

SQL Database Definition

The exporting of data from Visual3D requires an external database with a predefined data structure.

There are no restrictions as to which database system is used since connection to it is provided by ODBC connections. C-Motion's sample code is provided using Visual Basic for Application in Microsoft's Access database system.

The data structure is normalized to eliminate redundant data entry. The specific data values for signals, metrics, and analog data, however, are stored in a large single string that must be parsed prior to using the data for analysis purposes. This approach was necessary due to the inherent speed limitations a relational database has with storing tremendous amounts of data.

Note that some RDBMS systems may need add-on modules or extensions to support Transforms or Pivot Table capabilities. MS Access has the functionality built-in, so some example code provided by C-Motion may not transfer directly to other systems.)

The picture below illustrates the how the Visual3D signal and event tree structure maps to database tables. The Patient Info table is provided as a sample illustration on where the relationship to patient information may be managed. The BLOB table is where the raw data is stored after the exporting process is done, and a Raw Data table is provided for the parsed results. Sample database definitions and sample routines for parsing and managing data are provided by C-Motion.

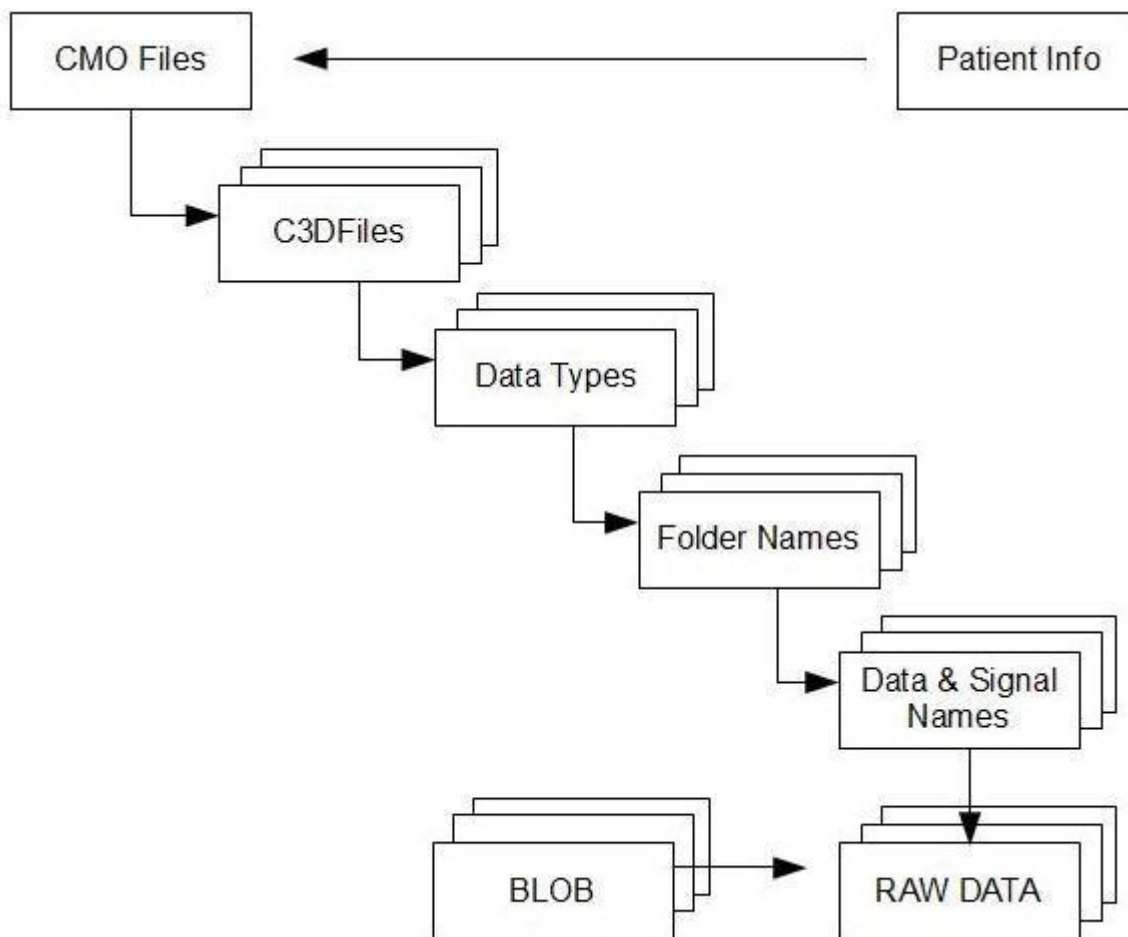


Table Structure Definition

The actual SQL database tables loosely follow the Visual3D data tree layout, and are defined below:

TABLE NAME	DESCRIPTION	COLUMNS
L0_cmo	Contains the CMO file names.	cmoID = Integer, auto-incrementing index cmoName = Text PatientID = link to a patient table (optional)
L1_c3dfile	The C3D files contained in each CMO file.	fileID = Integer, auto-incrementing index cmoID = link to L0_cmo (foreign key) fileName = Text
L2_type	The signal or data TYPE in each C3D file. (Examples are: TARGET, EVENT, ANALOG, LINK_MODEL_BASED, etc.)	typeID = Integer, auto-incrementing index fileID = link to L1_c3dfile type_Name = Text (beware the "type" or "typename" may be reserved words in some RDBMS programs)


```
        For x = 1 To UBound(frameparts) Step 2
            SQLstr = "INSERT into l5_framedata "
            SQLstr = SQLstr & " (dnameID, framenum, label, dataValue) VALUES "
            SQLstr = SQLstr & "(" & recset!dnameID & ", " &
frameparts(0)
            SQLstr = SQLstr & ", '" & frameparts(x) & "', " & frameparts(x+1)
            SQLstr = SQLstr & ");"
            DoCmd.RunSQL SQLstr
        Next
    Next
    recset.MoveNext
Loop
```

Note that this routine could potentially take a very long time to run since a SQL INSERT is performed for every component of every frame of data.

However, once parsed and inserted into the L5_framedata table, it is very easy and fast to access the data for analysis. This is where pivot tables play an important role. They permit the dynamic storage of data components to be managed since it is impossible to accommodate a data structure for every potential digital and analog motion capture related signal generator.

However, pivot tables can be avoided if only certain information, in a consistent format is needed, then new tables can be created to improve the speed of the parsing process.

From:
<https://has-motion.com/wiki/> - Wiki Documentation Site

Permanent link:
https://has-motion.com/wiki/doku.php?id=visual3d:documentation:definitions:file_formats:sql_database_definition&rev=1721231108

Last update: 2024/07/17 15:45

