## **Diving into Parquet**

written by DaltonRuer | January 31, 2025



## Diving into Parquet: It's Not Just About the File Format

I'm finally diving into something I've been hinting at for a while: Diving into parquet files. We're going to look at using parquet both internally and externally for storage because:

- As I shared in my <u>Qlik Community Post Making Sense of</u> <u>QVD Files</u>, it's not about the file format, it's about the concept of reusing data.
- I want to prove that you can share the intellectual property in your applications, regardless of format, for reporting and other uses.

While I will be describing the progression with a few screenshots embedded, the video at the bottom is your way to actually see everything in action. I know this is knew and I didn't want you think I was faking anything with manipulated images.

## Setting the Stage: Data and Metadata

Let's get oriented. In my example, I have a simple data model: patients and encounters.

~	🗘 DaltonTest Store Metadata into QVD 🛛 …
	S History S Data sources
1	Patients:

2	Load * Inline [
3	PatientID, Name, Age
4	1, Dalton Ruer, 60
5	2, Qlik Dork, 99
6	3, Dan Wizard Pilla, 999
7	4, Levi TheMan Turner, 999
8	];
9	
10	Encounters:
11	Load * Inline [
12	EncounterID, PatientID, Arrival, Discharge
13	101, 1, 12/2/1999 1:42 PM, 12/3/1999 2:17 AM
14	102, 2, 12/4/2012 3:17 AM, 12/5/2012 1:08 PM
15	103, 5, 12/17/2023 9:01 PM, 12/17/2023 10:12 PM
16	104, 1, 4/12/2003 7:16 PM, 4/14/2003 9:04 AM
17	105, 2, 7/4/2013 8:27 AM, 7/4/2013 10:15 PM
18	];
19	
20	TableCommentMap:
21	Mapping Load * Inline [
22	Table, Table Comment
23	Patients, Patients or those who have sought a consultation in our health system'
24	Encounters, Encounters can be admissions, visits or consultations with our system at the main hospitals or ambulatory locations
25	
26	
27	FieldCommentMap:
28	Mapping Load * Inline
29	Field Comment
30	PatientID, Unique Identifier for our Patients table which is assigned by the Mountain Dew Fountain System
31	Name, the full name for our patients. Not all patients contain a middle name.
32	Age, the age of our patients in terms of wiscom exhibited based on our Al systems recognition of their cognitive abilities. May not be their actual
33	Encounteriu, unique laentifier for our Encounters table which is assigned in our Code Red EHR System
34	Arrival, the time our at based factal scanning system registered the patient coming in the siloing glass doors
35	Discharge, The time our AL based tactal scanning system registered the patient warking out our silding glass doors
30	l i

Figure 1: Screenshot of inline data load script to understand the data values and metadata being used

I have four patients and five encounters. A couple of my patients, Dan The Wizard and Levi the man Turner, are so healthy they have no encounters at all. As I've previously discussed with you about <u>metadata</u>, our AI engine interprets their "cognitive wisdom" as 999 years old. This highlights the importance of metadata, which helps me understand my data fields correctly. Without it I might think I have really bad data. In my storing application (Figure 1), I've set up comments, table maps, field maps, and tag maps. This metadata travels with QVD files when the data is stored. When I read the data back (Figure 2), I can see my table comments, field comments, and tags.

Patients		Encounters
PatientID @	•	PatientID 🔑
Namo		EncounterID
Ago		Arrival
Age		Discharge

Density	100%	*	Patients		
Subset ratio	100%	÷.	PatientID	Name	Age
Has duplicates	true		1	Dalton Ruer	60
Total distinct values	3		2	Qlik Dork	99
Present distinct values	3		3	Dan Wizard Pilla	999
Non-null values	4		4	Levi TheMan Turner	999
Tags	Snumeric Sinteger %AI %Dimension				
Comment	The age of our patients in terms of wisdom exhibited based on our AI systems recognition of their cognitive abilities. May not be their actual biological age.	Ŧ			

Figure 2: Data Model View of the simple inline data we created in figure 1.

# Switching Gears: From QVD to Parquet

Now, here's where it gets interesting. I'm going to switch from QVD to parquet format. I'm still using the same data-patients, encounters, table maps, comment maps, and field tag maps. But this time, instead of storing to a qvd file, I'm going to store to parquet, using the DataFiles, in the Dalton space. I'm creating files named Patients.parquet and Encounters.parquet. When I store to parquet format, my data model in the storing application remains the same, as you'd expect, with all the metadata intact so nothing to see there.

56 Store Encounters into [lib://DaltonTest:DataFiles/encounters.parquet] (parquet);

<sup>54 🛛 // 🧩 🧩 🦗</sup> We will never ever never need the data in QVD format always going to use Parquet down the road internally

<sup>55</sup> Store Patients into [lib://DaltonTest:DataFiles/patients.parquet] (parquet);

Figure 3: You can store to parquet just like you do with QVD by changing the extension and the format to parquet

When loading the data from parquet files instead of QVD, I still get my four rows of patients and five encounters. The data is all there. However, if I check my data model viewer, I have a problem: I've lost all the metadata.

```
1
     Patients:
 2
    LOAD
 3
        PatientID,
 4
       Name,
 5
        "Age"
    FROM [lib://DaltonTest:DataFiles/patients.parquet] (parquet);
 6
 7
 8
9
     Encounters:
10 LOAD
11
        EncounterID,
       PatientID,
12
13
       Arrival,
14
      Discharge
15 FROM [lib://DaltonTest:DataFiles/encounters.parquet] (parquet);
16
```

Figure 4: To read from Parquet you simply change the file extension you are reading from and the format

Patients		Encounters	
PatientID 🖉	•	PatientID 🖉	
Name		EncounterID	
Age		Arrival	
		Discharge	

Age		Patients				
Density	100%	PatientID	Name	Age		
Subset ratio	100%	1	Dalton Ruer	60		
Has duplicates	true	2	Qlik Dork	99		
Total distinct values	3	3	Dan Wizard Pilla	999		
Present distinct values	3	4	Levi TheMan Turner	999		
Non-null values	4					
Tags	\$numeric \$integer					

Figure 5: Data Model View of application that read from Parquet shows all of the data, but the metadata is missing

Parquet simply doesn't retain the metadata. Therefore, if you are using files internally and need to keep your metadata, sticking with a QVD file format is likely your best option. Metadata aside, if I am just looking at the visuals, they remain the same whether I read from .qvd or .parquet files. The end user doesn't need to know, or care about, the underlying architecture.

□□□ Good time to notice that Dan Wizard Pilla and Levi TheMan Turner have no encounters, and that EncounterID 103 has an associated PatientID that doesn't actually exist. Keep note in your head of this as it will be important soon. Now back to post. □□□

								🖒 DaltonTest		Read	from QVD ····			
🗆 Assets  🥺	Insight Advisor	[C] [2] [2] [0] No selection												
My new s	sheet													
Q PatientID		Q EncounterID												
				PatientID (	2	EncounterID	Q	Name Q	Age	Q	Arrival	Q	Discharge	Q
	1	1	01		1		101	Dalton Ruer		60	12/2/1999 1:42 PM		12/3/1999 2:17 AM	
	2	1	02		1		104	Dalton Ruer		60	4/12/2003 7:16 PM		4/14/2003 9:04 AM	
	3	1	13		2		102	Qlik Dork		99	12/4/2012 3:17 AM		12/5/2012 1:08 PM	
					2		105	Qlik Dork		99	7/4/2013 8:27 AM		7/4/2013 10:15 PM	
	4	1	)4		3 -			Dan Wizard Pilla		999	-		-	
	5	1	05		4 -			Levi TheMan Turner		999	-		-	
					5		103	-	-		12/17/2023 9:01 PM		12/17/2023 10:12 PM	

Figure 6: Screenshot of a some filter panels and table object showing the data read in from parquet tables

## Taking It External: Storing Parquet in S3

While end users don't care about architecture, your boss' boss just may. He might insist on not retaining so much IP inside of proprietary Qlik QVD files. I just showed you could do that. Yay.

But you know that boss' bosses can be bossy and their next edict is that it has to be shareable outside of your Qlik environment. Don't panic that's easily doable.

Let's go ahead and store those Parquet files into an Amazon S3 bucket, rather than our data files library. Agility is the whole purpose of the libraries to begin with. I can't help it if you were lazy and just always picked the default "DataFiles" library.

I'm using the same four patients, five encounters, and mappings. This time, I'm storing to an Amazon S3 bucket (Figure 7).

58 // \* \* But need to store to parquet for external usage 59 Store Patients, Encounters into [lib://Amazon\_S3\_V2/HealthMart.parquet] (parquet); 60

Figure 7: Storing both the Patients and Encounters table into a single parquet file called HealthMart

Did you actually read the image or did you just scroll your eyes over it. Because if you read it, and understood the implications in it ... your mind would be blown. In my previous videos and my Community Post: <u>Making Sense of</u> <u>QVD files</u>, I highlighted over and over that QVD files were "Qlik Virtual Data warehouse tables." They are 1 for 1 for internal tables. But wowza, when you store to Parquet there is no 1 for 1 table limit. As it might be your "Gold/Mart" layer when you store it ... the cool thing is it can actually be a "MART" that contains all of the needed tables.

Just like a Data Mart, it has 1 center and is most likely a Star schema. If you want to dive into this, the first file must be the center of that star or snowflake in your Qlik data model. If I have other data islands for example, I can not write those tables into the same parquet "mart." Notice in Figure 7, that rather than naming the file like I normally would with a table name ... I actually called my "HealthMart." Maybe, just maybe ... your boss' boss edict to externalize was

actually a good thing for you. Sure you were ticked off at first, but now you just might feel like a superstar. Just like Qlik Data Product rolling up the entire data model for easy end user consumption ... you've encapsulated all of the tables into 1 parquet file.

My loading app is going to change to read externally from the Amazon S3 bucket and when I choose to read the "HealthMart.parquet" file (Figure 8) I see something interesting, and perhaps confusing.

Notice that I don't see a Patients table or an Encounters table. Instead I see a "HealthMart" table that sure looks like my Patient data. And another table called PatientID:Encounters. Even more interesting perhaps is the surrogate that got created "%Key\_PatientID:Encounters." Are those good things?

k Analytics app Dat	a load editor 🗸	🔿 DaltonTest 🛛 🧱 F	Read from QVD ····	
Select data from Healthl Tables Q Filter tables HealthMart	Mart.parquet → File format Apache Parquet Fields			
PatientID:Encounte	4 PatientID	Vame	✓ Age	✓ %Key_PatientID:Encounters
	1	Dalton Ruer	60	0
	2	Qlik Dork	99	1
	3	Dan Wizard Pilla	999	2
	4	Levi TheMan Turner	999	3

Figure 8: Load select from our Parquet mart, not table, file showing that much like an Excel load with multiple worksheets you can choose which tables in the mart you want. Also shows that the tables aren't the same table names we stored them as.

#### Mart vs Table

When we stored the data "as a mart" not as a "table" Qlik chooses to use the name we give the file for that center spoke. We called the file "HealthMart.Parquet" and it said "rock on Qlik Dork I will refer to your data mart as "HealthMart." That's a great thing.

Understanding the other ... is a bit more complex. But if I do the job right, you will understand it in no time. Within Qlik's Associative Engine we have no problem at all with the fact that we have 2 patients with no encounters, and an encounter that had no patient. But If we are "externalizing" our data beyond Qlik's walls ... then we can no longer think with an Associative Engine mindset. We have to flatten the data out, the way a SQL VIEW would look. Within Qlik you don't have to think about Inner, Outer, Left, Right, Upside down or inside out joins. But in a Data Mart world you do again.

So, let's take a quick look, just for giggles and grins at the Encounters "table" that is in our mart in Figure 9. A few things you need to notice: There are only 4 valid encounters listed, what happened to 103 and why are there 2 blank encounters but values for that weird surrogate key?

k Analytics app Data	oad editor 🗸	🗘 DaltonTest	Read from QVD ····	
Select data from HealthM. Tables	art.parquet File format Apache Parquet ▼ Fields			
PatientID:Encount 4	EncounterID	Arrival	✓ Discharge	%Key_PatientID:Encounters
	101	12/2/1999 1:42 PM	12/3/1999 2:17 AM	0
	104	4/12/2003 7:16 PM	4/14/2003 9:04 AM	0
	102	12/4/2012 3:17 AM	12/5/2012 1:08 PM	1
	105	7/4/2013 8:27 AM	7/4/2013 10:15 PM	1
				2
				3

Figure 9: Load Select for the Encounters table data read from our parquet mart.

Most Data Mart generation systems create the surrogate key to utilize to join facts/dimensions not knowing if you might perhaps change the value. Maybe Dalton Ruer with PatientID 1 today, becomes 000001 tomorrow. Using this surrogate key value instead allows the relationships in the marts to keep on ticking without having to cascade changes to tons of tables

down the road. Maybe we wouldn't want anyone to see the actual PatientID field because it was PII. Well that surrogate key is sure handy dandy. Thanks for elevating our game Qlik. The guestion we are left with then is "why do we have 2 blank encounter rows with Dan and Levi's patient surrogate key?" Since Patients was the center of the star, we MUST RETAIN all of it's information. Qlik has flattened out that in-memory Associative Engine version of Encounters into more of a SQL view, and like a Left Join with the Patients information. We end with blank encounters the up 2 but %Key PatientID:Encounters surrogate value. And just like a Left Join we do not return Encounter 103. Aha. Now I remember why I fell in love with the Associative Engine 15 years ... I hated dealing with the symptoms of joins when the underlying data wasn't perfect and had referential integrity. Almost like I created that inline data many posts earlier for this very reason.∏

#### Swapping Left and Right

I hear you Mr./Mrs. Smarty Pants. Yes I did change my mart storage to store Encounters and then Patients. And you are 100% correct ... I do get to see all of my encounters.

PatientID:/Patients	s Filter tables HealthMartEncoun	Fields				Q, Filter fields
101         1         1/2/1999 1/42 PM         1/2/1999 2:17 AM         0           102         2         1/2//2012 3:17 AM         1/2//2012 1:08 PM         1           103         5         1/2/17/2023 9:01 PM         1/2//7/2023 1:01 2 PM         2           104         1         4/1/2/003 1:01 FPM         3	PatientID:Patients	3 EncounterID	PatientID	Arrival	✓ Discharge	%Key_PatientID:Patients
102         2         12/4/2012 3:17 AM         12/5/2012 1:08 PM         1           103         5         12/17/2023 9:01 PM         12/17/2023 1:01:2 PM         2           104         1         4/12/2003 7:16 PM         4/14/2003 9:04 AM         3		101	1	12/2/1999 1:42 PM	12/3/1999 2:17 AM	0
103         5         12/17/2023 9:01 PM         12/17/2023 10:12 PM         2           104         1         4/12/2003 7:16 PM         4/14/2003 9:04 AM         3		102	2	12/4/2012 3:17 AM	12/5/2012 1:08 PM	1
104 1 4/12/2003 9:04 AM 3		103	5	12/17/2023 9:01 PM	12/17/2023 10:12 PM	2
		104	1	4/12/2003 7:16 PM	4/14/2003 9:04 AM	3
105 2 7/4/2013 8:27 AM 7/4/2013 10:15 PM 4		105	2	7/4/2013 8:27 AM	7/4/2013 10:15 PM	4

Figure 10: Screenshot of load select where I created the parquet mart with encounters first and patients second

Then Encounters becomes the LEFT side of a LEFT JOIN ... and we end up with an entirely different issue with our Patients table. Dan and Levi, the stars of my AI Cognitive Wisdom Age calculation to demonstrate the importance of metadata are entirely gone from my system. Like I said ... I hated dealing with the symptoms of joins when the underlying data wasn't perfect and had referential integrity problems like I introduced. Be sure and consider this and other limitations of storing to Parquet files in general. Especially keep this in mind if you are going to try and create a Parquet MART.

I entitled the post Diving into Parquet and we just dove. Depending on your data it might look like a reverse 4.5 somersault in the pike position with no splash on entry, or it could look like a belly flop. I merely wanted to point out that it is an option. Within Qlik Talend Cloud you would likely use our Data Product to surface a wonderful mart of all of the required tables. When sending data externally this is an option you might consider, as long as you understand the risks. If you have integrity issues, then I strongly suggest sticking with pure table based parquet files. That way all your data will travel forward.

Select data from HealthMartEncounters.parquet									
Tables	$\rightarrow$	File format							
C Eiltertebler		Apache Parquet 🔻							
C Filter tables		Fields							
HealthMartEncount	5	T I I I I I I I I I I I I I I I I I I I							
PatientID:Patients	3	✓ Name	Age	Key_PatientID:Patients					
		Dalton Ruer	60	0					
		Qlik Dork	99	1					
				2					
		Dalton Ruer	60	3					
		Qlik Dork	99	4					

Figure 11: Screenshot of the load select focusing on the loss of patients if we stored encounters into our parquet mart first

 $\Box\Box$  If you got excited about the idea of having all of the parquet tables in one place as a data mart, but really want to ensure all of your data for all of your tables is maintained; Don't give up hope as there just might be an iceberg in the waters ahead that can make you look like a hero.  $\Box\Box$ 

### The Metadata Challenge

Let's put the referential integrity issues aside, since I forced them to make a point. We need to get back to the fact that we lost our metadata when saving to table based or mart based Parquet files. I really REALLY **REALLY** think it's important that we keep it. If you need a refresher on the

importance of metadata, please take the time to read my previous post that I've shared multiple times entitled: The Importance of Metada.

If you agree with me that anyone, including people external to Qlik should know that the Age field isn't biological then we need to solve this problem. Right?

In the previous examples the metadata was created simply using MAPPING LOADS. That's in memory only and can't be shared. But nobody will think less of us if we make our Metadata a "real table." Because "real tables" can be stored. If I alternatively switch to a real table that contains all of the information my code would look like this

20 Health\_Metadata: Load \* Inline [ Table, Table Comment, Field, Field Comment, Field Tag Patients, Patients or those who have sought a consultation in our health system Encounters,Encounters can be admissions or visits or consultations with our system at the main hospitals or ambulatory locations Patients, PatientID, 'Unique Identifier for our Patients table which is assigned by the Mountain Dew Fountain System, %Dimension Consustent Device Table and the average of the system at the Mountain Dew Fountain System for an event of Constraints of the Mountain Dew Fountain System for a syst 21 24 Encounters, , PatientID, 'Unique Identifier for our Patients table which is assigned by the Mountain Dew Fountain System, %Dimension Patientd, , Name, The full name for our patients. Not all patients contain a middle name., %Dimension 26 Patients, , Age, The age of our patients. Not all patients contain a mitude name, summission Patients, , Age, The age of our patients in terms of wisdom exhibited based on our AI systems recognition of their cognitive abilities. May not be Encounters, , EncounterID, Unique Identifier for our Encounters table which is assigned in our Code Red EHR system, %Dimension Encounters, , Arrival, The time our AI based facial scanning system registered the patient coming in the sliding glass doors, "%AI %Dimension" Encounters, , Discharge, The time our AI based facial scanning system registered the patient walking out our sliding glass doors, %AI 28 29 30 31 32 33 1; // Create mapping tables that will map the comments and tags to tables and fields 35 TableCommentMap: Mapping Load 37 Distinct Table, [Table Comment] Resident Health\_Metadata; 39 FieldCommentMap: 40 41 Mapping Load Distinct Field, [Field Comment] 42 43 44 Resident Health\_Metadata; 45 FieldTagMap: 46 Mapping Load Distinct Field, [Field Tag] 47 48 Resident Health Metadata; // Map the comments and tags now to the tables and fields Comment Tables using TableCommentMap; Comment Fields using FieldCommentMap; 50 Tag Fields using FieldTagMap; 53

Figure 12 Screenshot of inline load script using a physical table store to contain all our metadata that is then read mapping table to apply the metadata to tables and fields

#### Solved

I have all the same tagging "maps" that I needed, but now I can literally just store the metadata table itself into a Parquet file as well. So, external users can see it, just like they see .ReadMe or .MD files etc.



Figure 12: Screenshot of store statements saving the tables to a parquet mart file, and also storing the metdata for the entire mart to a parquet file as well

Notice my last step it is to simply Drop the Health\_Metadata table because I don't want a weird data island in the model. But as a challenge ... feel free to comment on any thoughts or ways you can see yourself actually using that Metadata Island inside of an application? As you think of ideas ... suddenly you will be loving METADATA as much as I do.

	Patients PatientID & Name Age		Encounters PatientiD p EncounteriD Arrival Discharge	Her Tai Fic Fic	eld Comment eld Comment eld Tag				
▼ Preview									
Health_Metadata	lth_Metadata Health_Metadata								
Rows 9	Table	Table Comment						Field	Field Comment
Fields 5	Patients	Patients or those who ha	itients or those who have sought a consultation in our health system						
Tags Sascii Stext	Sascii Stext Encounters Encounters can be admissions or visits or consultations with our system at the main hospitals or ambulatory locations			atory locations					
	Patients						PatientID	'Unique Identifier for our Patients table which is assigned by the Mountain Dew Fountain Sy:	
	Encounters							PatientID	'Unique Identifier for our Patients table which is assigned by the Mountain Dew Fountain Sy:
	Patientd							Name	The full name for our patients. Not all patients contain a middle name.
	Patients							Age	The age of our patients in terms of wisdom exhibited based on our AI systems recognition of
	Encounters							EncounterID	Unique Identifier for our Encounters table which is assigned in our Code Red EHR system
									Later the second of the second s

Figure 13: Screenshot of the Data Model view showing our Metadata table

## Reading the Metadata

In my reading application all I have to do is implement the same basic code structure. Instead of an INLINE statement to build the metadata, I literally just read it from the file. Then I still go through and create the maps and then comment/tag the Tables and Fields. That's just to easy.

```
19
     Health_Metadata:
20
     LOAD
21
         "Table",
         "Table Comment",
22
23
         "Field",
24
         "Field Comment",
25
         "Field Tag"
26
     FROM [lib://Amazon_S3_V2/HealthMart_metadata.parquet] (parquet);
27
28
     // We want to split out into maps any metadata we have
29
     // Create mapping tables that will map the comments and tags to tables and fields
     TableCommentMap:
30
31
     Mapping Load
32
     Distinct Table, [Table Comment]
33
     Resident Health Metadata;
34
35
     FieldCommentMap:
36
     Mapping Load
37
     Distinct Field, [Field Comment]
     Resident Health Metadata;
38
39
40
     FieldTagMap:
41
     Mapping Load
42
     Distinct Field, [Field Tag]
43
     Resident Health_Metadata;
44
45
     // Map the comments and tags now to the tables and fields
46
     Comment Tables using TableCommentMap;
47
     Comment Fields using FieldCommentMap;
     Tag Fields using FieldTagMap;
48
```

Figure 14: Screenshot of the load script for an application that reads the metadata

## Video

I realize it's hard to really grasp these very new concepts by just reading and seeing images. To that end please feel free to listen to me while you see the loading actually occuring.

### Bonus

Just because I appreciate your actually reading the entire post I wanted to throw in a little bonus. I wanted you to see that not only does it all work with external S3 storage and Parquet ... you can track the lineage of it as well. How cool is that? Now the Dork in me wants to have 2 different storage applications to see that the two tables in the healthmart file come from different locations.

