26   1.0        0
  aio   0 s   0.2     1080      883      188        0      849   1.3        0
  aio   1 i   0.2     1072      874      189        0      830   1.3        0
  aio   2 s [illegible] 884     852       28        0      699   1.3        0
  aio   3 i [illegible] 647     831       13        0      769 [illegible] 0
  aio   4 i [illegible] [illegible] 832   16        0      712 [illegible] 0
  aio   5 i   0.2 [illegible]   811       12        0  [illegible] [illegible] 0
  pio   0 i   0.0 [illegible]     0        2        0 [illegible]  [illegible].7 0
  lio   0 i   0.0        3 [illegible]     3        0 [illegible]   0.8        0
```

S	indicates in System call (VP is currently busy)
I	indicates the VP is currently idle

Monitoring I/O Resources by Disk/Chunk *onstat -g iof*

```
AIO global files:
gfd pathname            totalops  dskread dskwrite  io/s
  3 /dev/tsinfo1            1506     1393      113   0.0
  4 /dev/tsinfo2             776        3      773   0.0
  5 /dev/tsinfo0           22205     7994    14211   0.4
  6 /dev/tsinfo3           55092    54102      990   1.1
```

dskread/dskwrite

Number of read/write system calls

io/s

Average I/O operations per second

onstat -p

```
dskreads pagreads bufreads %cached dskwrits pagwrits bufwrits %cached
1083      1653      193884  99.44   3860     14583    59454    93.51

isamtot   open     start     read     write    rewi
139364   21933    22499    36438   16624     17

gp_read  gp_write gp_rewrt gp_del   gp_alloc gp
41         15         147       0        4          0

ovlock   ovuserthread ovbuff   usercpu  syscpu
0          0                0         24.07     2.87     7            17

bufwaits lokwaits lockreqs deadlks  dltouts  ckpwaits compress seqscans
0           1          54737    0           0          4           1623        690

ixda-RA  idx-RA   da-RA    RA-pgsused lchwaits
0             0          0          0                349
```

Percentage of reads from shared memory relative to disk reads. OLTP system should be above 95 %. Indicator of too few buffers in system.

onstat -p

```
dskreads pagreads bufreads %cached dskwrits pagwrits bufwrits %cached
1083      1653      193884   99.44    3860     14583     59454     93.51

isamtot  open      start      read
139364  21933    22499    36438

gp_read  gp_write gp_rewrt gp_de
41            15            147          0

ovlock    ovuserthread ovbuff   usercpu  syscpu   numckpts flushes
0            0                    0          24.07      2.87      7                17

bufwaits lokwaits lockreqs deadlks  dltouts  ckpwaits compress seqscans
0            1           54737         0            0               4           1623        690

ixda-RA  idx-RA   da-RA    RA-pgsused lchwaits
0             0          0          0                 349
```

If ratio between seqscans and isamtot is greater than 1%, then we might want to check index usage.

onstat -p

```
dskre
1083

isamt
1393

gp_read  gp_write gp_rewrt gp_del   gp_alloc gp_free  gp_curs
41          15           147        0         4            0          4

ovlock   ovuserthread ovbuff   usercpu  syscpu
0           0                   0          24.07      2.87      7

bufwaits lokwaits  lockreqs deadlks  dltouts  ckp
0           1           54737    0             0          4

ixda-RA  idx-RA   da-RA    RA-pgsused lchwaits
0            0          0          0                349
```

Number of times we had to wait on a buffer in the buffer pool. This can indicate that a single page is being altered too much. Also this can occur if the page is being flushed to disk too often (LRU_MIN set to 0)

Number of times lock request issued on a locked table/page/row. If ratio between lokwaits/lockreqs is too high, then applications may be single-threading.

onstat -p

```
dskreads pagreads bufreads %cached dskwrits pagwrits bufwrits %cached
1083      1653      193884   99.44    3860     14583    59454    93.51

isamtot   open      start     read      write     rewrite  delete  commit  rollbk
139364   21933     22499     36438     16624      1747      1445    1160     0

gp_read  gp_write gp_rewrt gp_del   gp_al
41          15          147        0        4

ovlock   ovuserthread ovbuff   usercpu  sys
0          0                 0         24.07      2.87

bufwaits lokwaits lockreqs deadlks  dltouts
0           1          54737    0           0          4        [illegible]       [illegible]

ixda-RA  idx-RA   da-RA    RA-pgsused lchwaits
0            0          0          0                349
```

Ratio between rollback and commits. If too high (> 1%) then application is probably not designed correctly. Need to examine why so many rollbacks and take corrective action.

onstat -D

```
Dbspaces
address  number   flags      fchunk   nchunks  flags
ad067d8  1        0x20001    1        1        N
b4ad870  2        0x28001    2        1        N S
b4d9720  3        0x28001    3        1        N S      informix sbindex
b4ec880  4        0x28001    4        1        N S      informix
b4ed9e0  5        0x28001    5        1        N S      informix
 5 active, 2047 maximum

Chunks
address  chunk/dbs  offset    page Rd  page Wr  pathname
ad06928    1     1    0       1621     15163   /work/mpruet/db/Chunk1
b4ad9c0    2     2    0       254      18      /work/mpruet/db/Chunk2
b4d9870    3     3    0       73       32      /work/mpruet/db/Chunk3
b4ec9d0    4     4    0       352      342     /work/mpruet/db/Chunk4
b4edb30    5     5    188000  243      233     /work/mpruet/db/Chunk1
 5 active, 2047 maximum

Expanded chunk capacity mode: disabled
```

Examine chunks with high activity. Consider splitting data on those chunks.

If the total number of IO for a single device exceeds 8/sec*, consider moving some data off.

Avoid having two chunks on the same device

onstat –g iof

```
AIO global files:
gfd pathname        totalops  dskread dskwrite  io/s
  3 RootChunk          11771     8723     3049  3.27
  4 Chunk1             25341    25341     4611  8.32
  5 Chunk2             12420     5436     6984  3.45
  6 SBChunk             1260     1120      140  0.35
  7 Chunk3               108      107        1  0.03
```

Need to consider moving data from this chunk. It is starting to get too much activity. If it is only one table, then consider fragmentation of that table.

Generic File Descriptor – an internal number that is used to identify the chunk across all virtual processors.

* 8 I/O's per second might not be a realistic trigger if the I/O complex can actually handle 24 I/O's per second with no problem. Need to know your system's capabilities

onstat -F

```
Fg Writes     LRU Writes     Chunk Writes
0              24             2829

address  flusher  state    data
add461c  0        I        0        = 0X0
      states: Exit Idle Chunk Lru
```

Writes done as part of a checkpoint

Writes done between checkpoints.

Foreground Writes – page flush requested by user thread - If this is ever not zero then buffer pool is too dirty. LRU_MIN/LRU_MAX, BUFFERS and/or CKPTINTVL need adjusting.

onstat –g ppf or sysmater:sysptprof

Partition profiles

partnum	lkrqs	lkwts	dlks	touts	isrd	iswrt	isrwt	isdel	bfrd	bfwrt	seqsc	rhitratio
0x100123	8698	0	0	0	3163	243	242	62	12209	1219	3	100
0x100124	4660	322	31	53	6553	3797	148	964	41278	11832	0	100
0x100125	1366	0	0	0	652	210	148	62	3850	1138	1	100
0x100126	1015	0	0	0	65	306	0	111	5330	1410	1	100
0x100127	771	0	0	0	251	110	0	0	1390	380	0	100
0x100128	506	0	0	0	72	357	0	0	1207	778	0	100
0x100129	534	0	0	0	940		0	0	1750	12	359	67

Partition number of table/fragment

Can be used to determine which tables/fragments tend to have the most activity.

Buffer Read Hit Ratio – Ratio of buffer reads that did not require a physical disk read

onstat –g ppf or sysmater:sysptprof

Partition profiles

partnum	lkrqs	lkwts	dlks	touts	isrd	iswrt	isrwt	isdel	bfrd	bfwrt	seqsc	rhitratio
0x100123	8698	0	0	0	3163	243	242	62	12209	1219	3	100
0x100124	4660	322	31	53	6553	3797	148	964	41278	11832	0	100
0x100125	1366	0	0	0	652	210	148	62	3850	1138	1	100
0x100126	1015	0	0	0	65	306	0	111	5330	1410	1	100
0x100127	771	0	0	0	251	110	0	0	1390	380	0	100
0x100128	506	0	0	0	72	357	0	0	1207	778	0	100
0x100129	534	0	0	0	940		0	0	1750	12	359	67

Number of times that a lock timed out

Number of deadlocks encountered

Times thread was put on wait for a lock

Lock Requests

onstat –g ppf or sysmater:sysptprof

Partition profiles

partnum	lkrqs	lkwts	dlks	touts	isrd	iswrt	isrwt	isdel	bfrd	bfwrt	seqsc	rhitratio
0x100123	8698	0	0	0	3163	243	242	62	12209	1219	3	100
0x100124	4660	322	31	53	6553	3797	148	964	41278	11832	0	100
0x100125	1366	0	0	0	652	210	148	62	3850	1138	1	100
0x100126	1015	0	0	0	65	306	0	111	5330	1410	1	100
0x100127	771	0	0	0	251	110	0	0	1390	380	0	100
0x100128	506	0	0	0	72	357	0	0	1207	778	0	100
0x100129	534	0	0	0	940		0	0	1750	12	359	67

In general numbers here indicate that the application may need some work.
Check for ‘hot rows’,.

In general numbers here indicate that we might want to check to see if adding an index would be in order

Sysmaster syssqexplain table

- Contains information about statements currently open and prepared statements
- Can be used to find queries that are requiring sorts, temporary tables, sequential scans, etc
- Includes actual query statement
- Can aid in finding problem queries

The first part of the output
is actual costs that the statement
has already encountered.

```
Select * from syssqexplain;


sqx_sessionid    23
sqx_sdbno         0
sqx_iscurrent     Y
sqx_executions    4
sqx_cumtime       0.21
sqx_bufreads      21
sqx_pagereads     4
sqx_bufwrites     0
sqx_pagewrites    0
sqx_totsorts      4
sqx_dsksorts      0
sqx_sortspmax     0
sqx_conbno        0
sqx_ismain        Y
```

Number of times the statement has been executed

Total time execution of statement has encountered

Statistics about statement.

Max disk space needed for sort

Select * from syssqexplain;

Pay close attention to queries that require an auto index, temp files, and sequential scans.

sqx_selflag	SQ_SELECT	
sqx_estcost	654	Optimizer Cost
sqx_estrows	200	
sqx_seqscan	2	Number Sequential Scans
sqx_srtscan	0	Number Sort Scans
sqx_autoindex	0	Number of Auto Indexes
sqx_index	0	
sqx_remsql	0	Number of Remote Queries Required
sqx_mrgjoin	0	
sqx_dynhashjoin	0	
sqx_keyonly	0	
sqx_tempfile	1	Number of Temp Files
sqx_tempview	0	Number of Temp Views
sqx_softheads	0	
sqx_sqlstatement	select * from account, transactions where account.acct_id = transactions.account_id order by tran_amount	Actual Query

Using IDS 10 features to Improve Performance

- Non PDQ Memory
- Fast Restart
- External directives
- Multi Fragments in a Single Dbspace
- Configurable Page Size
- General Speed Improvements
  - Code Path Reduction
  - Scalability

External Directives

- Allows DBA to modify the query plan w/o changing the application
- Directives not required to be in the application
- Different mode of directives: INACTIVE, ACTIVE, TEST ONLY
- To create external directives use the `SAVE EXTERNAL DIRECTIVE` statement

```
SAVE EXTERNAL DIRECTIVES
/*+ AVOID_INDEX (table1 index1)*/
ACTIVE FOR SELECT col1, col2
FROM table1, table2
WHERE table1.col1 = table2.col1
```

Fast Restart

- The physical log uses large I/O operations to seed the buffer pool with the most recently modified pages
- Allows for fast recovery to work mainly with pages which are pre-loaded into the buffer pool
- The buffer pool is seeded with commonly modified data for the users
- Helps to prime the database buffer cache

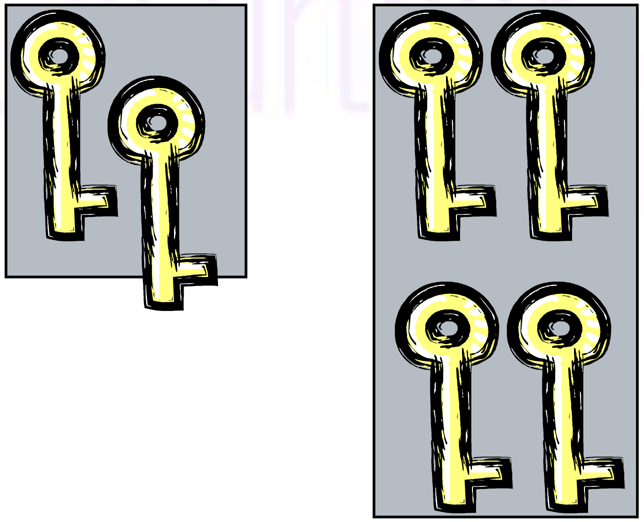
Setting the Default lock mode for Table

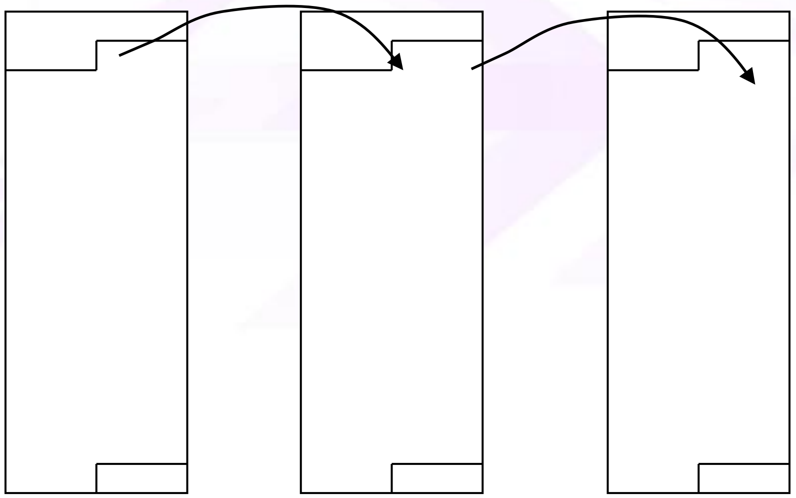
- DEF_TABLE_LOCKMODE
  - Onconfig parameter
  - Values PAGE | ROW
  - Default PAGE
- IFX_DEF_TABLE_LOCKMODE
  - Environment variable parameter
  - Values PAGE | ROW

If DEF_TABLE_LOCKMODE is set to ROW it sets the lock mode to row for every newly created table. This parameter has no effect on the lock mode for existing tables.

Configurable Page Size

- Reduces the number of rows that span pages (i.e long rows)
- Allows for large index key sizes
- Allows for more index keys per page, reducing depth of index





Onconfig Parameters

- OPTCOMPIND
  - 0 – give preference to nested loop join (OLTP)
- DIRECTIVES
  - 1 – turn on optimizer directives
- OPT_GOAL
  - Tells the optimizer what is your definition of fast
    - 0 first row returned
    - -1 time to return all rows
- EXT_DIRECTIVES
  - Enable external directives

Onconfig Parameters

- NUMCPUVPS – The endless debate.
  - Given enough physical processors performance will peak out between 16 to 32 CPU VP's but you need to not use all the physical processors for CPU VP's.
  - Hyperthreading
  - On a small box; <= 4 CPU's, you might use all 4 CPU's for CPU VP's.
  - On a larger box; > 4 CPU's, you might use 5 CPU VP's per 6 CPU's.
  - However, there are many factors which can affect this.
    - Are there many client processes running on the same box?
    - Are you using NET VP's for the poll threads handling many connections?
    - Have you disabled KAIO and configured many AIO VP's.
  - The only real rule is to add CPU VPs until it no longer provides any improvement.

Onconfig Parameters

- RESIDENT
  - Set to 1
  - If you have large amounts of memory use **-1**
    - This makes a big performance difference on big Sun boxes.
- NOAGE
  - Set to 1
- Processor Affinity (AFF_SPROC/AFF_NPROCS)
  - Turn on if engine is only thing running on the machine

Onconfig Parameters

- CLEANERS
  - < 20 disks – set to 1 per disk
  - 20 - 100 disks – set to 1 for every other disk
  - > 100 disks - set to 1 for every four disks
  - Striping and RAID-5
    - Cut number of cleaners down by 1/3

Onconfig Parameters

- LRU
  - Set to max of 128. 512 on 64-bit platforms
- LRU_MIN/LRU_MAX
  - Avoid setting LRU_MIN to 0 as it can cause additional buffer waits
  - 1 and 2 in most large memory OLTP systems – higher values for DSS systems. In 9.4 you can set these to float values like ".2". This is useful when 1% of a huge buffer pool is large causing slow checkpoints.

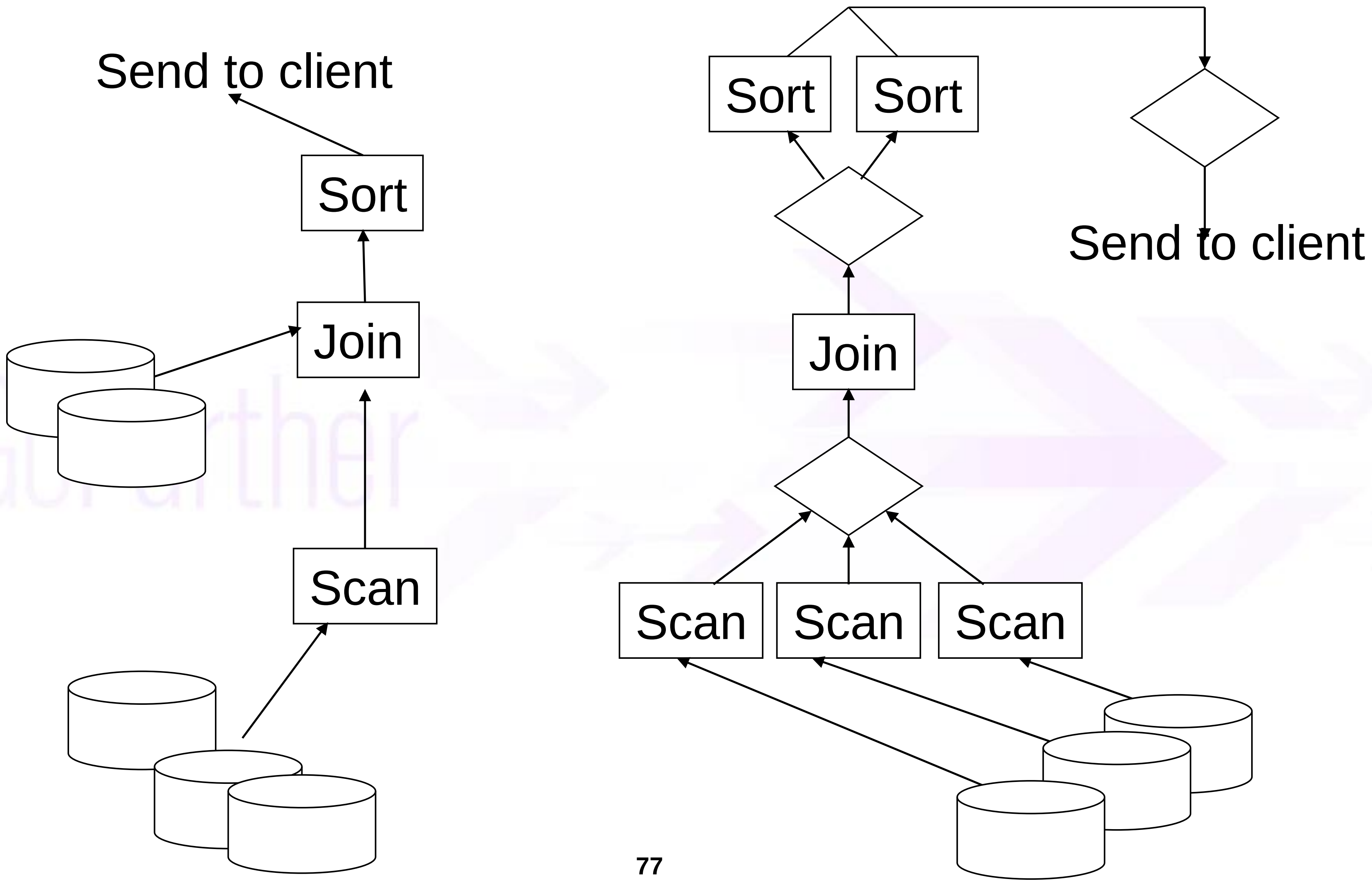
Onconfig Parameters

- Read Ahead
- RA_PAGES
  - Number of pages to read ahead when scanning
  - 32-64
  - Too large a value can cause additional buff waits and/or block the IO channel.
- RA_THRESHOLD
  - Point where next big buffer read is triggered

PDQ/Fragmentation

- Consider fragmenting any large table in a dbspace that is getting a lot of IO activity
- Consider fragmenting any large table if scans must be done against the table
- Do not put multiple fragments of the same table on the same physical device
- Avoid using round robin fragmentation for indexes.
- Do not over-fragment.
  - The cost of managing fragmentation can outweigh the benefits when there are excessive fragments.

Typical Query with Non-PDQ vs PDQ



PDQ Configuration

- MAX_PDQPRIORITY
  - Set highest percentage of PDQ resources that a single client can use
- DS_MAX_QUERIES
  - Max number of DSS queries that can be run together
- DS_TOTAL_MEMORY
  - Total memory reserved for PDQ
- DS_MAX_SCANS
  - Max number of parallel scans allowed. Leave at default (1048567)

PDQ Configuration

- If the site is primary a DSS system, then it is recommended that most of the allocated memory be in the virtual buffers and that DS_TOTAL_MEMORY be very large
- PDQ can be used in smaller memory environments by setting PDQ_PRIORITY to 1 so that only parallel scans can be done.

PDQ Online Configuration

- onmode can be used to dynamically change PDQ parameters
  - onmode –M (DS_TOTAL_MEMORY)
  - onmode –Q (DS_MAX_QUERIES)
  - onmode –D (MAX_PDQPRIORITY)
  - onmode –S (DS_MAX_SCANS)
- It is not recommended to try to do DSS queries at the same time that the system is doing OLTP

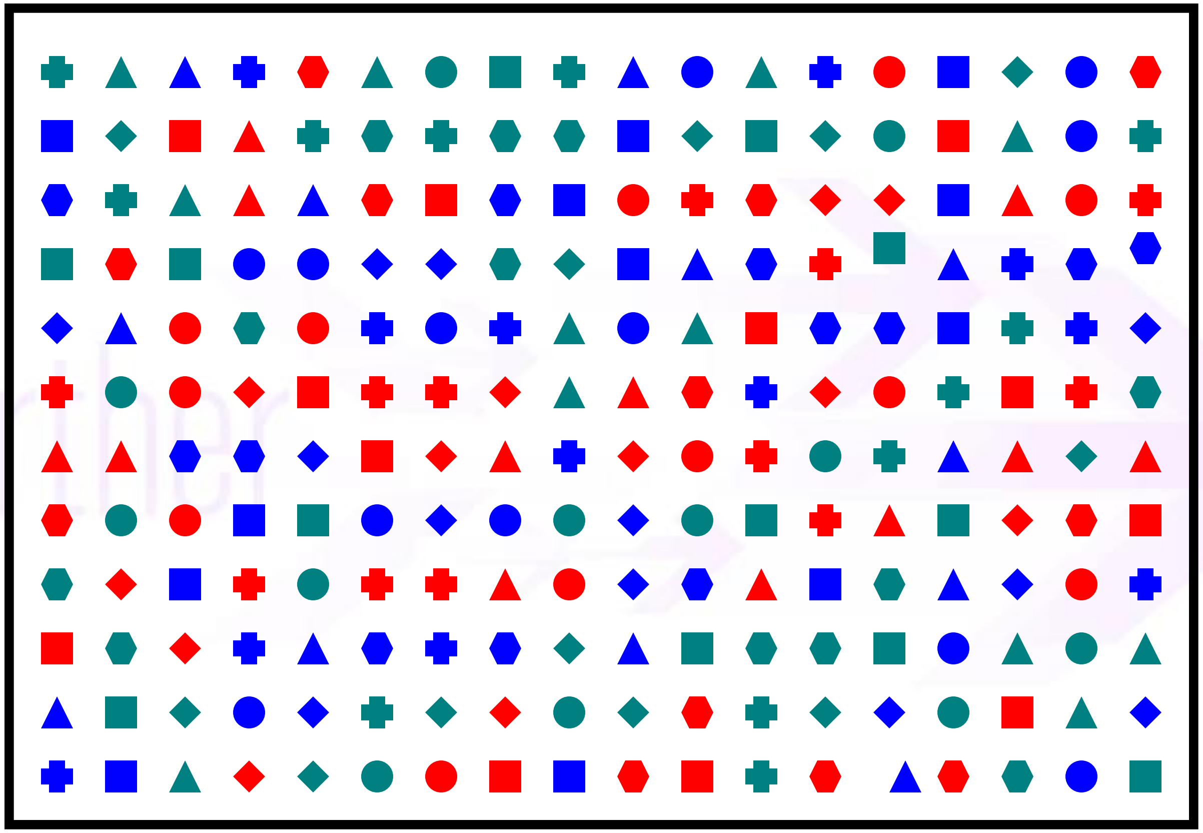
Fragmentation Overview

- Introduction to Fragmentation
- Defining Fragmented Tables
- Altering/Attach/Detach
- Monitoring/Guidelines

What is Fragmentation?

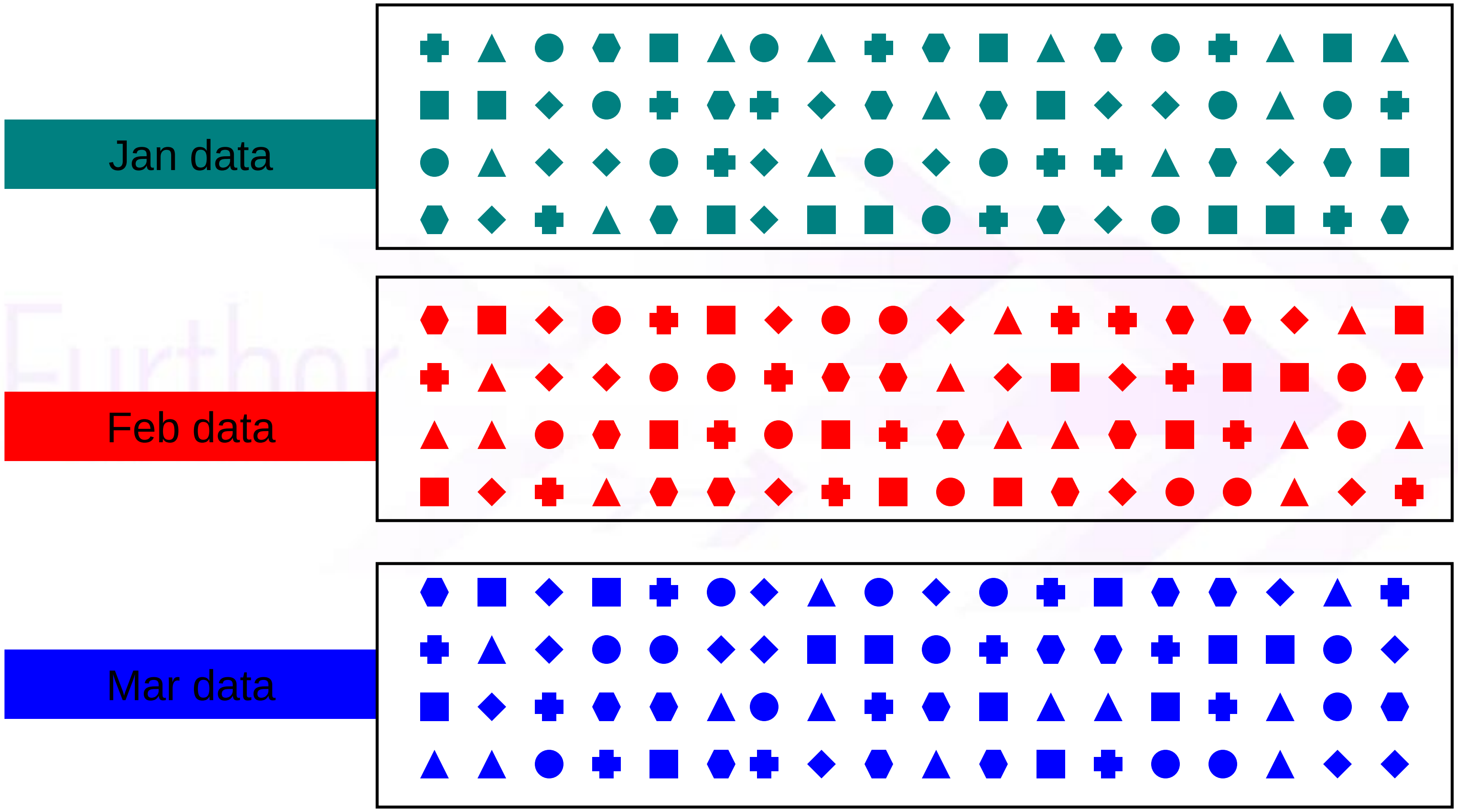
- Also called “Table Partitioning”.
- Fragmentation is a database server feature that allows you to
  - Control where data is stored at the **table** level
  - Distribute data from one table across separate dbspaces and partitions in dbspaces
  - **Creating multiple partitions in the same dbspace is a feature introduced in 10.00 .**
- Transparent to end users and client applications.
  - No changes for users and applications whether the table is fragmented or not fragmented.

Data in a non-fragmented table.

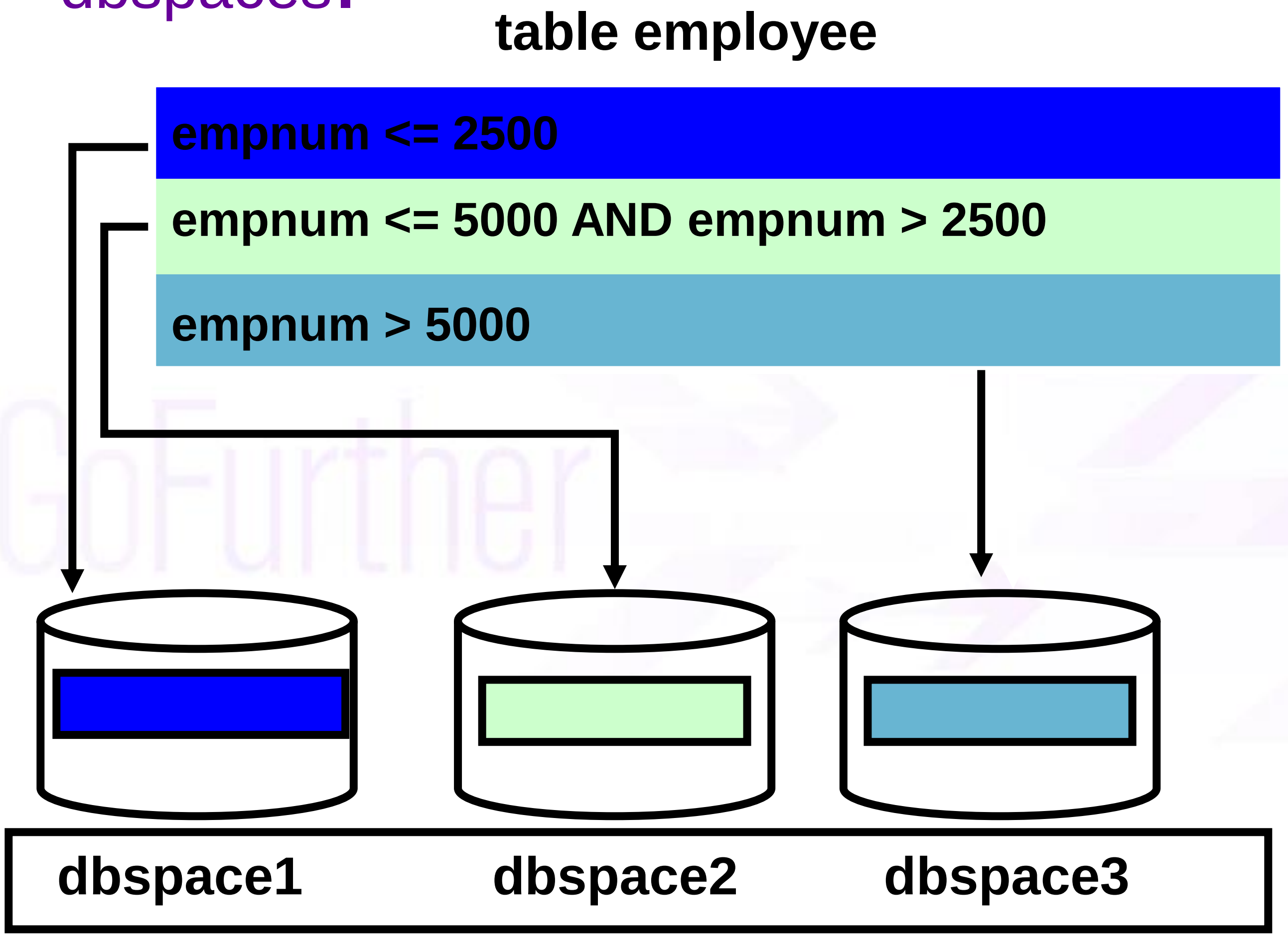


Fragment by expression

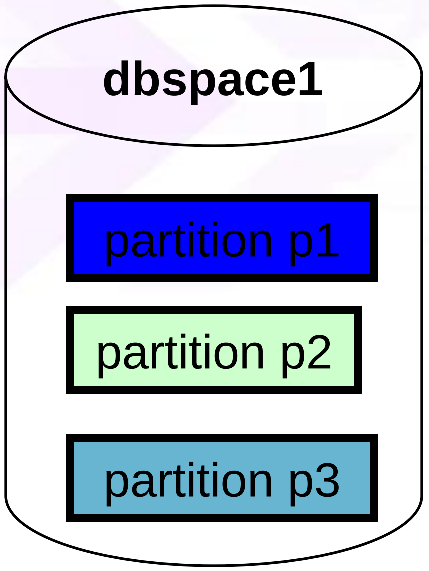
Data in 3 table fragments/partitions/data ranges.



Fragmentation is distribution of data into separate partitions. The partitions can be in the same dbspace or different dbspaces.



OR, each fragment/partition can be in the same dbspace.

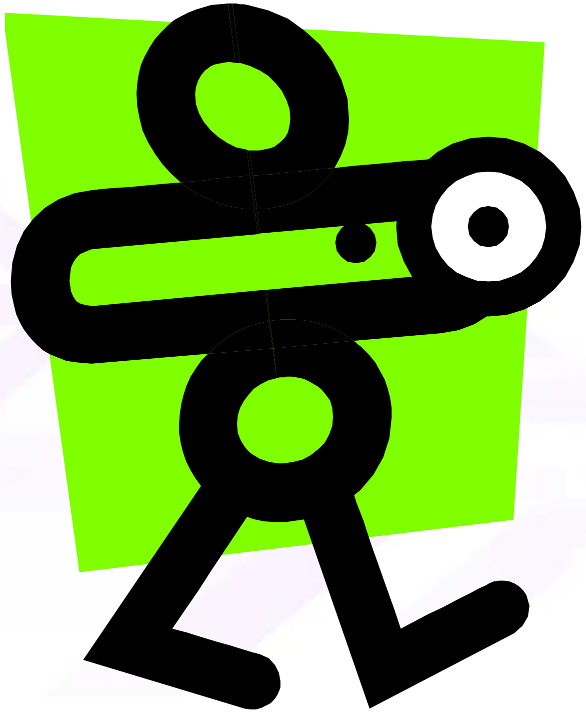


Create a Fragmented Table

- **This syntax (fragments/partitions in different dbspaces) works on all IDS versions.**

```
CREATE TABLE employee (
        empnum SERIAL,
        ... )
FRAGMENT BY EXPRESSION
                empnum <= 2500
                        IN dbspace1,
                empnum <= 5000 AND empnum > 2500
                        IN dbspace2,
                empnum > 5000
                        IN dbspace3
EXTENT SIZE 100 NEXT SIZE 60
LOCK MODE ROW;
```

Greater Fragmentation Less Administration



- Multiple Fragments in a single DBSpace
  - Can use a large degree of fragmentation on both index and data fragments without needed more dbspaces
  - Useful for fragment elimination
  - Not as useful for parallel scanning

Create a Fragmented Table

- **This syntax (fragments/partitions in same dbspace) is available from IDS 10.00.**

```
CREATE TABLE employee (
        empnum SERIAL,
        ... )
FRAGMENT BY EXPRESSION
        partition p1 (empnum <= 2500)
                IN dbspace1,
        partition p2 (empnum <= 5000 AND empnum > 2500)
                IN dbspace1,
             (empnum > 5000)
                IN dbspace2 ;
```

Partition name defaults to dbspace2 since it was not specified. However, partition name should be explicitly stated to avoid maintenance complications.

Why Fragment?

- **Fragment Elimination**
  - **Improve query performance.**
  - **A large table can perform like a smaller one.**
- Larger Tables -- increase table capacity.
  - Internally, each table fragment is treated like a table. So, some limits/properties that apply to a table (like maximum number of extents) apply to each fragment/partition.
- Balanced I/O
- If using PDQ, parallel scans, inserts, joins, sorts, aggregates, groups. (DSS systems)

Fragment Elimination (Set Explain Output)

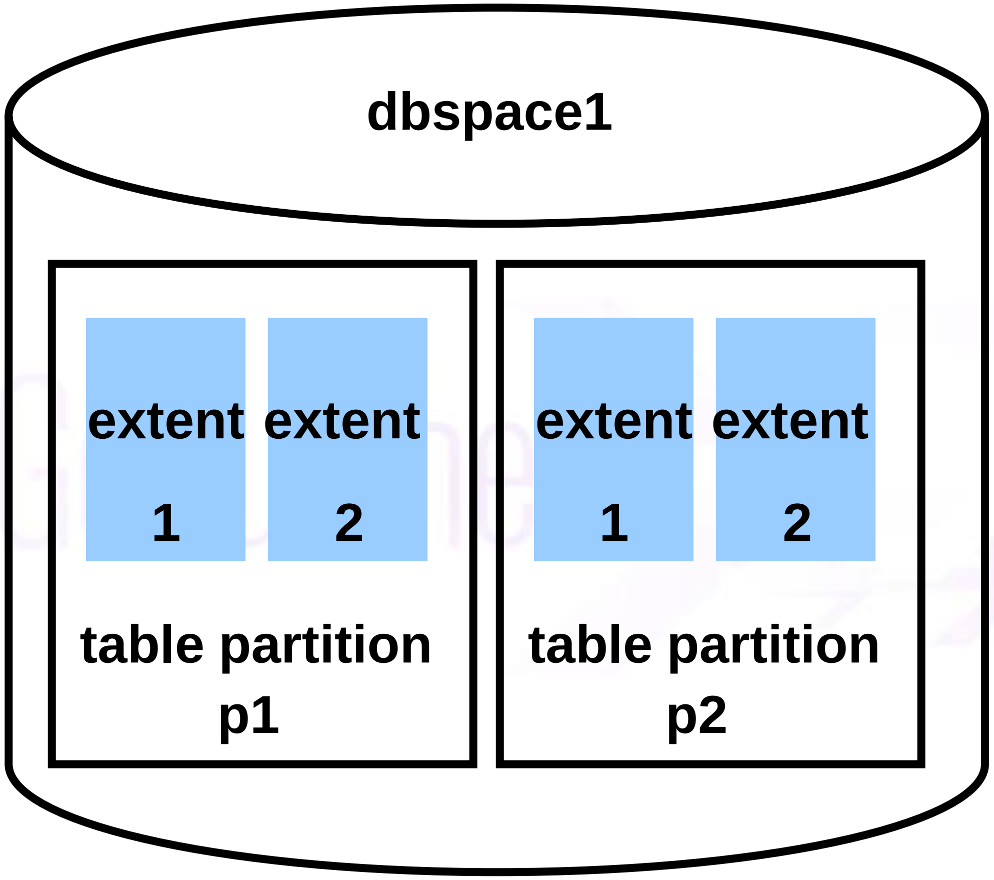
QUERY:
select * from tab1 where empnum < 1000
  1) informix.tab1: SEQUENTIAL SCAN (Serial, **fragments: 0**) **← 0 is the first fragment**
Filters: informix.tab1.empnum < 1000

QUERY:
select * from tab1 where empnum > 2700 and empnum < 4000 AND informix.tab1.empnum > 2700)
  1) informix.tab1: SEQUENTIAL SCAN (Serial, **fragments: 1**)
 Filters: (informix.tab1.empnum < 4000 AND informix.tab1.empnum > 2700)

QUERY:
select * from tab1 where empnum > 6000
  1) informix.tab1: SEQUENTIAL SCAN (Serial, **fragments: 2**)
        Filters: informix.tab1.empnum > 6000

QUERY:
select * from tab1 where empnum > 4000
  1) informix.tab1: SEQUENTIAL SCAN (Serial, **fragments: 1, 2**) **← 2 fragments scanned**
        Filters: informix.tab1.empnum > 4000

Fragments and Extents



- A table fragment is also called a “tablespace” or a “partition”.
- A table fragment has a fragment id, also known as tblspace id, or a partnum.
- When creating fragmented tables, the given extent size is specified for each fragment, not the entire table.

“Fragment BY” Types

- Expression based
  - Allows fragment elimination.
- Round robin
  - Creates even data distribution by randomly placing rows in fragments.

```
CREATE TABLE frag_table1(
            col1 integer,
            col2 char(20),
            ...)   FRAGMENT BY ROUND ROBIN
               in dbspace1, dbspace2 ;
```

Fragment By Expression

```
CREATE TABLE frag_table2(
        col1 INTEGER,
        col2 CHAR(20),
        ...)   FRAGMENT BY EXPRESSION
                col1 <= 10000 AND col >= 1 IN dbspace1,
                col1 <= 20000 AND col1 > 10000 IN dbspace2,
                REMAINDER IN dbspace3;
```

- REMAINDER IN dbspace will hold rows that do not evaluate into any of the expressions. (Always placed last; REMAINDER fragment cannot always be eliminated from scans.)
- Row is put in the first dbspace where the expression evaluates to true.

Expressions allowed in Fragment By Expression

- Can use any column in the table as part of the expression.
- No subqueries or stored procedures.
- Up to 2048 fragments and conditions.
- >, <, >=, <=, IN, BETWEEN
- AND, OR
- NOT, MATCH, LIKE
- HASH functions (MOD)
- Keep expressions simple.
- Avoid using these by themselves (fragment elimination does not occur).
  - !=, IS NULL, IS NOT NULL
- See “Effectiveness of Fragment Elimination” in the IBM Informix Dynamic Server Performance Guide.

Using Hash Functions

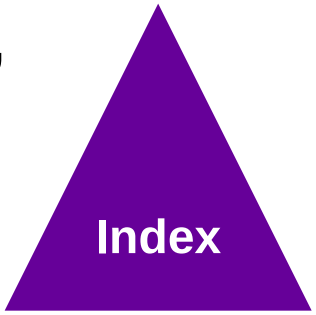
```
CREATE TABLE frag_table(
      customer_num SERIAL,
      lname CHAR(20),
      ...)   FRAGMENT BY EXPRESSION
               MOD(customer_num, 3) = 0 IN dbspace1,
               MOD(customer_num, 3) = 1 IN dbspace2,
               MOD(customer_num, 3) = 2 IN dbspace3;
```

- **Hash expression allows fragment elimination when there is an equality search (including inserts and deletes).**
- Useful when data is accessed via a particular column but the distribution of values within the column is not known.

Indexes on Fragmented Tables

- If you do not want to fragment your index, specify the dbspace to put the index in

```
CREATE INDEX <idx1>... IN
      DBSPACE <dbspace1>;
```



- If dbspace is NOT specified, index will be fragmented, following the fragmentation scheme of the table -- **BAD** if table is fragmented by ROUND ROBIN.

Indexes (cont'd)

- You can fragment indexes by expression.
- If fragmenting a unique index, the unique column must be used in the fragment by expression.
- System indexes created automatically to support constraints are NEVER fragmented. They are created in the dbspace where the database is created.

Index

Create Index Statement

- Index Fragmented By Expression

```
CREATE INDEX idx1 ON table1(col1)
            FRAGMENT BY EXPRESSION
            col1 < 10000 IN dbspace1,
            col1 >= 10000 IN dbspace2;
```

- Index Not Fragmented

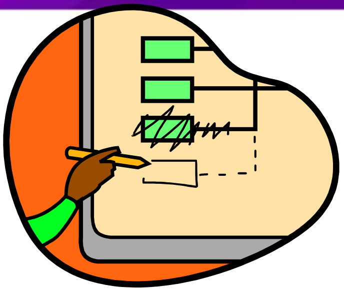
```
CREATE INDEX idx1 ON table1(col1) IN dbspace1;
```

Fragment Permissions

- Permissions *may* be granted on a fragment basis.
- If fragment permissions are to be used, fragmentation scheme must be “by expression”.
- A user who has table privileges on a fragmented table has the privileges implicitly on all fragments of the table.

```
REVOKE FRAGMENT ALL ON
        customer (p1) FROM harry

GRANT FRAGMENT ALL ON
        customer (dbspace1, dbspace2) TO millie
```

Alter Fragment Statements



- ALTER FRAGMENT ... INIT
  - Initialize a new fragmentation scheme on an existing table or index.
- ALTER FRAGMENT ... ADD (or DROP)
  - Add an additional fragment to (or drop a fragment from) an existing fragmented table or index.
- ALTER FRAGMENT ... MODIFY
  - Modify a fragmentation expression or dbspace in a table or index fragmentation expression.
- ALTER FRAGMENT ... ATTACH (or DETACH)
  - Combine tables with identical structures into a single fragmented table, or move a fragment into a separate table.

ATTACH

- Combine two non-fragmented tables with identical schemas into one fragmented table.
  - Before IDS 10.00, the two tables had to be in separate dbspaces.
  - With the partition name feature however, the two tables can be in the same dbspace (must use “as partition <partition name>” clause).

Alter fragment on table mytab1 ATTACH mytab1 as partition p1, mytab2 as partition p2;

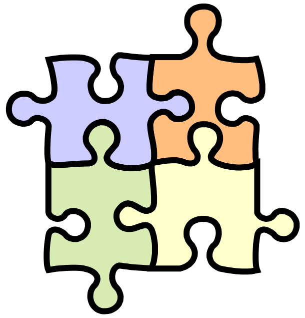
- Combine a non-fragmented table to a fragmented table.

Alter fragment on table mytab1 attach mytab3 as partition p3;

- Can use BEFORE and AFTER clause if using fragmentation by expression.

Alter fragment on table f1 attach f1 as (col1 <= 100 and col1 > 0), f2 as (col1 > 100) ;

Alter fragment on table f1 attach f3 as (col1 <= 0) before dbspace1;



ATTACH

- Referential, primary key, unique constraints, and serial fields are not allowed in the tables being attached.
- Surviving table cannot have check or not null constraints.
- Index rebuilds can be avoided IF:
  - there is no data overlap
  - index for the new fragment is on the same set of columns as the index of the target table.
  - index has the same properties (unique, duplicate) as the index of the target table
  - index for the new fragment is not in any of the dbspaces used by the target table's index fragments.

DETACH

Separate a fragmented table into multiple tables.

```
Alter fragment on table f1 DETACH dbspace2 f2;
Alter fragment on table mytab1 DETACH
partition p2 mytab2;
```

- Detach command does not work on fragmented tables with rowids.
- Index rebuilds may be necessary if the index fragmentation strategy is not similar to the table fragmentation. (If similar, the index fragments corresponding to the detached fragment will be dropped.)

Q&A

Thank you!!