PharmaSUG China 2023 - Paper DS-159

Improving CDISC Basic Data Structure Requirements on PARAMCD for ADaM Automation

Yu Bo*, Zhiheng Chen, and Kang Zhao*, AstraZeneca

ABSTRACT

The ADaM Basic Data Structure is the dominant data structure of ADaM and deserves special attention because of its use of value level metadata. During the development of an ADaM automation product at AstraZeneca, there are gaps found between the automation requirements and CDISC standards. The metadata-driven automation process is explained and the authentic form of parameterized variable notation in the ADaM specifications is introduced in this paper. Furthermore, the concept of primary keys in the automation process is discussed and reemphasized.

To enable universal references to parameterized variables to achieve automated processes, we create two notation forms and suggest PARAMCD to be the sole primary key variable for value-level metadata. It is possible that there are more complex use cases where the proposal may not be directly applied. However, we hope to encourage wider discussions on notation forms to facilitate clinical data transformations.

INTRODUCTION

Data Transformation Team at AstraZeneca is tackling the challenge of ADaM automation.

With the development of automatic tools for data generation, more complicated points must be considered for data structure.

CDISC ADAM DATA STRUCTURES

The three standard data structures defined in ADaM v2.1 and ADaM OCCDS v1.0 are the Subject-Level Analysis Dataset (ADSL) Structure, the Basic Data Structure (BDS), and the Occurrence Data Structure (OCCDS). Relatively speaking, it is simpler and more straightforward to generate ADSL and OCCDS datasets than BDS datasets since there is no value level metadata for ADSL and OCCDS, which can be derived variable-by-variable with minimal effort.

BASIC DATA STRUCTURE (BDS)

A BDS serves as a powerful container of clinical analysis data which can be reorganized or summarized under the structure of one or more records per subject, per parameter, per timepoint (if applicable). Whether direct clinical test results or derived values for complex analysis can fit naturally in this structure given the great flexibility of BDS.

The ADaMIG specified a set of standard variables that can be used in a BDS dataset. Despite the identifier and timing variables, PARAM & PARAMCD are the two most important key variables within a BDS.

METADATA-ONLY AUTOMATION

Data specification is crucial for generating ADaM datasets, as it contains all the information required to create the variables, such as their names, types, derivations, and so on. To ensure the quality of the specification, the derivations should be as clear and precise as possible, so that each variable can be derived without ambiguity.

SIMPLE CASES

As an example, the derivation for ADSL.EOSDY (Study Day of End of Study) is:

* These authors contributed equally to this work

```
Set to End of Study Date ADSL.EOSDT
- Date of Randomization ADSL.RANDDT
+ 1 day
```

CDISC ADaM and ADaMIG provide naming conventions for variables (CDISC 2009). References to variables are assumed to be the same. In simple cases like the above, they are clear and sufficient.

NOTATIONS

Parameterized variables complicate variable references. For example, ADTR.AVAL is a parameterized variable of which the derivation depends on the value of another variable, namely ADTR.PARAMCD.

Despite many possible descriptive texts, its reference in essence is:

```
ADTR.AVAL where ADTR.PARAMCD = 'TRLONDD'                    (a)
```

It gets more complex with more conditions:

```
Observations in ADTR
    where ADTR.AVAL where ADTR.PARAMCD = 'TRLONDD' >= 0
    and ADTR.TRINTRFL not equals 'Y'                        (b)
```

And it is getting worse and worse:

```
ADTR.AVAL where ADTR.PARAMCD = 'TRLONDD' and ADSL.FASFL = 'Y'    (c1)
```

or undistinguishably:

```
ADTR.AVAL where ADSL.FASFL = 'Y' and ADTR.PARAMCD = 'TRLONDD'    (c2)
```

or worse:

```
ADTR.AVAL where ADTR.PARAMCD = 'TRLONDD' where ADSL.FASFL = 'Y'  (c3)
```

To normalize, we propose a new notation in the form of

```
DATASET.PARAMCD=CDVAL>VARIABLE                              (N1)
```

With it, the examples above can be rewritten as:

```
ADTR.PARAMCD=TRLONDD>AVAL                                   (a')
```

```
Observations in ADTR
    where ADTR.PARAMCD=TRLONDD>AVAL >= 0
    and ADTR.TRINTRFL not equals 'Y'                        (b')
```

```
ADTR.PARAMCD=TRLONDD>AVAL where ADSL.FASFL = 'Y'            (c1')
```

```
ADTR.PARAMCD=TRLONDD>AVAL where ADSL.FASFL = 'Y'            (c2')
```

```
ADTR.PARAMCD=TRLONDD>AVAL where ADSL.FASFL = 'Y'            (c3')
```

Note that **(c1')**, **(c2')** and **(c3')** are exactly the same, with the spec-writer fully aware that ADTR.AVAL is a parameterized variable while ADSL.FASFL is not. The notation increases such awareness and helps ease the spec-writing.

MORE ON BDS

According to ADaMIG, PARAM shall contain all information required to describe the content of AVAL/AVALC, including but not limited to units, specimen type, position, and transformation function (CDISC, 2019). Because PARAM and PARAMCD is 1-1 mapping, PARAM and PARAMCD can be used interchangeably.

However, in practice, there are occasions where PARAM/PARAMCD cannot uniquely identify records per subject per timepoint if the records are derived or imputed within the parameter. In such cases, DTYPE must be populated to indicate the algorithm or method to generate AVAL/AVALC, and thus be added to the key variable list.

In 2020, AstraZeneca metadata standards team added a variable called PARQUAL, which was proposed in the draft version of ADaMIG v1.2 though not accepted by the final ADaMIG, to the AstraZeneca ADaM standard specifications. The purpose was to distinguish the evaluators of tumor scans, the results of which would be put into oncology efficacy analysis datasets, e.g., ADRESP, ADTTE. Consequently, PARQUAL must be added to the key variable list now that PARAMCD no longer determines derivation rules for parameterized variables as specified in value-level metadata.

For example, in the Visit Response Analysis Dataset (ADRESP), we may have two records for the same PARAM/PARAMCD with different PARQUAL values – "INVESTIGATOR" vs. "INDEPENDENT ASSESSOR", which indicates the source of the parameter. This is another example that shows that PARAM cannot contain all the information to distinguish a unique record.

Table 1. Example ADRESP dataset showing ADERSP.PARAMCD is not a sufficient single primary key add-on for parameterized variable ADRESP.AVAL

USUBJID	**VISITNUM**	**PARAMCD**	**PARQUAL**	AVAL
u1	v1	p1	INVESTIGATOR	r1
u1	v1	p1	INDEPENDENT ASSESSOR	r2

ASTRAZENECA ADAM TRANSFORMATION ENGINE

INTRODUCTION

AstraZeneca is developing an automation product to generate ADaM datasets under the framework that CDISC ADaM Standard governs. The product is driven by a domain-specific language (DSL) so that the states of tabular data are unambiguously specified. With that, an interpreter consumes the specification and input datasets together to generate dataset results directly. It improves efficiency by providing fully automated execution, while more importantly assures a single-source of truth to describe the transformation process.

CHALLENGES BASED ON CURRENT PRACTICE

While developing the engine, we used it to get the efficacy datasets for an oncology study. All efficacy ADaM datasets follow BDS and were generated using both SAS and the engine. In the dataset metadata tab of the specification, we normally have a column showing the primary keys, based on which we sort and uniquely identify the observations.

A primary key is a unique identifier for each record in a data structure, such as a table or a graph. Primary keys ensure that the data is consistent, accurate and reliable, and that there are no duplicates or errors. Primary keys also enable efficient operations on the data, such as searching, sorting, filtering, and joining. Without primary keys, the data structure would be vulnerable to corruption, inconsistency, and deficient performance. Therefore, primary keys are essential for designing and maintaining a robust and scalable data structure. In practice, however, when analyzing the data, primary keys are used to indicate the analysis groups based on the demands. Such misuse of concepts compromises ADaM automation. The primary key is an important database concept. In the following context, we correct the meaning of primary key to follow its meaning in database design, which is used to meet the primary key constraints and indicates a list of variables which can uniquely define a record.

As typical specification for parameterized variables, all BDS datasets have a variable level metadata tab and a separate parameter tab that contains assigned values for parameters, e.g., PARAM, PARAMCD, and derivation methods for parameter value dependent variables such as AVAL/AVALC, ADT, etc. The structure of the parameter tab is depicted in Table 2.

Table 2 Structure of the Parameter Tab

PARAMCD	PARAM	AVAL	...more parameterized variables...
assigned value	assigned value	derivation rule	...
…			

The engine will first create a master table with one record per different combination of primary keys before PARAMCD based on the derivation rules in the variable tab, then create one sub-table for each PARAMCD based on the parameter tab.

However, CDISC standards do not force PARAMCD to be the only extension to primary keys for parameterized variables, which leads to a recursive expansion in execution path when a sub-PARAMCD primary key variable such as PARQUAL occurs.

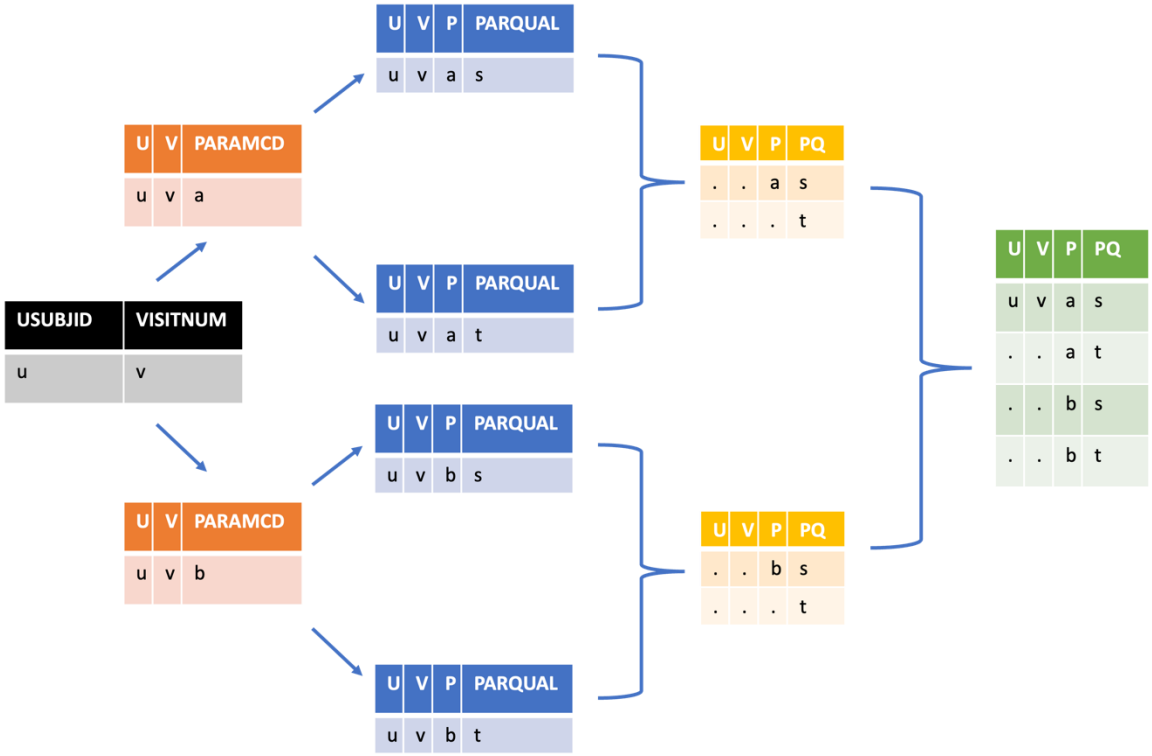

Figure 1. Illustration of execution path with sub-PARAMCD primary key variable

The derivation of PARAMCD spawns sub-tables, each for one PARAMCD value. By creating new tables instead of overwriting the original table, it makes referring to any non-parameterized variables during the derivation of parameterized variables simple, guaranteeing no redundant records, which would otherwise appear if inquiring non-parameterized variables from tables containing PARAMCD.

In Figure 1, because of the similarity of PARQUAL to PARAMCD, a nested layer of sub-tables creation-and-combination is triggered, making it memory expensive.

For example, there are five parameters in the Visit Response Analysis Dataset (ADRESP) and the primary keys specified in the specification are USUBJID, AVISIT, PARAMCD, and PARQUAL. Specifically, three of the five parameters have two different values for PARQUAL, i.e., “INVESTIGATOR” and “INDEPENDENT ASSESSOR”, and the rest two parameters have missing value for PARQUAL. Thus, the number of observations in the parameter tab is 3*2+2 = 8, and the values of PARAMCD and PARQUAL must be combined to define the derivation of parameter value and other variables.

If we insist on using both PARAMCD and PARQUAL as composite keys, nested sub-tables under PARAMCD for each PARQUAL value must be created, complicating execution paths and compromising speed. Additionally, the reference of a parameterized variable will be DATASET.VARIABLE where DATASET.PARAMCD=CDVAL and DATASET.PARQUAL=PARQUAL_VAL. Static check of the reference using composite keys would become harder since multiple conditions linked by Boolean operators require execution to check the logic.

PROPOSED SOLUTION

We propose a solution with a simple principle to demand the uniqueness of PARAMCD values, i.e., instead of using the pair of (PARAMCD, PARQUAL) as part of the candidate key, we demand PARAMCD by itself to be sufficient in uniquely deciding derivation rules for parameterized variables. For example, we have a PARAMCD = “p1” with PARQUAL = “INVESTIGATOR” and “INDEPENDENT ASSESSOR”:

PARAMCD	**PARQUAL**	AVAL
p1	INVESTIGATOR	r1
p1	INDEPENDENT ASSESSOR	r2

With proposed approach:

PARAMCD	PARQUAL	AVAL
p1	INVESTIGATOR	r1
p1b	INDEPENDENT ASSESSOR	r2

Back to the illustration in Figure 1, by enforcing PARAMCD to be the single primary key variable add-on, the creation-and-combination behavior happens only once, hence more memory efficient. Moreover, the execution path is relieved from recursion.

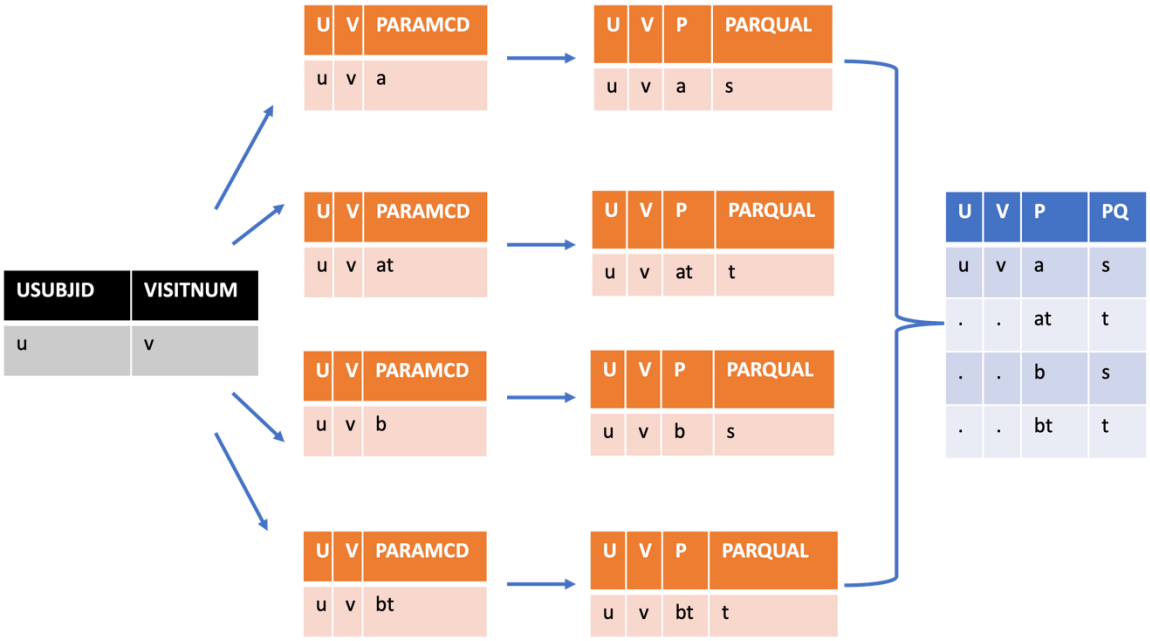

Figure 2. Illustration of execution path with PARAMCD as single add-on to primary key variables

It is worth mentioning that the new PARAMCD values "at" and "bt" in Figure 2 are just to illustrate the idea. The actual values persisted to dataset can actually be "a" and "b" as would with the approach using PARAMCD and PARUQAL pair. This is achieved by introducing parameterized variable notation of the form:

```
DATASET.PARAMCD=CDVAL                                    (N2)
```

With that, the derivation rule for **EXAMPLE.PARAMCD=at** can simply be:

```
Set to "a"                                               (d)
```

Having notations **(N1)** and **(N2)**, with the assumption that PARAMCD is a single add-on to primary key variables, all CDISC variables including parameterized ones can be uniquely and easily referenced. It makes it possible for automated processing given a variable system.

DISCUSSION AND FUTURE WORK

MORE COMLICATED CASES INVOLVING DTYPE

As mentioned earlier, DTYPE is to specify different algorithms to generate AVAL/AVALC on selected PARAMCD values. Like PARQUAL, DTYPE depends on PARAMCD values; unlike PARQUAL, however, PARAMCD values serve only as a conditional criterion and are irrelevant regarding AVAL/AVALC derivations. In other words, DTYPE values determine the derivation of parameterized variables other than PARAMCD.

Using AVAL as an example again:

Table 3. PARAMCD value determines parametrized AVAL where DTYPE is not being applied

PARAMCD	AVAL
v1	r1
v2	r2

Table 4. DTYPE associated with PARAMCD determines AVAL where DTYPE applies

PARAMCD	**DTYPE**	AVAL
v1		r1
v1	t1	**rt1**
v1	t2	rt2
v2		r2
v2	t1	**rt1**

Therefore, a possible solution is to re-use Table 3 together with a new:

Table 5. DTYPE functions as the spawning column, using target PARAMCD values as matching criteria accordingly

DTYPE	AVAL
t1 where PARAMCD = v1 or v2	**rt1**
t2 where PARAMCD = v1	**rt2**

```
DATASET.PARAMCD=v1>AVAL
DATASET.DTYPE=t1>AVAL where DATASET.PARAMCD = v1                    (N3)
```

or an equivalent and more aligned notation:

```
DATASET.PARAMCD=v1>AVAL where DATASET.DTYPE = t1                    (N3')
```

With this modification, we do not need to expand the Parameter Tab with DTYPE. For more complicated situations involving creating additional records based on imputation or simulation, we can also put the imputation or simulation method into the DTYPE tab like Table 5 to make the content in Parameter Tab clear.

This case will be verified in the engine soon and we're confident of the generalizability and flexibility of our proposal.

CONCLUSION

The use of our authentic form of parameterized variable notation has proved to be efficient and convenient in the automation process, from both the programmer's and reviewer's perspectives. We'd like to emphasize the essentialness of standardizing the metadata structure of ADaM BDS datasets in ADaM automation product development.

REFERENCES

CDISC Analysis Data Model Team (2009). Analysis Data Model, Version 2.1.

CDISC Analysis Data Model Team (2019). Analysis Data Model Implementation Guide, Version 1.2.

ACKNOWLEDGMENTS

Heartful thanks to China Biometrics Leader Team from AstraZeneca, Shanghai, China, for their supports on the project and this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Yu Bo
AstraZeneca
yu.bo@astrazeneca.com

Zhiheng Chen
AstraZeneca
zhiheng.chen@astrazeneca.com

Kang Zhao
AstraZeneca
kang.zhao@astrazeneca.com