

# Optimizing Record Selections - Part 1 Under the Hood of the 4D Data File

By Walt Nelson and Jean-Pierre Ribreau

4<sup>th</sup> Dimension Technical Note 97-11

Technical Notes for 97-05-May 1997

## Introduction

---

As your data file becomes larger, and as you add more 4D Client or Web connections, optimization becomes increasingly important. The purpose of this two-part Technical Note is to give you some insight into how you can optimize the handling of your record selections in 4D.

¥ In Part 1, we discuss the organization of a 4D data file and how it uses disk space and memory.

¥ In Part 2, we examine the concept of the current record, the current selection, and named selections in 4D, and discuss principles that can help you optimize performance when you have large data files and/or large numbers of users.

Unless otherwise noted, everything in this technical note applies equally to Macintosh and Windows, to 4D (single user) and 4D Client/Server, and to 4D versions 3 and 6.

## Data File and Data Segments

---

A 4D data file is composed of one or more physical files.

On the Macintosh, the data file consists of only one file on disk, usually named DatabaseName.data. On Windows, the data file can be divided into two files:

DatabaseName.4DB and DatabaseName.RSR.

However, you can also have a segmented data file with up to 64 data segments. In each data segment you can store up to 2GB of data. For the sake of simplicity in this discussion, we will use a single data segment as an example.

As a matter of fact, the discussion that follows applies to all databases, whether or not the data file is segmented. High level routines of the 4D database engine treat the data file as one huge, unique storage area, no matter how that storage is split (or not) among actual disk files.

In the following explanation, the term data segment refers to one of the data segments of a segmented data file or to the data file itself (which, after all, is the first data segment of a non-segmented data file).

## Simplified Overview of the Organization of a Data Segment

---

A data segment is composed of the following parts:

- ¥ Header(s)
- ¥ Primary Record Address table
- ¥ Secondary Record Address table
- ¥ Indexes
- ¥ The actual records

**Important:** These different parts are allocated dynamically in the data segment(s), depending on the operations performed on the database.

In this technical note, we will only take a quick look at the address tables.

### Record Address Tables

A 4D database can contain up to 255 tables (referred to as 4D tables in this section). For each 4D table, there is a set of primary and secondary address tables stored dynamically in the data file.

The Record Address tables of a 4D table describe the exact location of each record within the file. The size of the Record Address tables are determined as follows:

<u>Address Table</u>	<u>V3</u>	<u>V6</u>	<u>Remarks</u>
Primary Record Address Table	16k	32k	4,096 entries
Secondary Record Address Table	16k	32k	Up to 4,096 secondary tables

In Version 3, when you create a new 4D file (4D table, in the V6 terminology) in the Structure window, 4D automatically creates one 16K Primary Address table and one 16K Secondary Address table. The Primary Address table contains the addresses of 4,096 Secondary Address tables, and each Secondary Address table can hold the addresses of up to 4,096 records. When the number of records in a 4D table exceeds 4,096, then 4D creates an additional 16K Secondary Address table to keep track of the new records. 4D continues to create secondary address tables until it reaches the maximum of 4,096 tables. This means that, you can have  $(4,096 * 4,096)$  16,777,216 records per 4D table; that is the theoretical limit.

Version 6 will not create any Secondary Address tables unless necessary. In other words, if a 4D table has less than 4,096 records, it will only have a Primary Address table and no Secondary Address tables. When the number of records goes beyond 4,096, then 4D creates a new Primary Address table, and one additional Secondary Address table, for a total of 64K additional space used. After that, 4D will continue creating 32K Secondary Address tables until it reaches the maximum of 4,096 tables. So you also can have up to 16,777,216 records per 4D table.

## Understanding the Address Table

Let's assume that you already have more than 4,096 records in the 4D table. This means that you have at least two address tables: one Primary Address table and one or more Secondary Address tables (v3), or, one Primary Address table and two or more Secondary Address tables (v6). In the Secondary Address tables, each entry contains the location, within the data segment(s), of a record. That location includes:

- ✖ The physical address of the first block of the record
- ✖ Checksum information that allows faster integrity checking (v6 only)

The position of a record in the address table is known as the record number. This number can range from zero up to position 16,777,215, depending on how many entries are in the address table. If you know the record number of a record, you can go directly to that record by using the `GOTO RECORD` command. When you use this command, 4D goes directly to that entry in the Address table, reads the record's address, and jumps directly to that address.

### Optimization Tip:

The `GOTO RECORD` command is the fastest way to randomly access a record in 4th Dimension. For the fastest possible access to records in a selection, use `SELECTIONTOARRAY([aTable])` to create an array of record numbers. Then, instead of doing a query for a record, you can use the `GOTO RECORD` command.

**WARNING:** Be aware that the address table entries of deleted records can be reused by new records added to the 4D table, so you can use this scheme if you know that record numbers are not going to change between the time you obtain them and the time and use them. In addition, new records added during a transaction do not get their final record numbers until the transaction is validated. For more information, see the Language Reference manual section [About Record Numbers](#).

## What Happens When You Save a Record

4D stores records in contiguous blocks of 128 bytes. This means that no record can be less than 128 bytes in size, and that the space occupied by a record on disk is always a multiple of 128 bytes.

When you create a new record and save it, 4D calculates the size of the record and looks for the first available contiguous blocks that are big enough to hold it. For example, if the size of the record is 300 bytes, 4D will look for the first available three contiguous blocks (128 \* 3 = 384 bytes). If you have any Text, Picture, or BLOB fields, then 4D will store that data in a separate place, with a pointer to that data stored within the primary blocks of the record.

## **What Happens When a User Modifies a Record**

---

When a user modifies the record, 4D takes the following steps:

1. Updates the 4D Backup Log File, if 4D Backup is running.
2. Checks the record map against the table's Master Structure Map. If the map has changed since the last time the record was saved, 4D will update the record map with the new field types and the new offsets.
3. Saves the record back to disk, into the same space if possible. For example, if the record size was 300 bytes, it was using 3 blocks for a total of 384 bytes. If the new version of the record is 384 bytes or less, then 4D will save the modified record in the same location. If the modified record size is greater than 384 bytes, then 4D will have to find a new location with enough contiguous blocks to store this record. It will then mark the old space as unused.
4. Updates the Record Address table, if necessary.
5. Updates the Indexes associated with this logical file, to reflect the new values in this record.

## **What Happens When a User Deletes a Record**

---

When a user deletes a record, the responses by 4D will vary, depending on the settings chosen for this table in the Table Properties dialog.

Table Properties

Privileges Triggers Color

Table name  
Table1

Triggers

- On saving new record
- On saving an existing record
- On deleting a record
- On loading a record

Attributes

- Invisible
- Completely Deleted

Done Apply

If the Completely Deleted option is selected, then 4D will do the following every time a user deletes a record:

1. Updates the 4D Backup Log File, if 4D Backup is running.
2. Updates the Address table entry to show that the record is deleted.
3. Updates Index table entries to show that the record is deleted.
4. Marks the Tag in the record header to show that the record is deleted.

If the Completely Deleted option is turned off for this table, then 4D will only:

1. Update the 4D Backup Log File, if 4D Backup is running.
2. Update the Address Table entry to show that the record is deleted; and
3. Update Index Table entries to show that the record is deleted.

This means that turning off the Completed Deleted option will improve performance. However, if you later Recover by Tags, there is a possibility that some deleted records will be recovered too, because the Tags in the record headers were not updated.

If the space that the deleted record occupied has already been overwritten, then there is no danger that the record will be recovered in a Recover by Tags. However, if the space has not been overwritten, then there is a possibility that you will recover the deleted record. So, you must evaluate the trade-off between improved performance (Completely Deleted is turned off) and data integrity (Completely Deleted is selected).

## **Summary**

---

In Part 1 of this technical note, you have learned more about the 4D data file and what happens within the data file when a user saves a new record, modifies a record, or deletes a record. You are now ready to learn, in Part 2, some principles that will help you to optimize the access to your large 4D data files.

## **See Also**

Optimizing Record Selections - Part 2.