CA IDMS Indexing Page Displacement

Dick Weiland
CA Technologies
Principal Software Engineer

The DISPLACEMENT parameter defines the number of pages that level-0 SR8 records are displaced from the index set's owner when they are stored. A DISPLACEMENT parameter can be specified for an index as part of the set definition within the database's schema or as a symbolic parameter defined during the creation of the index's area and passed through the DMCL used at run-time. Refer to the CA IDMS Database Administration Guide for the proper coding of these entities.

This parameter is perhaps the most powerful tool to affect the performance of a system-owned sorted index that a DBA has when defining an index. To understand why this parameter is so valuable it is necessary to understand the impact of the number of levels within an index structure, the difference between level-0 and intermediate SR8 records, and the impact of having to split records before they contain the number of entries specified by the Index Block Count (IBC).

Index Levels

An index structure is composed of three main components:

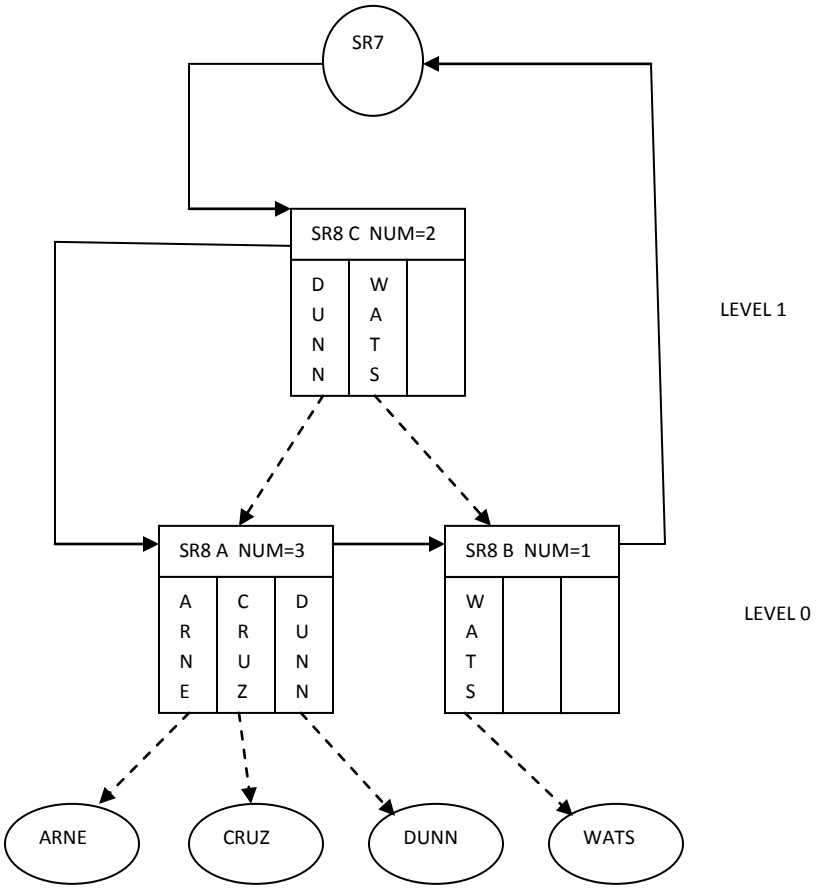
- a set owner, which in the case of a system-owned index is an occurrence of a record called an SR7
- records known as SR8 records which make up the bulk of the index structure
- the data records being indexed

The SR8 records are connected to the index's SR7 occurrence through a standard chain set. However these records also form a pointer array in a binary-tree arrangement.

When the first record is connected to the index, the SR7 record is stored and an SR8 record is created that contains an entry representing the data record. This SR8 is at the lowest level of the binary tree and all SR8 records at this level are known as level-0 SR8 records. Additional data records inserted into the index result in entries being added into this first level-0 SR8 until that SR8 occurrence contains the number of entries specified by the IBC. A subsequent data record being inserted into the index causes the first level-0 to be split into two SR8 records. One of the major rules of indexing states that the highest level of an index structure can only contain a single SR8 occurrence. Prior to the split the highest level of our index was level-0 but the split resulted in two SR8s at level 0. To comply with the stated rule a new level, level-1, must be spawned and we now have a two level index structure. All SR8 records not on level 0 are referred to as intermediate SR8 records.

Diagram 1 represents an index on EMPLOYEE records where the symbolic key is the employee's last name. The IBC for this index was defined as 3 indicating that any SR8 record can have no more than entries.

Diagram 1



The concept of levels within the index is significant in that the number of levels making up the index has a direct correlation to the amount of work needed to access a data record using the index's binary-tree structure. Assume that an attempt is being made to access the data record whose symbolic key is DUNN through the index. To locate the data record CA IDMS must read the SR7 record which gives direct access to SR8 C. SR8 C then has a pointer to SR8 A which in turn points to the desired data record. Our two-level index required that two SR8 records had to be accessed in order that the desired data record could be located. In many cases these SR8 occurrences would be on different pages implying

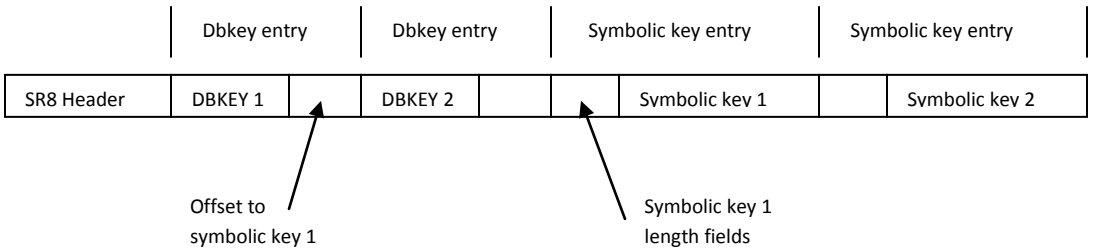
that two physical I/O operations would have to be performed. If the index contained five levels it can be assumed that five SR8 occurrences would have to be accessed increasing the overhead of navigating through the index structure.

A general recommendation is that medium to large indexes contain 3 or 4 levels which is generally controlled by the number of data records to be indexed and the IBC selected for the index. If an IBC is selected that is too small the number of levels within the index may increase since more SR8 record occurrences will be needed to contain entries for all of the data records. Selection of an IBC that is too large can cause SR8 records to be so large that it is difficult to define a page size for the index's area that is large enough to accommodate multiple SR8 occurrences and can also lead to increased dbkey contention in indexes that have a high volume of insertions or deletions. Proper sizing of an index is described in the CA IDMS Database Design Guide.

Level-0 and Intermediate SR8 Records

The need for a DISPLACEMENT parameter is increased when the difference between the manner in which level -0 and intermediate SR8's are maintained is considered. Both types of SR8 records have the same general structure. Diagram 2 represents the general format of an SR8 record used by an index sorted by a symbolic key. For the purposes of this discussion the IBC for the index involved is considered to be 2.

Diagram 2



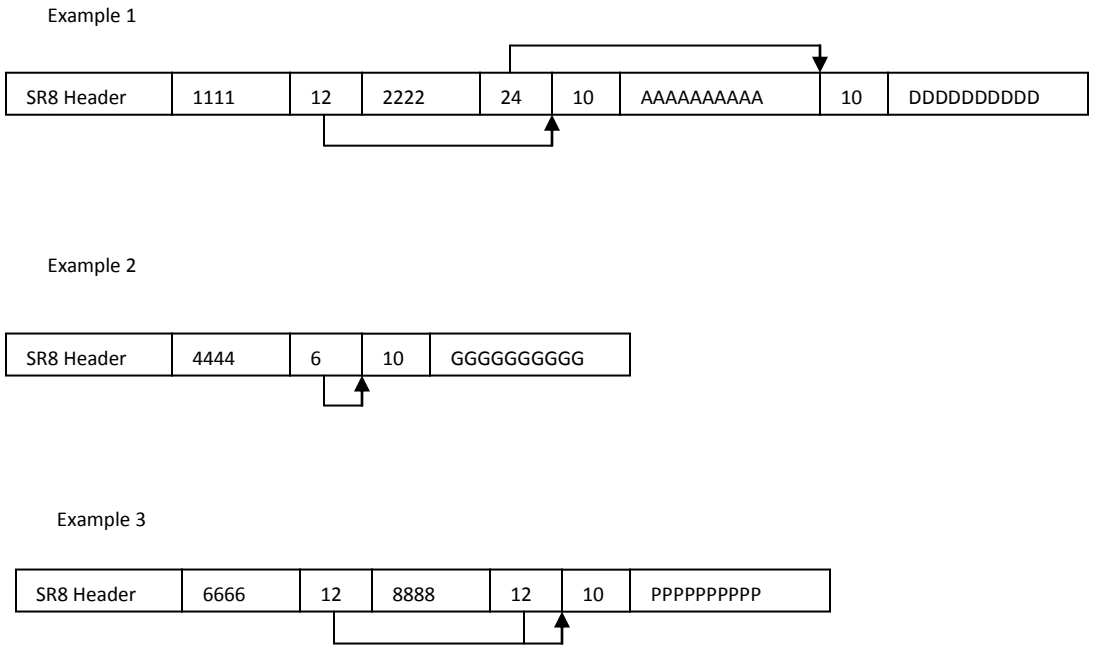
Each SR8 record contains a 32 byte header. This portion of the record contains the pointer positions that connect the SR8 to the index owner and also contains a pointer to an SR8 on the next highest level in the binary-tree structure which contains the entry for this SR8. For the SR8 on the highest level of the

index this pointer will contain null values (-1). Also contained in the SR8 header are counters describing the contents of the SR8 and flags indicating various characteristics of the index.

The IBC defines the maximum number of entries that can be found in the remainder of the SR8 record. An entry can be thought of as a combination of a dbkey and the symbolic key associated with that particular dbkey. However as can be seen in Diagram 2 all of the dbkey entries occur first within the SR8 followed by all of the associated symbolic keys. The symbolic keys are sorted based on the definition of the index and the dbkey entries are maintained in a corresponding order. A 2-byte offset pointer within the dbkey entry allows CA IDMS to locate the symbolic key associated that dbkey.

Although level-0 and intermediate SR8s contain the same components, the way they are maintained on the database is very different. Despite the IBC defined for an index, the level-0 SR8 records will be stored on the database as a type of variable length record. Assuming that our index allows duplicate symbolic keys, Diagram 3 shows 3 examples of how a level-0 SR8 may be stored on the database. Once again for the sake of discussion we will consider the index's IBC to be 2. Each symbolic key will be considered to be 10 bytes long plus a 2-byte length field.

Diagram 3



In Example 1 the SR8 contains entries for 2 records with dbkeys of 1111 and 2222. Each of these records has a different symbolic key which are present within the symbolic key portion of the SR8. Given an IBC count of 2 this SR8 occurrence would be stored at its maximum length. Example 2 contains only a single entry and the space for a second dbkey/symbolic key entry is not maintained on the database. Finally, Example 3 contains entries for 2 records but these records have the same symbolic key. The SR8 occurrence will be stored with the symbolic key specified only once to save space. As record entries are added or removed from a level-0 SR8, the length of that SR8 will be adjusted accordingly.

Intermediate SR8 occurrences are different in that they are always stored as 'full size'. The term 'full size' means that when an intermediate SR8 is stored on the database its length is large enough to accommodate the number of dbkey/symbolic key entries specified by the index's IBC. If Example 2 in Diagram 3 was an intermediate SR8, the dbkey/symbolic key would be maintained within the record in the same manner but the SR8 record would contain an additional 18 bytes of unused space following the symbolic key when it was stored on the database. If a new entry were added to the record the existing unused space would be utilized to create the new entry.

This does not mean that once stored an intermediate SR8 record's length will never change. If the index is created with an IBC of 80 all intermediate SR8 records will initially be stored with enough space to hold 80 entries. But if at some point in the future the IBC was increased to 100 any subsequent insertions that would increase the entry count past 80 will result in the SR8 being expanded to include the new entry up to a maximum of 100 entries before a split was required. However each new entry added would increase the SR8 by enough space to hold a new dbkey and symbolic key even if the symbolic key was not required due to it being a duplicate of an existing key.

In the same manner reducing the IBC from 80 to 60 would cause any intermediate SR8 that contained over 60 entries to be split when a new entry is added to the SR8. Both the original and new SR8 generated by the split would be stored with enough space to contain up to 60 entries.

Premature Splits

One of the reasons to utilize a properly sized DISPLACEMENT is to minimize the occurrences of premature splits. A premature split is a situation where a new entry is to be inserted into an SR8 record that contains fewer entries than specified by the index's IBC but there is insufficient space on the SR8 record's database page to expand the SR8. When this situation occurs, the target SR8 is split as if the number of entries with the newly inserted key had exceeded the IBC. However, this results in two smaller SR8 records than would normally be created had the specified IBC actually been exceeded.

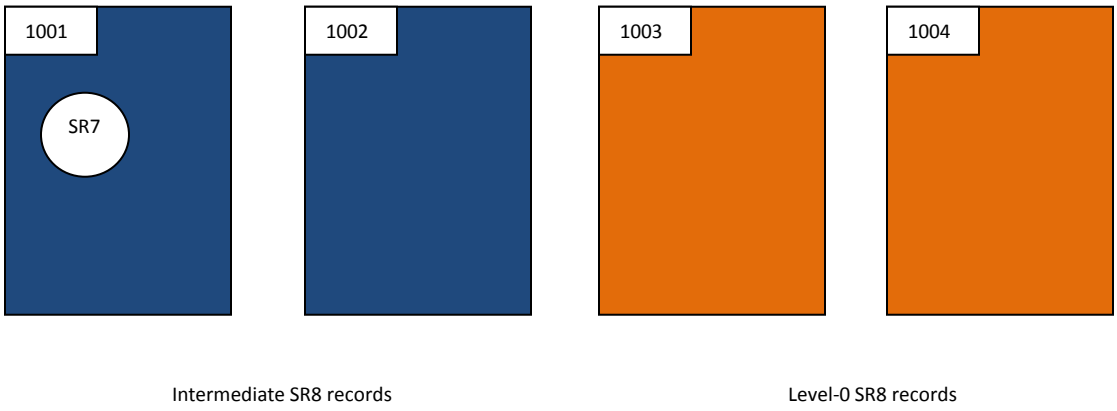
Excessive numbers of premature splits can cause the index to perform poorly since they lead to more SR8 records than anticipated which in turn will increase the number of levels needed to support the binary-tree structure of the index. As discussed earlier, more levels imply that more SR8 records must be read in order to access a record through its symbolic key, increasing a task's I/O requirements.

In addition premature splits within linked indexes will also increase the number of orphaned records existing within the database. Orphaned records tend to increase the amount of I/O required to serially walk the index.

Effects of DISPLACEMENT

The specification of the DISPLACEMENT clause tells CA IDMS how many pages to skip between the page on which the index's owner is stored and the page onto which the first level-0 SR8 is to be stored. This effectively separates the level-0 SR8 records from the intermediate SR8's within the index's area. Diagram 4 represents the pages used for the various types of SR8 records given the specification of a 'DISPLACEMENT IS 2' clause within an index's definition.

Diagram 4



When the first data record is connected to the index, the set's SR7 is targeted to a page within the index's page range by using the index name as the SR7 record's calckey. In the above example the SR7 is stored on page 1001. The first SR8 to be stored will be a level-0 SR8 and because of the 'DISPLACEMENT IS 2' clause that SR8 will be stored on page 1003. Eventually enough records will be added to the index causing the single level-0 SR8 to be split. Assuming sufficient space, the second SR8 will also be stored on page 1003. In addition the creation of a second level-0 SR8 causes the first intermediate SR8 (level-1) to be spawned. This level-1 SR8 will be stored on page 1001 with the SR7 again assuming sufficient space exists on the page.

It is important to note that the number of pages used for the index's displacement must be properly sized using the formulas available in the CA IDMS Database Design Guide. Use of the DISPLACEMENT clause does not restrict either type of SR8 record from being stored on pages intended for the other type of record. If the need arises to store another intermediate SR8 and enough space does not exist on pages 1001 or 1002 CA IDMS will continue its standard overflow search for available space and will store the intermediate SR8 on page 1003 or beyond. In the same manner overflow of level-0 SR8 records may cause them to wrap around to the beginning of the index's area and eventually result in the records being stored on pages 1001 or 1002.

Without the specification of the DISPLACMENT clause the evel-0 and intermediate SR8 records for the index would all be targeted to page 1001. This intermingling of the two types of SR8 records can lead to conflicts due to the way these records are maintained. Since level-0 SR8 records are typically smaller than full size they may take up enough space on a page such that a new full size intermediate SR8 will not fit and have to be overflowed. This can result in more physical I/O while navigating the intermediate levels of the index. In the same way, the presence of intermediate SR8 records on pages containing level-0 SR8s may result in insufficient space for the level-0 SR8 records to expand to their maximum size resulting in premature splits.

Although the DISPLACEMENT clause is a highly effective tool for the tuning of system-owned sorted indexes its value is most effective when the index structure is segregated into its own unique page range. This can be accomplished by assigning the index to its own area or by subdividing an area such that multiple indexes reside in the area but each is restricted to its own unique sub-page range. Mixing multiple indexes into the same range of pages can result in level-0 SR8 records of one index mixing with intermediate SR8s of one of the other indexes negating any benefit achieved by specifying the DISPLACEMENT feature.

Specifying the DISPLACEMENT clause for user-owned sorted indexes also has little positive impact. When an index is defined as user-owned, a user-defined data record serves as the owner of the index set. That means that if there are 1000 occurrences of the data record type defined as the index's owner there will be 1000 independent index structures within the owning record type's area. For any one of those index occurrences DISPLACEMENT will function as described earlier. However there is no way to control that the pages to which you displaced the level-0 SR8 records for one set occurrence may not be the pages which are to be used for intermediate SR8 records of another index occurrence due to the placement of that second index occurrence's owner record. As a result you will still experience the intermingling of level-0 and intermediate SR8 records within the owner record's database area.

Summary

In conclusion it is very important that the DISPLACEMENT clause is specified for any system-owned index and especially those sets that experience high volumes of insertions and deletions. It is also necessary

to properly select the proper number of pages to be used on the parameter based on the number of data records to be indexed, the IBC selected for the index, and the page size selected for the index's database area. These numbers can be reliably calculated using the formulas provided in the CA IDMS Database Design Guide. Omission of this feature or the improper sizing of the displacement area for the index can result in increases in the amount of I/O needed to navigate the index structure causing poor application performance.