Data Management for Cost Engineering Projects

By: Cara Cuiule, Unison

Introduction

As cost estimation data becomes more plentiful, data management becomes essential for organizing and preparing data for analysis. A data management system can help preserve integrity. It can also strengthen the traceability of the data, which is important for key stakeholders to trust analysis of the data. Even if a formal data management system is unnecessary for a project, concepts from database management can still be incorporated, which can help support future cost estimates. However, it needs to be organized in a way that makes sense to those who are using the data [1].

This paper will go over the three different types of data structures: structured data, unstructured data, and semi-structured data. First, structured data and relational databases will be reviewed. Next, semi-structured and unstructured data will be explained, followed by non-relational databases. Then, there will be an explanation on how to store large amounts of data. Existing data repositories used in cost estimation will be explored briefly. Finally, a case study of a data repository that contains structured, unstructured, and semi-structured data will be reviewed.

Data Types

Structured Data

There are three types of data that will be discussed in this paper. The first is structured data. Structured data is data that is formatted in rows and columns, such as a typical Excel spreadsheet. It has a predetermined data type [2]. It is used all the time to perform cost estimates, as many of the types of tools used to develop a cost estimate (such as multi-variate analysis or machine learning algorithms) will only take structured data as an input. Structured data is stored in relational databases.

Basic Relational Database Terminology

A relational database is a collection of tables that are joined by common keys that link the tables together. Each table represents an entity, which is the “noun” that the table is about. The columns of the tables are attributes, which describe entities. For example, a table for a cost estimation project might be an entity that describes the project level data. In this table of fabricated data, the entity would be considered the project, because each attribute (aka column) describes the project. Each row, also called a record, describes one example of that entity [3]. Another entity could be System, and a table describing the system would contain attributes that are specific to each system:

Project ID	Phase	Start Date	End Date	System ID	Submission Date
Proj_1	A	01/01/1999	01/01/2000	System_1	01/01/2010
Proj_1	B	01/01/2000	01/01/2002	System_1	01/01/2010
Proj_2	A	01/01/1999	01/01/2009	System_2	01/01/2010
Proj_2	B	01/01/1999	01/01/2009	System_3	01/01/2010

Table 1: Project Table

System ID	System Name	Platform	Organization
Sys_1	Drone Variant X	Unmanned Air	Org A
Sys_2	Drone Variant Y	Unmanned Air	Org A
Sys_3	Rotorcraft Variant B	Manned Air	Org B

Table 2: System Table

Primary keys are keys that uniquely identify every entity. So, for the first table, each project would be assigned a special key for the Project ID attribute. If the primary key is the actual name of the project, it would be considered a natural key [3]. This would be easy to use if it was guaranteed that each system would have a unique name, but that is not always the case depending on the scope of the dataset. The Viper is the name of a rotorcraft but also a type of missile, so it might not be wise to use that name for a database that includes both aircraft and missiles [4] - [5]. However, there are unique identification numbers for military systems, such as the US Tri-Service Aircraft Designation System, that assigns a unique ID to each system [6]. If the US Tri-Service Aircraft Designation System is already an attribute of the System table and a key, then it would be a natural key.

In the table above, the System ID is a synthetic key because the values (e.g., “Sys_1”) have no meaning to the user other than being a unique identifier [7]. A foreign key is a key that connects two tables together. For example, the System ID is not the primary key in the Project table, but directly refers to a unique system in the System table.

To design a relational database, an Entity Relational Diagram (ERD) can help determine how tables can be related. Each table is represented by a box, with different connections. Tables can have different types of relationships, which dictate how many foreign keys can be related to each primary key:

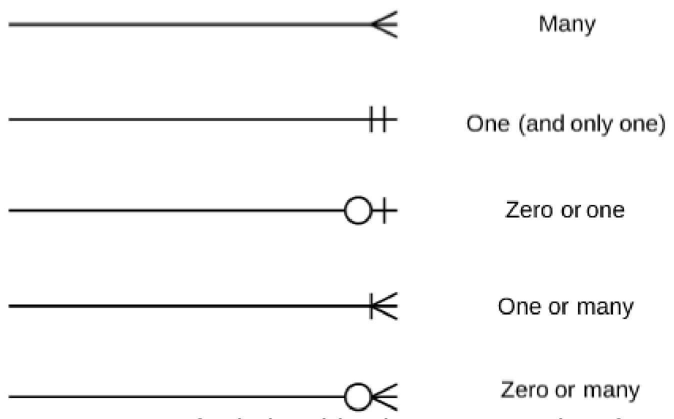


Figure 1: Types of relationships in an ERD, taken from [8]

The relational data tables above can be represented by the following ERD. Notice that a System ID can only have one record in the System table but can have many entries in the Project table. A system can also have zero entries in the Project table, such as a very new system that has no additional information yet.

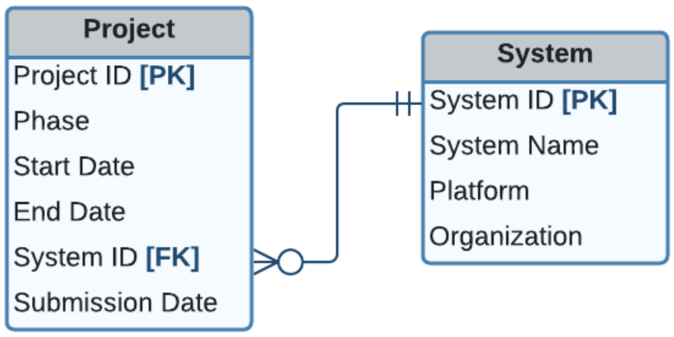


Figure 2: ERD between the Project and System tables

For a more detailed review on ERDs, review [3].

Relational Database Management System (DBMS)

Relational databases can be built in a Relational Database Management System. A list of a few popular ones can be found below. Most offer a free version:

- Microsoft SQL Server [9]
- Oracle Express [10]
- PostgreSQL (open-source database) [11]
- Microsoft Access [12]

Note: There are also programs that can be used to build relationships between tables but are not an RDBMS. One popular program is PowerBI, because relationships can be defined between tables. It is useful because it has built-in Business Intelligence capability. The desktop version of this application is not recommended for large datasets or tables with many normalization steps.

Demonstrating the steps to setting up a relational database in a RDBMS is outside the scope of this paper. For more information, see [3]. This reference also explains designs to account for physical properties of the database. For example, the number of characters designated for that attribute or restrictions on valid data values needs to be set as the database is being created.

Structured Query language (SQL) is the language used to build and maintain relational databases in RDBMs. It is a procedural language, so it is different than declarative language. SQL does not create functions; it creates tables and normalizes data with a set list of commands. It is also used to query the database. These queries can generate reports, which are specific views or outputs of the database [3].

When to use a Relational DB

It is important to remember that relational tables have a set number of rows and columns and are not easily augmented. When a table is created, the user must also designate which values can be left null (blank) or not. It is not always clear during the data gathering stages if a column value will always be available during data collection.

If there is data where the attributes from the same table vary greatly as new data is added, then it might be more appropriate to use a non-relational database. That will be discussed in a later section.

Advantages of Using a Relational Database

There are appropriate situations where small amounts of structured data can be placed in flat files in a program like Microsoft Excel. However, an Excel file can become unmanageable when there are dozens of columns at once; this can be overwhelming to maintain. Excel is limited to a little over a million rows [13]. Unlike relational databases, it is also difficult to set rules for column values, such as ensuring that the only values of a column are numerical values between 0 and 100. When more data is added, this can make data management a lot tougher for the organization or user that maintains the file.

When set up correctly, databases have security protections from the RDBMS software that assign users specific roles and permissions. Therefore, only certain users are allowed to change data. Users who want to analyze the data can query the database for the subset of the data they need. Therefore, they can manipulate it without accidently changing the original data.

A database will increase traceability due to its structure, organization, and data protection. Data types and rules for data values are set before inputting any data. The user who is managing the database can put the data in a central repository and then other users can query all or portions of the data. Users can feel confident that they are using the most authoritative version of the data at the date they performed the query. These qualities are critical for good analysis.

A properly set up relational database can also reduce the need for duplicated data, which can make data easier to edit. For example, consider a table in the report view of the Federal Logistics Information System (FLIS) database system which can be accessed through the LogiQuest website [14]. This data takes a National Item Identification Number (NIIN), which represents a replacement part, and then identifies contractors that manufacture that part. Each contractor has a specific CAGE code assigned as a numerical identifier. A small sample is below:

NIIN	CAGE	COMPANY_NAME	ADDRESS
012308556	00000	ORDNANCE CORPS	BATTLE CREEK, MI 49017
010120829	00000	ORDNANCE CORPS	BATTLE CREEK, MI 49017
012308601	00000	ORDNANCE CORPS	BATTLE CREEK, MI 49017
010875766	00000	ORDNANCE CORPS	BATTLE CREEK, MI 49017
005936718	00026	MMC INTERNATIONAL CORP.	INWOOD FINANCE, NY 11096
005312068	00026	MMC INTERNATIONAL CORP.	INWOOD FINANCE, NY 11096
012608349	00026	MMC INTERNATIONAL CORP.	INWOOD FINANCE, NY 11096
009121304	00060	HARCOSEMCO LLC	BRANFORD, CT 06405
006292043	00060	HARCOSEMCO LLC	BRANFORD, CT 06405
012861223	00062	MARINE ELECTRIC SYSTEMS, INC.	SOUTH HACKENSACK, NJ
013168990	00062	MARINE ELECTRIC SYSTEMS, INC.	SOUTH HACKENSACK, NJ

Table 3: Report table where NIINs are connected to CAGE codes and their companies.

While this data is organized and perfectly acceptable in the report view, this format is not ideal for data storage. If any Company Name, Address, or CAGE code needs to be updated, it needs to be updated for multiple rows. Also, the Company Name and Address are dependent on each CAGE code. Since there are millions of NIINs, this can get messy.

Compare this to the way the data is stored in the PUBLOG database from the Defense Logistics Agency [15]: one table that relates NIIN identifiers to CAGE code, then another that relates CAGE codes to their respective Company Names and Addresses. In this case, if an address needs to be changed, it only needs to be modified in one record:

NIIN	CAGE
012308556	00000
010120829	00000
012308601	00000
010875766	00000
005936718	00026
005312068	00026
012608349	00026
009121304	00060
006292043	00060
012861223	00062
013168990	00062

Table 4: Table comparing NIIN to CAGE codes only

CAGE	COMPANY_NAME	ADDRESS
00000	ORDNANCE CORPS	BATTLE CREEK, MI 49017
00026	MMC INTERNATIONAL CORP.	INWOOD FINANCE, NY 11096
00048	ABRAMS INSTRUMENT CORPORATION	LANSING, MI 48910-3688
00060	HARCOSEMCO LLC	BRANFORD, CT 06405
00062	MARINE ELECTRIC SYSTEMS, INC.	SOUTH HACKENSACK, NJ

Table 5: Table comparing CAGE codes to company names and addresses

A cost estimator may receive a report version of the data like Table 3. In situations where a cost estimator finds a flaw in the data, such as with one of the addresses, the data should first be changed in the database, and then another query should be run on the data to generate a new report. It is not worth changing the original report, because it is harder to change the data in multiple places and is more likely to cause an error.

Disadvantages of Using a Relational Database

There are some downsides of using relational databases. Data requirements must be set ahead of time. Columns can be added or changed but doing so can cause issues and cost a lot of effort [16]. Data security, which is **critical** for organizational success, must also be decided in the requirements phase. Experience and knowledge are required to build a database that is secure and organized. SQL is not a common skill in the cost analysis community. Many cost estimators would need other resources in their organization to help build the database.

If the data is a small and simple flat file, using a program like Excel is perfectly acceptable and encouraged. There are ways to protect the data, such as by locking the data from being edited. There are several configuration management systems that can also help as a repository for different versions of a flat file data. Most importantly, it is estimated that only 20% of data comes as structured [17]. While data that is not structured in tables can be converted to tables after being collected, this takes extra time. Organizations also prefer to store unstructured data. As mentioned before, making changes to the schema of structured data takes a lot of effort. Cost analysts and estimators typically use structured data [16]. The other two types of data, unstructured and semi-structured data, will be reviewed in the next sections.

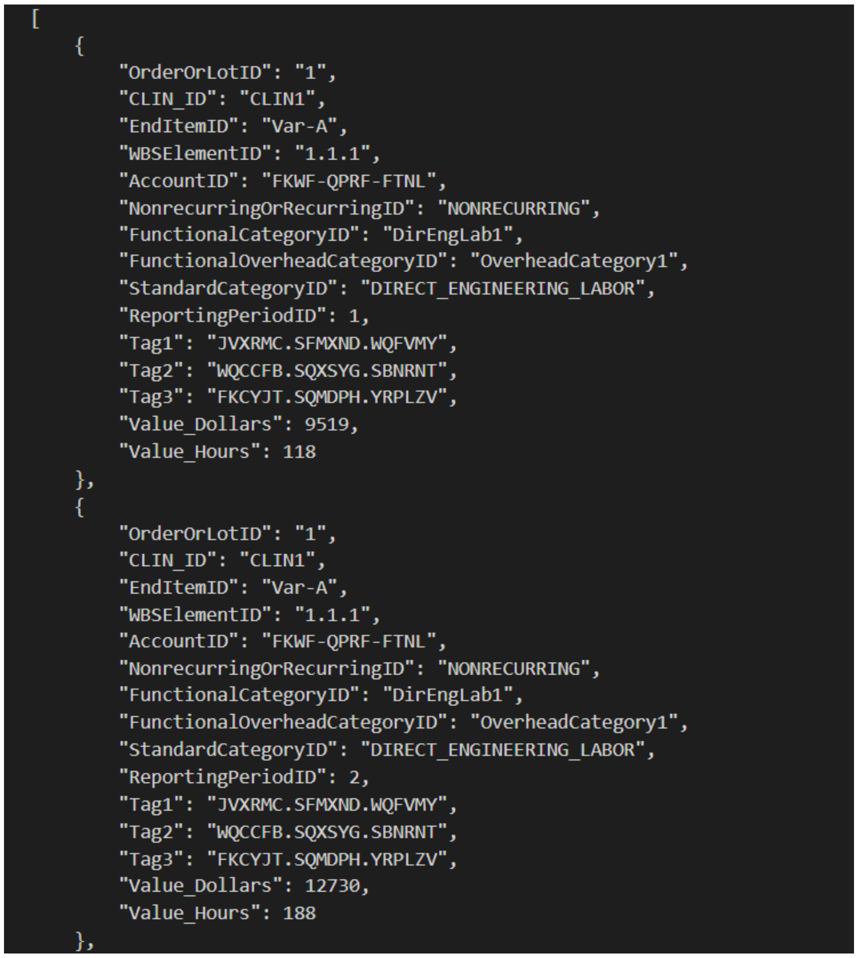
Unstructured Data

While structured data can be stored as tables, unstructured data cannot be stored in a pre-defined column and row format. A perfect example of this would be text from an open-ended survey question, or videos from a conference. There are additional examples and explanations of unstructured data in [17]. With some additional effort, is possible to create structured data from unstructured data.

It is unclear if unstructured data is ever represented in a formal type of data model like structured data can be, as that does appear to be uncommon [18]. It would be hard to represent unstructured data in a diagram because there is no standard format. This is important to note for a diagram created for the case study is explained in a later section.

Semi-Structured Data

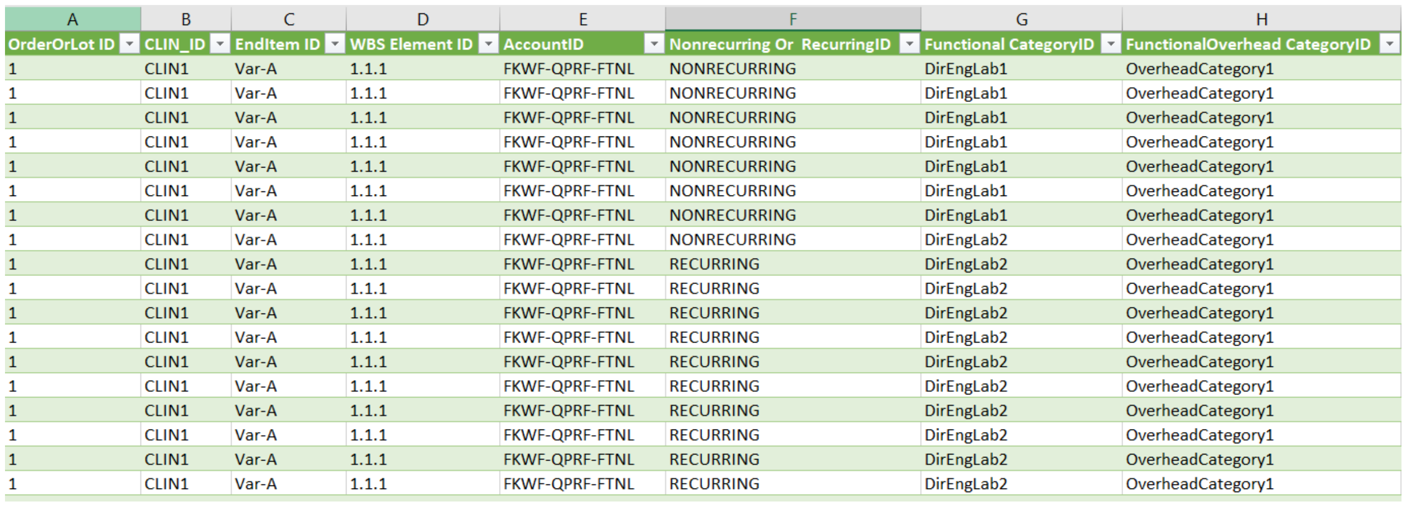
Semi-structured data is another type of data that cannot be placed into tables. Unlike unstructured data, though, it has keys and tags that keep the data organized. A great example of this is JSON data: a JSON file is **not** in a tabular format, but there is an organizational structure to it. An example is shown below from JSON cost and effort data downloaded from the cPet demo section on the CADE website. Each set of curled brackets are a new entry into the data repository [19].



```
[
    {
        "OrderOrLotID": "1",
        "CLIN_ID": "CLIN1",
        "EndItemID": "Var-A",
        "WBSElementID": "1.1.1",
        "AccountID": "FKWF-QPRF-FTNL",
        "NonrecurringOrRecurringID": "NONRECURRING",
        "FunctionalCategoryID": "DirEngLab1",
        "FunctionalOverheadCategoryID": "OverheadCategory1",
        "StandardCategoryID": "DIRECT_ENGINEERING_LABOR",
        "ReportingPeriodID": 1,
        "Tag1": "JVXRMC.SFMXND.WQFVMY",
        "Tag2": "WQCCFB.SQXSYG.SBNRNT",
        "Tag3": "FKCYJT.SQMDPH.YRPLZV",
        "Value_Dollars": 9519,
        "Value_Hours": 118
    },
    {
        "OrderOrLotID": "1",
        "CLIN_ID": "CLIN1",
        "EndItemID": "Var-A",
        "WBSElementID": "1.1.1",
        "AccountID": "FKWF-QPRF-FTNL",
        "NonrecurringOrRecurringID": "NONRECURRING",
        "FunctionalCategoryID": "DirEngLab1",
        "FunctionalOverheadCategoryID": "OverheadCategory1",
        "StandardCategoryID": "DIRECT_ENGINEERING_LABOR",
        "ReportingPeriodID": 2,
        "Tag1": "JVXRMC.SFMXND.WQFVMY",
        "Tag2": "WQCCFB.SQXSYG.SBNRNT",
        "Tag3": "FKCYJT.SQMDPH.YRPLZV",
        "Value_Dollars": 12730,
        "Value_Hours": 188
    },
```

Figure 3: Sample CPet JSON demo files downloaded from [19]

JSON data could be converted to a tabular format easily using Excel, but it does not always end up being well-populated. The table is well-filled out for this file because the tags are consistent across all the entries in the data (which makes sense for a sample file):



A	B	C	D	E	F	G	H
OrderOrLot ID	CLIN_ID	EndItem ID	WBS Element ID	AccountID	Nonrecurring Or RecurringID	Functional CategoryID	FunctionalOverhead CategoryID
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	NONRECURRING	DirEngLab1	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	NONRECURRING	DirEngLab1	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	NONRECURRING	DirEngLab1	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	NONRECURRING	DirEngLab1	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	NONRECURRING	DirEngLab1	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	NONRECURRING	DirEngLab1	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	NONRECURRING	DirEngLab1	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	NONRECURRING	DirEngLab2	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	RECURRING	DirEngLab2	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	RECURRING	DirEngLab2	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	RECURRING	DirEngLab2	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	RECURRING	DirEngLab2	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	RECURRING	DirEngLab2	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	RECURRING	DirEngLab2	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	RECURRING	DirEngLab2	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	RECURRING	DirEngLab2	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	RECURRING	DirEngLab2	OverheadCategory1
1	CLIN1	Var-A	1.1.1	FKWF-QPRF-FTNL	RECURRING	DirEngLab2	OverheadCategory1

Figure 4: CPet demo file converted to an Excel table

On Handling Unstructured and Semi-Structured Data

To summarize the past sections, structured data is data that easily fits into a table, and the format is determined ahead of time. Semi-structured data cannot fit into a table but still has identifying tags on the data. Unstructured data does not align with either format.

Cost analysts and engineers are typically business users and need the data to be structured [16]. There is no section of the CEBOK that teaches analysts in the cost community on how to handle unstructured data or semi-structured data [20]. After all, CERs are created from structured data. Therefore, it is necessary to make useful unstructured and semi-structured data structured for analysis. If semi-structured and unstructured data has a unique identifier tying it to data that is already structured, it will be possible to merge the two. This is demonstrated in a paper by Mansuri and Sarawagi [21] which extracts a structured format from unstructured text files.

Non-Relational Databases

As mentioned before, it is not possible to store semi-structured and unstructured data in relational databases without any modifications. To store these types of data in a RDBMS, changing the data structure after initialization takes time and effort. It is good to store unstructured data in an appropriate repository before it can be transformed into structured data.

Therefore, non-relational databases have been developed to handle these different types of data in a flexible way. They are called NoSQL, or "not only SQL" databases. All three types of data can be stored in these repositories at the same time [22]. Like SQL, NoSQL is not a common language in the cost estimation community. It is likely that a database administrator would need to set up these types of repositories. To read more about the different types of non-relational databases, see [23].

Storing Large Amounts of Different Types of Data

The previous sections reviewed basic data models and types of data. This section will review some ways that multiple types of databases can be stored in the same repository. There are two different storage models that will be briefly discussed for large amounts of data: data warehouses and data lakes. They are environments

meant to store current and historical data meant for analysis, not just the present version of the dataset. They also store ways to extract and normalize data, along with analysis tools [24]. Structured data can be stored in data warehouses. Data lakes are mainly for unstructured data that might not be used [25]. Once the unstructured data is processed, it could be moved into a data warehouse.
The differences between the two can be summed up in this table:

	Data Warehouse	**Data Lake**
Data Structure	Structured - collection of relational databases	Unstructured - collection of any type of database
Schema Creation	Schema is created before data is written to the repository ("schema-on-write")	Schema is created when the data is read from the repository ("schema-on-read")
Structure Flexibility	Not Agile, takes more effort to add new sources of data	Very Agile, new sources of data scan be added quickly
Type	SQL	NoSQL
Processing Speed	Faster	Slower
Normalization Method	Extract – Transform – Load (ETL)	Extract – Load – Transform (ELT)

Table 6: Chart modified from [26] *and* [27]

If structured, unstructured, and semi-structured data is collected, what kind of schema can be used? There are steps that can be taken to combine all three types of data in a hybrid data lake, which could also transfer some (or all) of the data into a Data Warehouse with an Extract Transform Load (ETL) process:

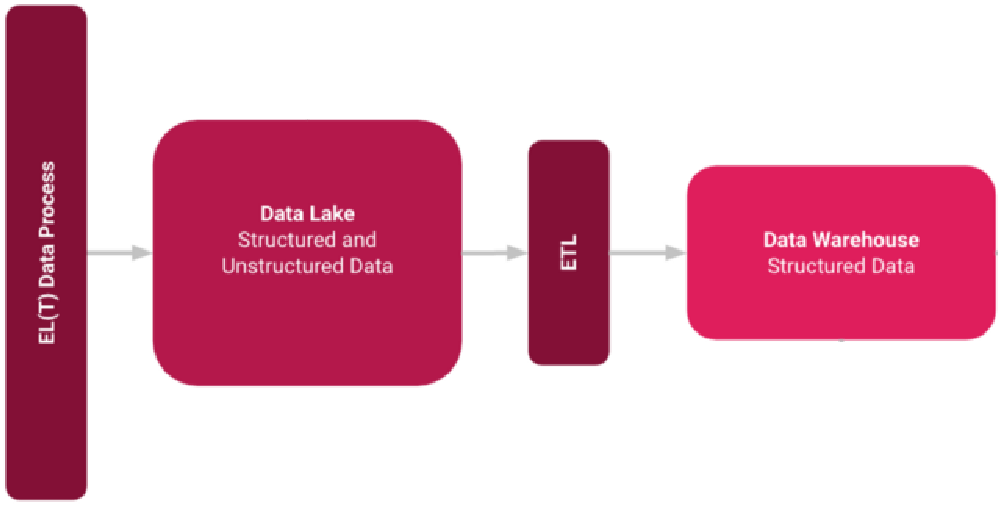
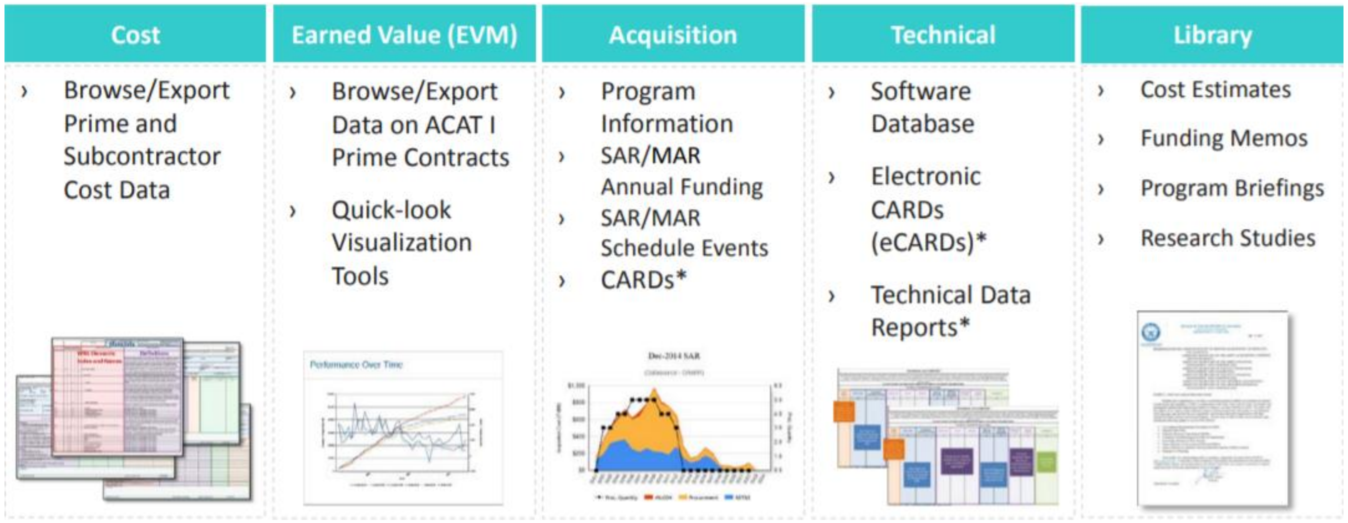


Figure 5: Chart modified from [26]

Existing Data Structures and Repositories in Cost Estimation

It is likely that many or all government agencies and contractors store data in a repository that is queried by an analyst. However, because data is sensitive, it is understandable that these organizations do not publicly describe their methodologies and schemas used to store data. In this section, two examples of DoD organizations or contractors that use data repositories to store data are highlighted.

The Cost Assessment Data Enterprise (CADE) takes semi-structured JSON or XML and creates relational databases from the data [28]. The data that is stored relates to acquisition, technical, effort, earned value management, and program information. This also includes raw data (the original submission of the data), and a document repository which is unstructured. The CADE is a good example of a data repository with both structured and unstructured data. The data is also presented in dashboards for the user to analyze. Data in CADE can be queried through a point and click menu, so using SQL or NoSQL is not necessary for accessing this data repository [29].



Cost	Earned Value (EVM)	Acquisition	Technical	Library
› Browse/Export Prime and Subcontractor Cost Data	› Browse/Export Data on ACAT I Prime Contracts › Quick-look Visualization Tools	› Program Information › SAR/MAR Annual Funding › SAR/MAR Schedule Events › CARDs*	› Software Database › Electronic CARDs (eCARDs)* › Technical Data Reports*	› Cost Estimates › Funding Memos › Program Briefings › Research Studies

Figure 6: Summary of structured and unstructured data in CADE repository, from [30]

As mentioned earlier, the FLIS, which is a database of spare parts, is maintained by the Defense Logistics Agency (DLA). Data can be accessed through WebFLIS, or other companies that combine WebFLIS data with other data sources such as LogiQuest or Haystack (IHS). Data can be queried through the search bars in those programs [31].

Case Study

This section proposes a data model for a hybrid data lake that is a combination of all three types of data: structured, semi-structured, and unstructured. Unfortunately, this schema is still in the planning stages and has not been fully implemented yet. Screenshots included in this section are draft screenshots from the current data.

This planned data repository includes hardware data from several projects that totals over 10,000 datapoints. The original data came from sources that focused on military electronics, commercial communications equipment, and firearms. In addition, there was aerospace and defense data from many different sources that the Unison Cost Engineering division had collected over several decades.

This section of the paper does not show all the attributes (columns) featured in the data, but it shows many of the important attributes.

Data Structure

Each of these categories represented in the repository will be stored separately. Each will be reviewed in detail:

Data Category	Data Type
1. Original Data Files	Any
2. Technical Data	Semi-Structured Collection of Tables
3. Categorical Data	Structured
4. Project Metadata	Structured
5. Component Level Data Sources and Data Quality	Structured
6. Component Financial Data	Structured
7. Calibrated/Tailored Data	Semi-Structured Collection of Tables
8. Resource Cost Data	Structured
10. Documentation	Any

Table 7: Data categories and their respective data types

For the purposes of this project, the structured and semi-structured data will be displayed in a diagram that is inspired by the ERD. Although semi-structured data cannot be represented in an ERD, the primary keys can still be displayed. For the purpose of brevity, not all column names will be included in this diagram. Altogether this repository is over 250 columns of data, so it is not sensible to name all the columns of data. Each row of the original data represents a hardware component, which is assigned a unique Index. The project or unique data source is also supplied with a Project Index. Most of the tables are related to one another by either the Index or Project Index.

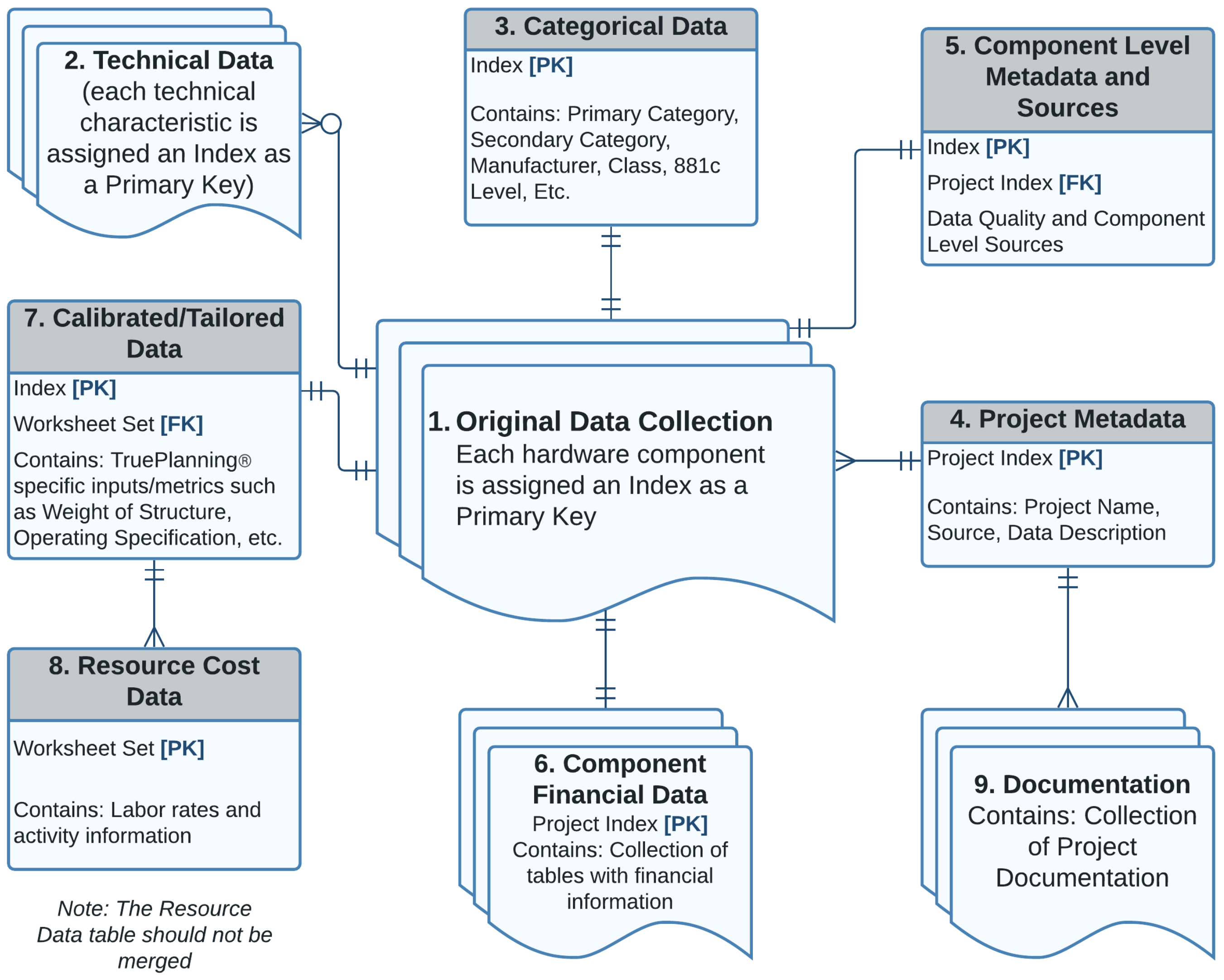

2. Technical Data
(each technical characteristic is assigned an Index as a Primary Key)
3. Categorical Data
Index [PK]
Contains: Primary Category, Secondary Category, Manufacturer, Class, 881c Level, Etc.
5. Component Level Metadata and Sources
Index [PK]
Project Index [FK]
Data Quality and Component Level Sources
7. Calibrated/Tailored Data
Index [PK]
Worksheet Set [FK]
Contains: TruePlanning® specific inputs/metrics such as Weight of Structure, Operating Specification, etc.
1. Original Data Collection
Each hardware component is assigned an Index as a Primary Key
4. Project Metadata
Project Index [PK]
Contains: Project Name, Source, Data Description
8. Resource Cost Data
Worksheet Set [PK]
Contains: Labor rates and activity information
6. Component Financial Data
Project Index [PK]
Contains: Collection of tables with financial information
9. Documentation
Contains: Collection of Project Documentation
Note: The Resource Data table should not be merged

Figure 7: ERD-like diagram of data categories

1. Original Data Files

The original data can come in any format. Most of this data comes in Excel files made up of one or more tables. But future data could be JSON or XML files. Since each data file has a unique format, the data will be semi-structured. The original data should be included in the repository for traceability purposes. Each row represents a hardware item, so it should be assigned a unique Index.

2. Technical Data

Technical Data is a collection semi-structured and unstructured data. This type of data usually comes in columns, so it looks structured, but the columns are very different for each project. For example, data from a historical data study of military and commercial equipment has the following technical columns:

Index	Generic Equipment Type	Frequency Band	Technology
MoD_209	COMMS	HF	Hybrid
MoD_292	COMMS	VHF	Semicon
MoD_291	COMMS	HF	Hybrid
MoD_249	COMMS	HF	Hybrid
MoD_257	COMMS	VHF	
MoD_250	COMMS	HF	Hybrid
MoD_163	COMMS	HF	Semicon, Digital
MoD_164	COMMS	HF	Semicon, Digital

Table 8: Military and commercial equipment related technical characteristics

Data gathered from a gun and ammunition research project has a completely different set of technical columns. A sample is shown below:

Index	Caliber	Capacity	Barrel Length	Action Type
GunAmmo_120	.243 Win	10	25.625 inches Heavy Profile	Bolt Action
GunAmmo_121	.308 Win	10	25.625 inches Heavy Profile	Bolt Action
GunAmmo_122	.300 Blackout	5	16 Inches (1:8) (Threaded)	Semi-Auto
GunAmmo_123	.30-30 Win	5	20 inches	Lever Action
GunAmmo_124	.45-70 Government	4	20 inches	Lever Action
GunAmmo_125	.22 LR	15-20	20 inches, Octagonal	Lever Action
GunAmmo_126	Other	10	20 inches, Octagonal	Lever Action
GunAmmo_127	Other	7	16.5 inches Octagonal	Lever Action
GunAmmo_128	Other	7	16.5 inches Octagonal	Lever Action

Table 9: Gun and ammunition related technical characteristics

These are just examples of the technical columns for different projects that currently exist in the data. It is not memory-efficient to make one large table of all the technical columns: there would be dozens of mostly blank columns. As of now, it would be best to keep these technical data tables separate. Thus, the Technical Data should be a collection of semi-structured data because the column names and value types for future projects cannot be anticipated ahead of time.

The repository also contains semi-structured technical data that is stored in a unique way. The data in Table 10 demonstrates a small sample of this data. The Property column contains the names of the technical attributes. The Value column contains the values of their respective properties.

Index	Property	Value
000014072	AVERAGE POWER RATING PER CHANNEL	27.0 WATTS OUTPUT
000033553	AVERAGE POWER RATING PER CHANNEL	1.0 KILOWATTS OUTPUT
000033553	BASIC SHAPE STYLE	1 RECTANGULAR OR SQUARE
000033559	ATTENUATION IN DECIBELS	4.0
000033559	AVERAGE POWER RATING PER CHANNEL	20.0 WATTS OUTPUT
000033559	BASIC SHAPE STYLE	RECTANGULAR OR SQUARE
000041222	ATTENUATION IN DECIBELS	80.0
000041222	BASIC SHAPE STYLE	RECTANGULAR OR SQUARE
000042019	BASIC SHAPE STYLE	RECTANGULAR OR SQUARE
000061816	ATTENUATION IN DECIBELS	10.0
000071876	AVERAGE POWER RATING PER CHANNEL	0.5 WATTS OUTPUT .
000079041	AVERAGE POWER RATING PER CHANNEL	1.0 WATTS OUTPUT
000082539	BASIC SHAPE STYLE	RECTANGULAR OR SQUARE
000087959	ATTENUATION IN DECIBELS	3.0
000087959	BASIC SHAPE STYLE	RECTANGULAR OR SQUARE
000087960	BASIC SHAPE STYLE	RECTANGULAR OR SQUARE
000099343	BASIC SHAPE STYLE	CYLINDRICAL

Table 10: Different layout of technical data table

For now, this is an efficient way to store data. Otherwise, each unique value in the Property table would be a different column, which could lead to a table with several dozen different columns that are mostly empty. Table 11 shows this alternate view of Table 10, where the data is pivoted to include the Property column values as header. Note that there are a lot of blank values in this view:

Index	AVERAGE POWER RATING PER CHANNEL	BASIC SHAPE STYLE	ATTENUATION IN DECIBELS
000014072	27.0 WATTS OUTPUT		
000033553	1.0 KILOWATTS OUTPUT	1 RECTANGULAR OR SQUARE	
000033559	20.0 WATTS OUTPUT	RECTANGULAR OR SQUARE	4.0
000041222		RECTANGULAR OR SQUARE	80.0
000042019		RECTANGULAR OR SQUARE	
000061816			10.0
000071876	0.5 WATTS OUTPUT		

Index	AVERAGE POWER RATING PER CHANNEL	BASIC SHAPE STYLE	ATTENUATION IN DECIBELS
000079041	1.0 WATTS OUTPUT		
000082539		RECTANGULAR OR SQUARE	
000087959		RECTANGULAR OR SQUARE	3.0
000087960		RECTANGULAR OR SQUARE	
000099343		CYLINDRICAL	

Table 11: Pivoted version of Table 10

This repository is representative of many different types of hardware components, so there are currently dozens of different types of technical columns for the hundreds of types of hardware components stored in this data repository.

3. Categorical Data

Categorical data includes things like manufacturer, class, primary category, secondary category, and if the data is proprietary (related to data security). Categorical data includes WBS/PBS breakdown structure assignments when applicable. There might seem to be some overlap between the Categorical and Technical data tables. However, Categorical Data is a set number of attributes and is therefore structured. Technical data is everything outside this set list of attributes. See the table below for a condensed example of the table:

Index	Primary Category	Class	881c Level1	Proprietary
KN_1	Airframe	Military	Aircraft System	No
KN_2	Fuselage	Military	Aircraft System	No
KN_3	Window Assembly	Military	Aircraft System	No
KN_4	Spoiler	Military	Aircraft System	No
KN_5	Aileron	Military	Aircraft System	No
KN_6	Throttle Quadrant	Military	Aircraft System	No
KN_7	Wing Flaps	Military	Aircraft System	No

Table 12: Categorical data table

4. Project Metadata

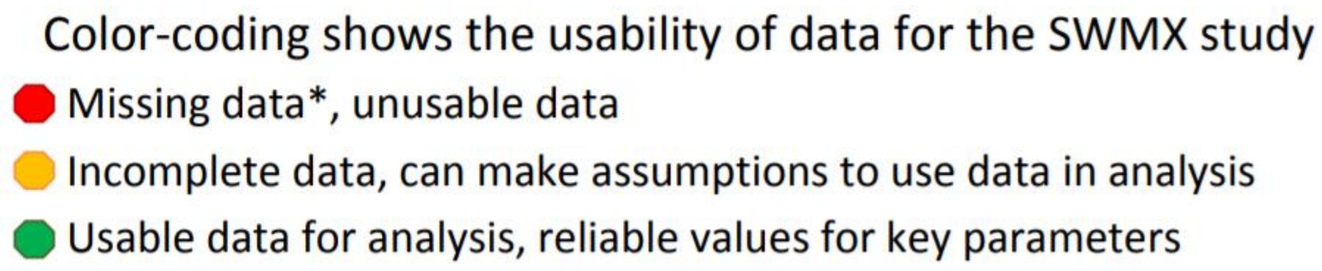
Project Metadata is a structured table that describes each of the projects at a higher level. It also includes the version of the TruePlanning software that the data was calibrated in, which will be explained further in the Calibrated/Tailored Data table. Overall, this is a simple and small table.

Project Index	Project Name	Project Data Description	Year of Study	TruePlanning Calibration Version
GunAmmoProj	Gun and Ammunition Item Lists	Firearms and Ammunition data webscraped from Internet	2018	16.2
KN	TruePlanning Knowledge Network	Legacy open-source data from PRICE Systems.	Not Available	Not Available
EO_IR	FLIR Item List	FLIR data gathered from the internet	2018	16.2

Table 13: Project Metadata Table

5. Component Level Data Metadata and Sources

This is a structured table that will include component level data such as data source, Project Index, the date the data was collected, and data quality. Note that the data quality descriptions are based on the data quality guidelines set from [32].



Color-coding shows the usability of data for the SWMX study

🔴 Missing data*, unusable data

🟡 Incomplete data, can make assumptions to use data in analysis

🟢 Usable data for analysis, reliable values for key parameters

Figure 8: Color-coded data quality categories from [32]

A small subset of the table is shown below:

Index	Project Index	Source	Data Quality - Total Weight	Date
GunAmmo_1002	GunAmmoProj	https://www.hyattgunstore.com/heckler-koch-vp9-flat-dark-earth-9mm-pistol-with-standard-sights.html	Green	4/12/2018
GunAmmo_1003	GunAmmoProj	https://www.hyattgunstore.com/heckler-koch-vp9-flat-dark-earth-slide-od-green-frame-9mm-pistol-with-standard-sights.html	Green	4/12/2018
GunAmmo_1004	GunAmmoProj	https://www.hyattgunstore.com/heckler-koch-vp9-9mm-pistol-with-standard-sights.html	Green	4/12/2018
GunAmmo_1005	GunAmmoProj	https://www.hyattgunstore.com/heckler-koch-vp9-sk-subcompact-9mm-pistol-with-night-sights-and-three-magazines.html	Green	4/12/2018

Index	Project Index	Source	Data Quality - Total Weight	Date
GunAmmo_1006	GunAmmo Proj	https://www.hyattgunstore.com/henry-mares-leg-pistol-in-22lr.html	Green	4/12/2018
GunAmmo_1007	GunAmmo Proj	https://www.hyattgunstore.com/henry-h006ml-mares-leg-lever-44-rem-mag-12.9-5-1-american-walnut-grip-blue.html	Green	4/12/2018

Table 14: Component-level data quality and source table

6. Component Financial Data

Financial data comes in different fiscal year references and currencies. It will need to be normalized to a specific fiscal year. This data could also include effort hours, activities, or phases in costs/hours, if the data is that specific. Every project could be normalized to a different currency or base year. While each individual table is structured, because the fiscal year and currency may be different, the headers for the data will need be different. Therefore, this data should be stored in more than one relational table. This column name format is unique to the TruePlanning software.

Index	Unit Production Cost (Currency;£,GBR,2003;Metric)
MoD_103	99.95
MoD_104	169.95
MoD_107	799
MoD_109	536.6220705
MoD_110	46.60194292
MoD_111	58.26700998
MoD_118	93.30887144

Table 15: Financial data table in British Pounds

Index	Unit Production Cost (Currency;$,USA,1984;Metric)
MoD_100	75.13627277
MoD_101	105.1907819
MoD_102	145.2634607
MoD_105	60.88041728
MoD_106	95.17261218
MoD_108	165.2998001

Table 16: Financial data table in USA dollars

7. Calibrated/Tailored Data

For certain organizations, data must be tailored to the models used in that organization. In this case, data is tailored to the TruePlanning inputs, schedule, and metrics. This case study involves hardware data. Hardware data is calibrated to inputs such as Weight of Structure, Weight of Electronics, Platform (Operating Specification), Manufacturing Complexity for Structure, and Manufacturing Complexity for Electronics. It also includes start and end dates for activities, which will be automatically generated by the model if it is not inputted by the user. This is also connected to the resource cost data through the Worksheet Set Column. This table is several hundred columns large, so a small subset of the data is shown below:

Index	Worksheet Set	Unit Production Cost (Currency;$,USA, 2018;Metric)	Total Weight (Weight;lbs; Metric)	Weight of Structure (Weight;lbs; Metric)	Weight of Electronics (Weight;lbs; Metric)
GunAmmo_2	GunAmmoWS	1575.956	6	6	0
GunAmmo_3	GunAmmoWS	1566.889	5.15	5.15	0
GunAmmo_4	GunAmmoWS	1575.956	6	6	0
GunAmmo_5	GunAmmoWS	1566.889	5.15	5.15	0
GunAmmo_6	GunAmmoWS	1575.956	6	6	0
GunAmmo_7	GunAmmoWS	987.7041	7.3	7.3	0
GunAmmo_8	GunAmmoWS	987.7041	7.3	7.3	0
GunAmmo_9	GunAmmoWS	1072.572	7.1	7.1	0

Table 17: Sample of Calibrated and Tailored data columns

8. Resource Cost Data

This data includes labor rates by resource based on the project. In this case, labor rates are from the TruePlanning software, which are determined based on industry averages. Since each Worksheet set is made up of many rows, it does not make sense to merge this table with the Calibrated/Tailored Data. A sample of the columns are below:

Worksheet Set	Cost Object	Activity	Resource	Country	Unit Cost	Cost Unit
GunAmmoWS	Hardware Component	Development Engineering		United States		
GunAmmoWS	Hardware Component	Development Engineering	Design Engineering	United States	98088.34	$/Year
GunAmmoWS	Hardware Component	Development Engineering	System Engineering	United States	127403.67	$/Year
GunAmmoWS	Hardware Component	Development Engineering	Support Engineering	United States	90293.21	$/Year

Table 18: Sample of Resource Cost data columns

9. Documentation

   The final category is documentation. This unstructured data exists for traceability purposes. It includes Word files which serve as documentation that explains data normalization. ETL files such as Python code should also be stored for traceability if there aren't any organizational repositories that hold code. There should be some type of tagging system so it is clear which project the documentation is tied to.

Conclusion

Data gathered from cost estimation projects can be from several different fields (or entities), such as financial data, technical data, and metadata. Therefore, it makes sense to organize this data based on these topics, or "entities". In addition, data can be structured, unstructured, or semi-structured. This also impacts how data will be stored in a repository.

Metadata can be structured, and so can common technical attributes. But there is plenty of data in cost estimating that comes in an unstructured or semi-structured format. Therefore, it is best to have a repository that combines the three types of data. Combining structured, semi-structured, and unstructured cost estimation data in a repository can be beneficial for maintaining an authoritative and accessible source of data. Although a cost estimator is unlikely to have the skillset needed for building a repository like a data lake or a data warehouse, this paper is still useful in introducing the concepts and providing a general outline for how a repository of cost estimation data might look like.

References

[1] [Online]. Available: https://www.cmu.edu/cee/projects/PMbook/05_Cost_Estimation.html.

[2] Enterprise Big Data Framework, "Three different data structures," [Online]. Available: https://www.bigdataframework.org/data-types-structured-vs-unstructured-data/.

[3] S. Conger, Hands-On Database: An Introduction to Database Design and Development.

[4] "AGM-80 Viper," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/AGM-80_Viper.

[5] "Bell AH-1Z Viper," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Bell_AH-1Z_Viper.

[6] "List of U.S. DoD aircraft designations," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/List_of_U.S._DoD_aircraft_designations.

[7] "Surrogate Key vs Natural Key Differences and When to Use in SQL Server," MS SQL Tips, [Online]. Available: https://www.mssqltips.com/sqlservertip/5431/surrogate-key-vs-natural-key-differences-and-when-to-use-in-sql-server/.

[8] Lucidchart homepage, "Entity-Relationship Diagram Symbols and Notation," [Online]. Available: https://www.lucidchart.com/pages/ER-diagram-symbols-and-meaning.

[9] Microsoft, "Try SQL Server on-premises or in the cloud," [Online]. Available: https://www.microsoft.com/en-us/sql-server/sql-server-downloads.

[10] Oracle, "Oracle Database XE," [Online]. Available: https://www.oracle.com/database/technologies/appdev/xe.html.

[11] The PostgreSQL Global Development Group, "PostgreSQL: The World's Most Advanced Open Source Relational Database," [Online]. Available: https://www.postgresql.org/.

[12] Microsoft, "Elevate data," [Online]. Available: https://www.microsoft.com/en-us/microsoft-365/access.

[13] "Excel Specifications and Limits," Microsoft, [Online]. Available: https://support.microsoft.com/en-us/office/excel-specifications-and-limits-1672b34d-7043-467e-8e27-269d656771c3.

[14] Terabase Corporation, "LogiQuest," [Online]. Available: http://www.logiquest.com/.

[15] Defense Logistics Agency, *PUBLOG.*

[16] "Structured vs. Unstructured Data: What’s the Difference?," IBM, [Online]. Available: https://www.ibm.com/cloud/blog/structured-vs-unstructured-data.

[17] MongoDB, "Unstructured Data," [Online]. Available: https://www.mongodb.com/unstructured-data.

[18] Wikipedia, "Data Model," [Online]. Available: https://en.wikipedia.org/wiki/Data_model#Overview.

[19] CADE, "CSDR Tools," [Online]. Available: https://cade.osd.mil/tools/csdr-tools.

[20] ICEAA, "COST ESTIMATING BODY OF KNOWLEDGE (CEBoK®) SYLLABUS," [Online]. Available: http://www.iceaaonline.com/ready/wp-content/uploads/2014/02/CEBoK_Syllabus.pdf.

[21] I. R. Mansuri and S. Sarawagi, "Integrating unstructured data into relational databases," [Online]. Available: https://www.cse.iitb.ac.in/~sunita/papers/icde06a.pdf.

[22] IBM, "NoSQL Databases," [Online]. Available: https://www.ibm.com/cloud/learn/nosql-databases.

[23] Microsoft, "Non-relational data and NoSQL," [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data.

[24] Oracle, "1 Data Warehousing Concepts," [Online]. Available: https://docs.oracle.com/cd/B10501_01/server.920/a96520/concept.htm#.

[25] Talend, "Data Lake vs Data Warehouse," [Online]. Available: https://www.talend.com/resources/data-lake-vs-data-warehouse/.

[26] Christianlauer, "Benefits of a Hybrid Data Lake," Towards Data Science, [Online]. Available: https://towardsdatascience.com/what-is-a-hybrid-data-lake-b7ef2c3cce0c.

[27] T. Henson, "How Schema On Read vs. Schema On Write Started It All," [Online]. Available: https://www.dell.com/en-us/blog/schema-read-vs-schema-write-started/.

[28] F. Janicki, "CADE Update," [Online]. Available: https://www.ndia.org/-/media/sites/ndia/divisions/ipmd/2020-01-meeting/b5-cade-update_200205231657.ashx.

[29] CADE, "CADE USER MANUAL," [Online]. Available: https://cade.osd.mil/content/cade/files/CADEUserGuides/D&A_User%20Manual_December2018.pdf.

[30] M. D. Fullwood and P. Braxton, "Lunch and Learn Defense Acquisition University," [Online]. Available: https://www.dau.edu/Lists/Events/Attachments/47/08-09-2017%20DAU-Lunch-Learn-CADE-final_MTaylor.pdf.

[31] Defense Logistics Agency, "Web Federal Logistics Information System (WebFLIS)," [Online]. Available: https://www.dla.mil/HQ/InformationOperations/About/Offers/Applications/WebFLIS/.

[32] C. C. Wallshein, W. Rosa, J. P. Dean, V. V. Welker and P. J. Braxton, "Software Maintenance Data Collection and Estimating Challenges," [Online]. Available: http://www.iceaaonline.com/ready/wp-

content/uploads/2017/09/LCC01_Presentation_SoftwareMaintenanceDataCollectionandEstChallenges_Welker.pdf.