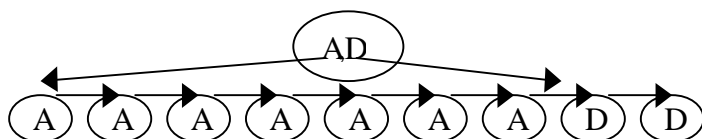


## Duplicates Slowing You Down?

Are you suffering from database performance problems, but don't understand why? Performance problems may be caused by many factors from over utilization to poor application design, or even inefficient algorithms used by DBMS itself. In this article, we will look at one such problem where DBMS itself may be contributing to performance problems – and discuss possible workarounds.

Large numbers of duplicates in sorted sets can cause severe performance problems when *inserting* new records into the set. This problem occurs when a record with a new unique key value is inserted with a value that sorts just after a key value that has many duplicates. In this case, DBMS needs to insert the new entry *between* the two key values – at the end of the long list of duplicates, and immediately before the next key value.

Assume we were to insert a "B" record into the following list: AAAAAADD



DBMS uses the index structure to quickly determine that "A" is the last key value that is less than the value to be inserted,

"B". DBMS then uses this index structure to locate the first "A" record. From there, it simply *walks the chain set* until it finds a record with a key value greater than "A" – each duplicate value may result in an I/O. Because the index structure itself needs to be updated, DBMS maintains an exclusive lock on the index – causing other processes needing this index to stall during this insertion.

*(Continued on page 8)*

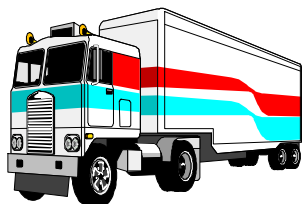
**Earn**  
**a \$2,000 Bounty.**  
**Help us hire our next**  
**Database Expert.**  
(Call for details.)

Software Concepts International, Inc.  
P.O. Box 908  
Granby, Connecticut 06035-0908  
U.S.A.

Table of Contents	
<b>Duplicates Slowing You Down?</b>	<b>1</b>
<b>We're On the Move Again</b>	<b>2</b>
<b>Investigating Performance Problems</b>	<b>3</b>
<b>Announcing Worldwide Service Agreement</b>	<b>3</b>
<b>Focus On Newest SCI Associate</b>	<b>5</b>
<b>DBA Training</b>	<b>6</b>
<b>Career Opportunities</b>	<b>7</b>
web site: <a href="http://www.sciinc.com">www.sciinc.com</a> .	

## We're On the Move Again!

SCI is moving its base of operations this summer.



We are picking up shop and moving the whole kit and caboodle to Nashua, New Hampshire in the summer of 1999.

SCI is truly an international company, with customers all over the world and little need to maintain a local presence. This is a perfect time to make the move, as we are poised for major growth in the next two to five years.

The move is actually expected to be smooth - basically a "non-event". This is all due to Bryan Holland, the forward-thinking creator of the SCI tool set. His design is such that changes, even as major as this, can be made extremely easily and with little or no risk.

Nonetheless, we will be providing a smooth migration of service for our customers. An improved technical infrastructure will be in place and fully tested before any transition. The changes will be relatively transparent to clients. We'll be contacting individual customers at the appropriate time to explain plans for the migration of support for their databases, and they will have the opportunity to provide input and be a party to all contingency plans.

The move was prompted by our realization that north central Connecticut was not a hot bed of DBMS talent. It has been hard to obtain a qualified applicant pool. Over the past several years there

has been a general migration of technical talent away from Connecticut as companies downsized and then downsized some more. Connecticut is one of the few states whose population has actually decreased in the last decade. And people from other places saw no reason to want to move to such an area.

On the other hand, Nashua is in an area that has been attracting technical talent over the past several years. Many high tech companies, both large and small, have moved in or expanded their operations in the area. The Rte 128 corridor northwest of Boston, long known for its high concentration of high tech companies, is within easy commuting distance.

Furthermore, several cities and towns in southern New Hampshire have the distinction of being among the best places to live as determined by Money magazine and other organizations.

With major installations of Compaq (Digital) and Oracle right down the road, we will be in good company. Come visit us!

## DBA *dvisor*

The DBAdvisor is published by **Software Concepts International, Inc.** We will consider for publication all submitted manuscripts and photographs. SCI welcomes your articles and suggestions but cannot be responsible for loss or damage. All information presented is believed to be accurate. However, we cannot be responsible for their accuracy or application.

COPYRIGHT © 1999 by *Software Concepts International, Inc.* All rights reserved. No part of this publication may be reproduced in any form without written permission from the publisher, and the inclusion of this copyright notice.

# Investigating Performance Problems

The CODASYL DBMS utility DBO/SHOW STATISTICS provides a dynamic view of the database environment. Use it to diagnose database performance problems and to determine if an application is using system resources effectively.

Performance issues discussed in this article include:

IO performance, effectiveness of buffering, index statistics, database verb usage and database locking statistics. Looking at each performance issue in turn, we'll discuss the information DBO/SHOW STATISTICS provides, and then present some alternatives for remedies.

DBO/SHOW STATISTICS collects statistics while the database is open and the application is running, showing dynamically what is happening.

Generally, it makes sense to sample data over relatively long periods of time (such as a work shift). However, significant response time problems may

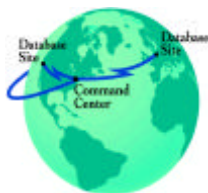
exist in short bursts and may not be obvious when examining long-term averages. Increasing the sampling frequency during problem times might provide the information needed to isolate the specific situations causing the delays.

Some of the statistics collected tell us generally what is happening with the database; others relate to parameter settings which can be changed to improve performance. With a well-designed database, we recommend a "steady-state" approach to analyzing statistics. By monitoring the statistics on a regular basis, the DBA becomes familiar with normal performance, notices deviations more easily, and can make timely adjustments.

Help pinpoint problem areas affecting performance with DBO/SHOW STATISTICS. Performance often

*(Continued on page 4)*

**Become familiar with "normal" performance for your database with "steady-state" monitoring.**



## Announcing Worldwide Service Agreement

SCI and Infineon Technologies AG, Siemens Semiconductor AG announce an agreement to provide Infineon Siemens with Remote Database Administration support worldwide.

Infineon Technologies Siemens Semiconductor, a leading manufacturer of semiconductor products worldwide has recently signed a Database Administration support agreement with Software Concepts International, Inc. (SCI), for the management of their WorkStream™ databases at all major semiconductor manufacturing sites worldwide. Infineon Siemens chose SCI based on their worldwide reputation as a leader in DBA support services, and their ability to support the high availability requirements of their semiconductor operations

degrades when databases grow in size/volume. If statistics are gathered as the database grows, comparisons of differences in transaction rates can be made. The performance of system-owned sorted sets also degrades at a certain level; by collecting a history of statistics at various database sizes, you can identify the level at which degradation occurs.

Let's look at specific performance issues and see how the information displayed on the screens in DBO/SHOW STATISTICS can assist in diagnosing problems.

**I/O Performance.** When an area is too full, DBMS may have to search several pages to find sufficient available space. The *Records Statistics* screen can provide some information about how much I/O is being done to find available space. Compare the 'average per transaction' for 'records stored' and 'pages checked.' If the average for 'pages checked' is significantly higher than for 'records stored', it means that DBMS is not able to store records on the target page and has to look at other pages before the store. This could mean the area is full, or that records are too clustered and that SPAM thresholds are probably not set optimally. Behind the scenes, DBMS calculates the target page based on the placement clauses in the storage schema and retrieves that page into the current buffer. If the target page has enough available space and the page is not already at the T3

threshold, the record is stored. If the page has insufficient available space or is already at T3, DBMS looks on other pages in the current buffer. This doesn't incur additional I/O, but the count of 'pages checked' goes up so the ratio to 'records stored' will be greater than one. If the ratio is 1, then DBMS is always able to store a record on the target page, which might mean that there is too much free space and no record clustering.

If you determine that an area is too full, changing the page size and allocation for the area or directing records to a new area can relieve the problem.

**Effectiveness of Buffering.** Sometimes much swapping of pages in and out of buffers occurs because of buffer pool overflows or lock conflicts, which occur when DBMS writes updated pages back to the database before the transaction is actually committed. This happens when a process requests an updated page in use by another process. The RUJ file records this read/write activity so that records can be reconstructed in the event of a

rollback. The *Summary I/O Statistics* screen shows the average RUJ file reads and writes per transaction. If the number

of 'RUJ writes' is greater than zero, then updates are occurring. If the number of 'RUJ reads' is greater than zero, then transactions are rolling back. If this number is large, you might investigate why the transactions aren't being completed.

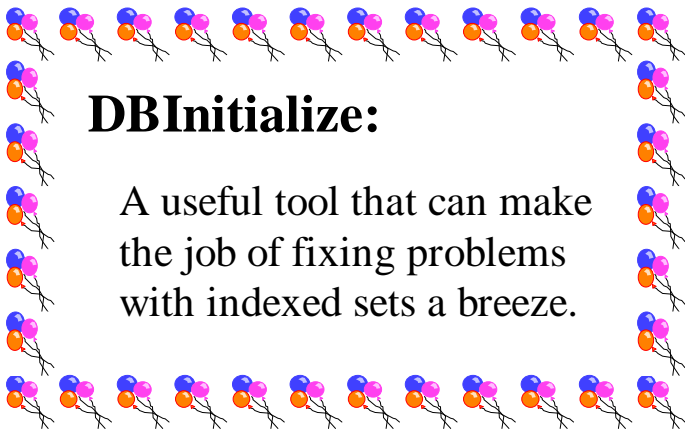
The *Physical IO Statistics* screens can give some clues about buffers. On the *Writes* version of the physical IO displays, look at 'pool overflow' and 'lock conflicts' in the 'unmarked buffer' section of the screen. If you see a relatively high number of pool overflows per transaction, you might need to increase the size of the buffer pool. The 'lock conflict'

**DBO/SHOW STATISTICS  
is a useful tool for helping diagnose  
performance problems.**

### How To Reach SCI

- visit our web site ([www.sciinc.com](http://www.sciinc.com))
- e-mail us at [info@sciinc.com](mailto:info@sciinc.com)
- call us @ 860 653-7718 or toll free in the United States, (888) CODASYL (263-2795)
- fax us @ 860 844-7035.





## DBInitialize:

A useful tool that can make the job of fixing problems with indexed sets a breeze.

*Duplicates (Continued from page 8)*

Duplicate keys could be eliminated or minimized by adding an additional field to the end of the key and populating this new field with either a unique sequence number or a “random” value (thus minimizing the number of duplicates). This solution causes two problems – first it will require changes to the database schema and your application. This is not a trivial task. In addition, this will create additional entries in the index structure itself, thus resulting in additional overhead – For example, if previously there was one key value with 1000 duplicates, there would be only one entry in the index. If we add “sequence” number to this index, thus making all key values unique, the index itself will contain 1000 entries (one for each unique key value). So, the overhead of the index structure itself may increase significantly.

### Adding Dummy Records:

By adding “dummy” records with unique key values that sort immediately following long lists of duplicates, you can minimize the impact of this performance problem. However, this assumes that these “dummy” records will not adversely affect your application. The other problem with trying to add these dummy records is that the process of inserting these dummy records can be extremely slow – after all, for each dummy record inserted, DBMS must walk all of the preceding duplicate entries (the situation we are trying to avoid in the first place).

☺☺☺

*Investigating (Continued from page 5)*

RECONNECT, STORE/PLACE) or reading activity (FIND/FETCH/GET). And check how many verbs are executed per transaction.

Verb statistics are best used to compare against “normal” performance, and to check against other statistics. If you find more read transactions, investigate IO performance. If there are more update transactions, investigate locking statistics. Infrequent or low numbers of COMMITs could also be causing lock conflicts.

**Database Locking Statistics.** Too many lock conflicts might be an indication of database bottlenecks or long transactions. On the *Summary Locking Statistics* screen, look at statistics for ‘rqsts not queued’, ‘rqsts stalled’ and ‘rqst deadlocks.’ Relatively high numbers mean contention problems or that the database design is not optimal.

The ‘blocking ASTs’ statistics indicate contention for frequently-accessed data. If you see a high number of ‘blocking ASTs’ relative to the number of ‘locks requested’ (more than 20%), **and** you have Adjusted Locking Granularity enabled, you may want to try disabling ALG.

If the ‘stall time x 100’ statistics look high, investigate details about the types of locks for that particular event. From the *Display\_menu* list, select *Locking – (one stat field)*, then select *stall time*

**Sign Up, Now!!!**

**CODASYL DBMS  
DBA Training**

**Software Concepts International, Inc.**

Next Class: November 8-12, 1999

at SCI's offices in Nashua, NH

Register @ (860) 653-7718

(Continued from page 6)

x100 to see which resources are involved with the longest stalls. Statistics for 'quiet point lock' are related to backups. If these numbers are high, consider adjusting or rescheduling the work load so that backup processes can run without interference. 'Freeze locks' refer to recovery processes that occur when a process aborts abnormally or is killed. If these statistics seem high, investigate why processes are being terminated. High rates for 'page locks' may indicate that the disk drives are slow, while high rates for 'record locks' indicate contention for data or index structures.

Lock contention can be related to transaction durations. Ideally, shorter transactions reduce database contention for records. However, shorter transactions may minimize the effectiveness of database buffering since every COMMIT and ROLLBACK

requires that all buffers be flushed to disk. In high contention environments, shorter transactions are preferable since this maximizes concurrency. In low-contention environments, longer transactions are preferable so that DBMS is able to optimize buffer utilization and asynchronous writes of the buffers to disk.

To reduce lock conflicts or deadlocks, optimize your database design, shorten transaction durations, or disable Adjusted Locking Granularity.

In summary, DBO/SHOW STATISTICS is a useful tool for helping diagnose some performance problems. In the next issue we will talk about additional ways you can use the tool to improve the performance of your system and database.



## Career Opportunities

Software Concepts International, Inc. is looking for a few talented individuals to join the company to help facilitate further growth of its successful international business.

### **CODASYL DBMS DBA**

Duties: Provide expert DBA service, remotely, to clients in Asia, USA, and Europe.

Provide expert on-site consultation as needed.

Knowledge of CONSILIUM WorkStream® or CA/ASK MANMAN®, a plus.

**Competitive compensation, flexible work arrangements, excellent benefits.**

### **Informix, Rdb or Oracle Expert :**

Duties: Migrate SCI's unique, comprehensive tool set to these alternate database environments. Provide expert DBA service, remotely, to clients in those environments.

Provide expert on-site consultation as needed.

Knowledge of system management in the HP-UX environment, a requirement.


(Continued from page 1)

Once the "B" record is inserted, the set now looks as follows:

AAAAAABDD

Adding a record with a value of "C" will be efficient, because the preceding key value, "B" is unique – thus, DBMS will not have to walk a long list of duplicates to insert the new entry.

Adding records that match existing keys is efficient – DBMS simply uses the index structure to locate the existing entry and then adds the new record to the beginning of the list and the index is updated accordingly. Thus, in the above example adding additional "A" records will be efficient.



**Order Now:  
Remote DBA  
Services with  
Hot Stand-by**

This performance problem only surfaces when inserting *new* key values that sort immediately following a key value that has many duplicates – in this case, DBMS will walk each duplicate member to find the location in which the new record should be inserted – possibly resulting in an I/O for each duplicate!

### **How to diagnose this problem:**

How do you know if you are suffering from this problem? First, you will want to get a list of all sorted sets in your database where duplicates are allowed. This information is available from the schema (\$ DBO/DUMP/SCHEMA <root-file>). For each sorted set, you will want to know which sets have many duplicates. Extracting the data from the database and counting the number of duplicates for each key value can do this. DBO/UNLOAD is a great tool for this. Applications, such as WorkStream, which have built in performance-monitoring tools can also be extremely helpful – look for "anomalies" in the time and I/O required to perform DML STORE, CONNECT and RECONNECT verbs. Then determine if these records are members of any sorted sets that allow duplicates. DBO/SHOW STATS (stalls) can also provide insight into what

resources are involved in a significant number of lock problems – if the index records are frequently involved in stalls, this also suggests that duplicate keys may be the problem.

### **Understanding the Indexed Set Mode:**

An INDEXED set in DBMS is based on a simple CHAIN set, with one notable exception – in *addition* to the simple chain set that links each member of the set, an INDEXED set also contains a balanced-tree index structure that points to the first record with a unique key-value. Thus, when performing lookups within indexed sets, DBMS is able to quickly traverse this index structure to find the first matching key. When inserting a new key value, DBMS uses the index structure to locate the *first* record with a key value that sorts *before* the new key to be added. Once it finds this record, it simply walks the CHAIN set to find the correct insertion point – each duplicate member may result in an I/O – so if there are 10,000 duplicates, you are in for a long wait.

### **Application Problem or DBMS Problem?**

Clearly, avoiding duplicates with the application would eliminate this problem...but is this *really* what you want to do in your application? In SCI's opinion, this is a DBMS problem, and not a problem with the underlying application. DBMS should simply (maybe not so simple), find the first record with a key value that is *greater* than or equal to the key to be inserted, use the CHAIN set's PRIOR pointer to find the insertion point. If this algorithm was used, there would be a maximum of 2 I/Os – A significant improvement!!!

### **Duplicates in the "real world".**

Okay, so maybe Oracle is not ready to fix the problem in the next release... so, what are we to do?

### **Workarounds:**

There are a few ways that you may be able to workaround this deficiency in DBMS. The first would be to modify the application so that the number of duplicate key values is either eliminated or minimized. The second would be to insert "dummy" records with key values that sort just slightly higher than those keys with long lists of duplicates.

### **Eliminating or Minimizing Duplicate Keys:**

(Continued on page 6)