Database Area Sweep Considerations

In today's data processing environment most applications focus on the availability and performance of online functions. However in the majority of large database applications the performance of the batch environment can be just as critical due to the large volume of data that must be processed within typically small windows. This can be especially true in applications where all entities within the database must be processed on a daily basis. The initial response to these types of applications is to use an 'area sweep' to access the database. But to ensure the successful completion of the batch routine within the prescribed timeframe it may be necessary to review the database structure to be navigated and the manner in which the sweep is performed.

In order for area sweep processing to be efficient it is necessary to insure that all of the data is accessed by the smallest number of physical I/O operations. This is where database design can have a great impact on the run-time environment. First it is important to define what is meant by an area sweep from the logical and physical perspectives. Logically an area sweep means accessing all occurrences of a record type by making one serial pass of the database area. This typically implies that the order in which the record occurrences are accessed is in physical sequence and not in a logical sequence relative to some data element's contents.

Area Sweep Implementation Methods

Physically an area sweep can be implemented in one of two basic methods. The more conventional method is the use of the 'OBTAIN/FIND NEXT record-name WITHIN area-name' command. This causes CA IDMS to use the dbkey identified as current of area as its starting point and access the next record occurrence of the specified type with a higher dbkey. It should be noted that omission of the 'record-name' operand from the command will cause the CA IDMS search to be satisfied by the next record occurrence located within the area with a higher dbkey regardless of its record type. If a record is not located on the same page as the current record of the area, the next page is read and searched for an occurrence of the specified record type. This type of processing will read every page in the area resulting in an equivalent number of physical I/Os.

The second method is to use some type of set to connect all occurrences of the target record type. This method can be differentiated by the types of sets that can be used to connect the records. The older method is to use a one-of-a-kind (OOAK) record as the owner of a chain set and to access all of the target record occurrences by walking the set through the execution of the 'OBTAIN/FIND record-name WITHIN set-name' command. This structure will result in CA IDMS accessing only those pages that contain record occurrences and therefore result in a significant reduction in the number of I/Os needed to sweep the database when fewer record occurrences exist than the number of pages defined within the area.

A drawback with the OOAK structure can be the options available for sequencing the records within the set. Since a chained set is involved, the insertion or deletion of a record from the set requires that NEXT and possibly PRIOR pointers in adjoining record occurrences must be maintained resulting in additional I/O and possible dbkey contentions. Also if the set is defined with an order of SORTED, the insertion point for a record being inserted is typically located by walking the set. For long sets this can result in a significant amount of physical I/O.

Since the introduction of indexing it is more common for applications to use a system-owned index on the target record type instead of an OOAK record and chain set. An index can provide all of the functionality of the OOAK structure but usually with less overhead needed to locate the new record's insertion point and to connect the occurrence. This is especially true when the order of the index is defined as SORTED. To accomplish the area sweep the 'OBTAIN/FIND record-name WITHIN set-name' command is used. This results in the occurrence that is current of set of the index to be used as the starting point of the search. CA IDMS then looks at the next entry in the level-0 index record (SR8) to determine the dbkey of the next data record to be accessed within the area.

To determine which method to use it is a good idea to consider some of the less obvious aspects of each structure. The use of an index or an OOAK structure over the 'OBTAIN/FIND NEXT record-name WITHIN area-name' method will only reduce I/O requirements when there are pages within an area that do not contain any desired record occurrences. When each page contains at least one record occurrence and the chain set or index have been ordered to insure that all records on a given page are referenced prior to moving on to the next page, the I/O required to access all data record occurrences is equal. In these situations the use of an index or OOAK structure may actually add a bit of additional I/O and CPU overhead to process these structures.

The index method of area sweeps may be the recommendation when the target record occurrences sparsely populate an area and isolating these records into their own densely packed area is undesirable. An index may also be more efficient if all of the occurrences of the target record type were not to be included in the sweep according to some pre-determined criteria. In this situation it would be possible to only connect to the index those records to be included in the area sweep. This removes the need for the application to access each record and then determine whether further processing is necessary.

Because indexes provide the same sequencing options as a chain set in addition to sorting by dbkey, the use of OOAK records is becoming a less frequently used option. OOAK records may still be required to contain database-wide totals, flags, or counters but are seldom used to own chain sets of records for area sweep type processing. This is due to the greater efficiency encountered when inserting or removing non-VIA records from an index structure than from a chain set or when accessing a single set member through its symbolic sort key. Also, due to the nature of the index structure the possibility of dbkey bottlenecks can be less than when an OOAK record is employed.

Design Evaluation

To evaluate the performance of an existing routine or to verify the results of a planned implementation it may be necessary to examine some statistics generated by CA IDMS to insure that the physical I/Os generated are at an expected or acceptable level. The numbers that should be initially examined are the PAGES READ and the RECORDS CURRENT OF RUN-UNIT values. These numbers can be obtained from the statistics blocks written to the journal files and reported through JREPORTs, 'ACCEPT STATISTICS' DML commands, or the application monitor portion of the CA IDMS Performance Monitor.

If an area sweep is being performed these numbers can be used to determine whether an excessive amount of I/O is being generated relative to the amount of work actually being accomplished. For example, if the statistics indicate that 1000 records became current of run-unit while 5000 pages were read it would be worth the effort to investigate the possibility that the 'OBTAIN/FIND NEXT record-name Within area-name' command was being used against an area that contained a small number of target records scattered across approximately 5000 pages. The database being navigated may be a good candidate for a system-level index or isolation of the target records into their own condensed area.

Examination of these statistics may also point out where the OOAK or index methods of sweeping may be inefficient. When these methods are used the number of records that were current of run-unit may be equal to or greater than the number of database pages that are physically read. However if the number of pages read is greater than the number of pages in the area a problem with the sequence of the set or index may exist. To verify the efficiency of these structures you could compare the number of pages in the area through PRINT SPACE or IDMSDBAN reports to the number of pages read to determine whether pages had been read multiple times. If this proves to be the case, the use of the target record's dbkey as the index sortkey or modifying the program to use the 'OBTAIN/FIND record-name WITHIN set-name' command could reduce the program's run-time.

You should be sensitive to the fact that the PAGES READ statistic will reflect pages read from all areas accessed by the run-unit which may mask the number of pages that were read as a result of the sweep. To accurately determine the number of I/Os needed to walk an OOAK set or index set may require execution of a program that just walks the target relationship and ignores all other record types. In many cases the number of I/Os can be estimated. If the target records are not stored VIA the OOAK record or the index you may assume that one I/O will be required per record occurrence regardless of the probability of multiple records being stored on the same database page. If the number of target records in the area exceeds the number of pages in the area an OOAK or non-dbkey sequenced index is probably not the most efficient implementation option.

Programming Considerations

On the programming side of area sweeps it is important that programmers are aware of the way that CA IDMS performs area sweeps when using the 'OBTAIN/FIND NEXT record-name WITHIN set-name' command. As stated earlier, this command uses the dbkey that is defined as the current of area as the starting point for the search of the next record occurrence. As a result it is possible that under the proper conditions a routine could bypass desired record occurrences or find itself in a processing loop. The diagram in Figure 1 can be used to illustrate the following scenarios.

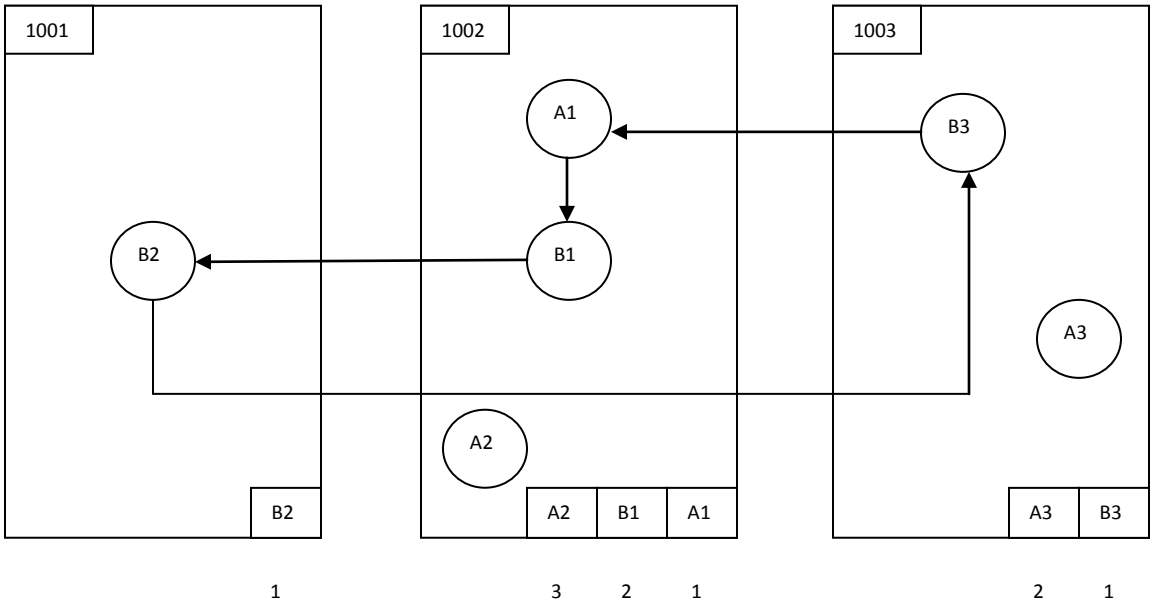

Figure 1

The first scenario is an example of a record being bypassed by a program performing an area sweep.

1. As a result of an 'OBTAIN NEXT A WITHIN AREA1' command occurrence A1 is returned to the program. This occurrence has a dbkey of 1002:1 which becomes current of area AREA1 and set A-B.
2. The program issues multiple 'OBTAIN NEXT B WITHIN A-B' commands to walk the A-B set until it locates occurrence B3. The dbkey of B3 is 1003:1which becomes the current of area AREA1 and set A-B.

3. According to some program criteria the program determines that it is no longer necessary to continue walking the A-B set and issues another 'OBTAIN NEXT A WITHIN AREA1' command to resume sweeping the area. Since the current of area dbkey is 1003:1 CA IDMS starts its search at that point within the area. As a result the next occurrence of an A record to be located will be occurrence A3 with a dbkey of 1003:2. The program is never aware that occurrence A2 with a dbkey of 1002:3 was bypassed.

The possibility of a processing loop is also caused by a similar situation as defined in the following scenario.

1. As a result of an 'OBTAIN NEXT A WITHIN AREA1' command occurrence A1 is returned to the program. This occurrence has a dbkey of 1002:1 which becomes current of set A-B and area AREA1.
2. The program issues multiple 'OBTAIN NEXT B WITHIN A-B' commands to walk the A-B set until it locates occurrence B2. The dbkey of B2 is 1001:1 which becomes current of set and area.
3. According to some processing criteria the program determines that it is no longer necessary to continue walking set A-B and issues an 'OBTAIN NEXT A WITHIN AREA1' command to continue the area sweep. Since the current of area dbkey is 1001:1 CA IDMS starts its search for the next record type A occurrence within the area from this point. As a result the next occurrence of record type A to be located will once again be A1 with the dbkey of 1002:1. The program has no way of knowing that this occurrence has already been processed and will repeat the sequence of instructions until it is manually aborted.

Neither of these situations would have occurred if the programs had continued the navigation of the A-B set until end-of-set had occurred since at that point the dbkey of occurrence A1 would have been set to be the current of area. However there are situations in which you may not wish to incur the overhead or processing complexity of accessing the remaining members of the A-B set. In those situations it is possible to avoid the problems described by always issuing a 'FIND CURRENT record-name' command before the subsequent 'OBTAIN/FIND NEXT record-name WITHIN area-name' command. The record-name operand is the name of the record type which is the target of the area sweep. The 'FIND CURRENT record-name' command will reset the current of area dbkey to the value representing the last accessed occurrence of the name record type. This insures that the area sweep would resume at the occurrence accessed by the previous 'OBTAIN/FIND NEXT record-name WITHIN area-name' command.

The situations described above would not occur when using the index or OOAK methods of area sweeps since the current of the named index or OOAK set is used to resume the walking of the index or set. This currency will not change as a result of accessing another record type through a different index or set.