ITRC has developed a series of fact sheets that summarizes the latest science, engineering, and technologies regarding environmental data management (EDM) best practices. This fact sheet describes:

- questions to consider when selecting an environmental data management system (EDMS)
- the importance of identifying the appropriate data model(s) for the environmental data to be managed
- alternative technologies for the implementation of an EDMS

Additional information related to EDMS is provided in the ITRC fact sheets that are presented under Data Management Planning, Data Quality, and Data Exchange topics.

1 INTRODUCTION

This fact sheet is intended to assist users in making decisions about choosing an environmental data management system (EDMS) that meets their needs and fits with their available resources. Different EDMS options may be best suited for different projects or programs. The information presented in this fact sheet may be particularly helpful in the following situations:

- New program with new data collection/reporting requirements.
- Existing programs whose current system is not adequately robust, customizable, efficient, accessible, or responsive, or has become obsolete due to lack of security or compatibility with modern operating systems.
- Increased scale of incoming data that is surpassing the capabilities of the current system.
- New requirements for data reporting or analysis that cannot be handled with the existing system.
- Opportunities for increased cost efficiency by transitioning to a more modern system.

Please note that although certain specific products may be mentioned in this fact sheet as examples, neither the Environmental Data Management (EDM) Best Practices Team nor ITRC offer these examples as an endorsement or recommendation.

2 ASSESS PROGRAM OR PROJECT NEEDS

Developing or deploying an EDMS requires making several decisions. Under the overall question of “How shall I implement an EDMS?,” there are several more specific decisions to address. These include:

- What data are needed for the analyses and reporting that need to be done?
    - What are the analyses to be conducted and the reporting requirements?
    - What types of data does each analysis need?
    - What is the quantity of data?
    - What are the sources of data?

- What is the most appropriate data model (see below) for these data?
    - What is the natural structure of the data?
    - What sampling or data characteristics must be represented to support analyses? (An example is replication to assess variability.)

- How do data users want to either access or be provided the data?
    - Direct access to database tables and views?
    - A graphical user interface allowing customized data selection and summarization?
    - A map interface? Do geospatial analyses need to be conducted within the database?
    - Exports in the form of spreadsheets, comma-separated values (CSV) files, or shapefiles?
    - Exports in the form of electronic data deliverables (EDDs) for import to other systems?
    - Exports in the form of report tables?
    - Is there a need to keep some data private or confidential? Can the system accommodate this need? How is it implemented?
    - For “unstructured” data such as photos, videos, or PDF documents, do these need to be accessed in

exports? Does your system allow for storage of these types of files, and if not, are links to external resources preferred or some other system of retrieval?
    - Other methods of data delivery, such as preformatted reports or submittals to a regulatory agency—Can the system directly export these sorts of data deliverables? More ideally, can these outputs be automated?

- What are the short-term vs. long-term requirements?
    - Are there priority analyses or reporting requirements to be met?
    - Are there priority users to be accommodated?
    - How do the requirements and desired schedule for EDMS implementation interact with organizational information technology (IT) capabilities and data management plans and governance? (See also the Data Management Planning and Data Governance Fact Sheets)

- Is there commercial software or other available data models (for example, from other state or federal systems) that can be used?
- Is the necessary level of technical expertise and funding available in the short term to create or install and to support the EDMS?
- Is the necessary level of technical expertise and funding available in the long term to maintain and operate—and upgrade, if necessary—the EDMS?

The needs of the program should drive identification of the requirements of the EDMS. The ability to achieve technical requirements may be tempered by resource limitations (funding, staff, or skills). Having a thorough understanding of the technical requirements can help ensure that the costs and benefits of any tradeoffs are also well understood.

Additional considerations that may affect these tradeoffs include:

- What are the costs of the current system, as well as the costs of any replacement?
    - Development—short-term and possibly long-term
    - Data migration—moving data from an existing system to a new one
    - Maintenance—software, hardware, licenses, security updates, staffing, and training
    - Operations—routine acquisition, presentation, analysis, and dissemination of data
    - Data storage for systems that require server space or cloud-based hosting

- What are the costs in terms of relatively intangible measures such as data quality, user experience for program staff, compatibility with other systems, and public acceptance?
- How do the organization's data governance and IT policies affect costs and decision-making?

Other information relevant to a needs assessment for an EDMS is presented in the Data Management Planning fact sheets.

3 IDENTIFY APPROPRIATE DATA MODELS

Environmental data are collected to represent, or model, real-world conditions. These conditions ordinarily include entities such as the locations sampled, the samples collected, and the measurements made on those samples. There are many-to-one relationships between entities like these. For example, there may be many samples at each location, and many measurements made on each sample. For an EDMS to efficiently handle environmental data, it must be able to represent (that is, model) all the entities and relationships of importance.

These three entities—locations, samples, and measurements—are ordinarily core components of any system for handling environmental characterization data. In practice, this simple set of entities often must address several additional levels of detail or complexity:

- Different investigations may use the same name for different geographic locations. Similarly, and conversely, different investigations may use different names for the same geographic locations.
- The actual geographic coordinates at which a sample is collected may differ from the target coordinates.

- Sampling designs may include replication, compositing, subsampling, and splitting of samples.
- Analytical measurements may be conducted by multiple laboratories, with different methods, and may be replicated and repeated.

Figure 1 is an illustration of how the number of entities needed to accurately represent real-world complexity, after accounting for the considerations above, may be larger than the simple three-entity model. This illustration applies specifically to environmental chemistry measurements. In this diagram, a gear deployment refers to the use of a piece of equipment or a sampling method that results in the collection of material such as soil, sediment, or surface water. An interpretive sample is a quantity of such material that represents environmental conditions at a single point in space and time, that is presumed to be uniform in physical and chemical composition, and that will be used to evaluate and interpret environmental conditions. A collection represents one or more interpretive samples that are closely related, such as horizons of a soil boring, where the entire boring is the collection and each horizon is an interpretive sample. Analytical samples represent subdivisions of an interpretive sample that are typically created and designated for separate analyses as a quality

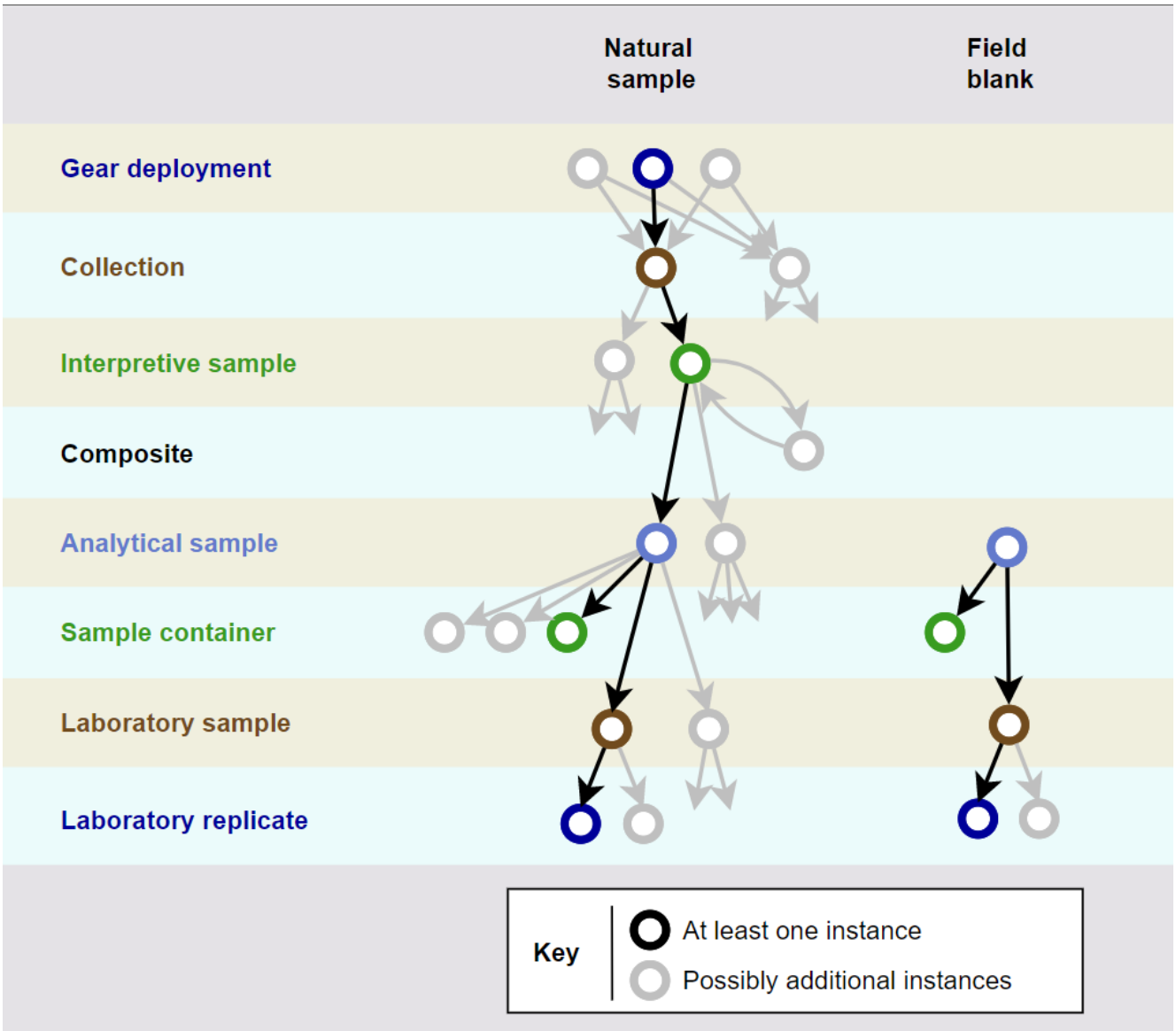

Figure 1. Conceptual model of an environmental sample, and the many different permutations that may result from one sampling event. This level of complexity is necessary to consider when developing or evaluating a data model.
Source: Integral Consulting, Inc. (used with permission)

control measure. Depending on the sampling design, collections, interpretive samples, and analytical samples may be three distinct entities that underlie the higher-level “sample” entity.

Data models for biological and ecological conditions will require additional or alternate entities and relationships. Species abundance measurements, toxicity testing results, bioaccumulation testing results, and histopathology observations all

require distinct features in a data model. Further, species abundance measurements may be made by point sampling, transect sampling, distance sampling, or mark/recapture studies, each of which may require a different data model. Data streams from continuously recording instruments may require data model components that are tailored specifically to each type of instrument that is used.

Varying approaches to design and usage of a data model can be illustrated with an example of fish sampling for human health and ecological risk assessments. Such a fish sampling program may conduct multiple trawls (gear deployments) to collect large fish of a certain species for human health risk assessment and small fish for ecological risk assessments. Multiple trawls may be necessary to collect sufficient fish. The large fish from each trawl may be put in one bucket and the small fish put in another bucket until sufficient fish are caught. These buckets may be separate collections, as shown in Figure 1, and there would be a many-to-many relationship between gear deployments and collections. After trawling is complete, the small fish may be grouped into (for example) five interpretive samples of ten fish each, multiple fish per sample being required to obtain enough mass for chemical analyses. Each large fish may be dissected into left and right fillets and the carcass, each of which will be analyzed and interpreted separately and is therefore a separate interpretive sample. (Analytical samples may be created by halving some fillets and submitting the halves separately to the laboratory.) The planned interpretation of data from the large fish, however, will require mathematical reconstruction of each entire fish from the measurements made on the fillets and carcass. Thus, the fillets and carcass of each fish must be kept associated. One way to do this is to add a "fish" entity to the data model between the collection and interpretive sample entities (per Figure 1). An alternate way to do this is to consider each large fish to be a separate collection. The latter approach uses the same data model for both large and small fish, which is simpler and therefore generally to be preferred. In this example, field replicates are collected for both large fish and small fish, but note that replication is at different levels in the data model: large fish are replicated at the collection level, whereas small fish are replicated at the interpretive sample level. These kinds of considerations of both the natural structure of the data and the intended use of the data should inform selection or design of a data model.

Determining the appropriate model for the data to be managed is a key step in the selection or implementation of an EDMS. If data have attributes or structure that cannot be represented in the EDMS, the result may be omission of data, use of complex encoding systems (for example, to put multiple attributes or identifiers in a single table column), or loss of data integrity (see the Data Quality fact sheets).

When data are to be transferred from one EDMS to another (as is often the case for data transmittals between organizations), differences in data models must be accounted for so that data are not lost or misrepresented. The ITRC Electronic Data Deliverables and Data Exchange Fact Sheet discusses approaches to data transfers in more detail.

Selection or development of a data model (or multiple data models) should be carried out jointly by personnel who have different perspectives or roles in the collection and use of the data. The level of data structure detail that is recognized by different individuals can differ according to their role, as illustrated in Table 1. For example, a data analyst who carries out statistical or geospatial analyses is likely to be focused on data at the level of the interpretive sample—that is, the characterization of environmental conditions at a specific point in space and time. In contrast, staff of an analytical laboratory do not need to know which analytical samples represent the same interpretive sample; in fact, they should not have access to this information if samples are submitted to the laboratory blind (USEPA 2014). A data model that is selected or designed solely by a data analyst may not meet the needs of a data validator, and vice versa.

One other consequence of the different perspectives associated with different roles is that certain terms, such as "sample," may mean different things to different individuals. Establishing clear terminology for different elements of the data model is important to prevent adverse consequences resulting from ambiguous communications.

Table 1. Perspectives on data model entities by role

Role	Data Model Entities				
	Collection	Interpretive Sample	Analytical Sample	Sample Container	Laboratory Sample
Data analyst	X	X			
Data manager	X	X	X	X	X
Field sampler	X	X	X	X	
Data validator		X	X	X	X

Analytical lab			X	X	X

Some examples of EDMSs that implement similar but different data models are:

- Washington State's Environmental Information Management System https://ecology.wa.gov/Research-Data/Data-resources/Environmental-Information-Management-database
- California's Geotracker system https://www.waterboards.ca.gov/ust/electronic_submittal/about.html
- USEPA Region 10's Portland Harbor Interim Database http://ph-public-data.com/document/PHIDB2021/
- Federal USEPA's Scribe system: https://www.epa.gov/ert/environmental-response-team-information-management
- Federal EPA's Safe Drinking Water Information System (SDWIS) https://www.epa.gov/enviro/sdwis-overview

Review of the features of these systems may be helpful when designing or selecting a data model for a new EDMS.

Developing a thorough understanding of the data model(s) appropriate to a program or project is essential prior to selecting the type of data system to be implemented. While the discussion and examples in the section above are meant to illustrate the thought process behind these considerations, it is important to reiterate that this process will be unique to an organization's choice of EDMS to adequately manage their particular data. It is beyond the scope of this paper to delve into all the many ways that a data model may need to adapt to different sorts of data—nonstructured data or citizen science data, for instance—but our hope is that these examples encourage you to perform this sort of baseline needs assessment before embarking on implementing a new system.

4 TYPES OF DATA SYSTEMS

There are a variety of technologies that can be used when implementing an EDMS. These vary in technical capability, technical demands on the program or project, and cost. Following are descriptions of the most common solutions encountered in state government and commercial environmental firms.

The following text describes data management systems roughly in order of less to more robust/complex. Table 2 provides qualitative assessment of multiple design considerations for EDMS types. Examine the pros and cons of each and compare to your organization's needs and make your best decision on how to proceed.

4.1 Paper Files and Other Nonstructured Data

Although it may seem counterintuitive in the twenty-first century and our "digital age," sometimes pen and paper is the easiest, most efficient solution. If your data management needs require only direct reporting of field notes and lab reports, a paper filing system may be sufficient. Of course, there are many downsides to this as a "complete" system—lack of searchability, space considerations for filing cabinets, and physical resources needed to print and store documents. To be clear—**this sort of system is not recommended as a preferred way of managing data within an organization**, but sometimes budget and other limitations can leave an organization with no better options.

Paper files are often used in the field for collecting notes and observations. There are some wonderful strategies for designing paper field forms in our fact sheets on Field Data Collection. Please refer to that section for best practices on how to use paper forms in conjunction with more robust *digital* data management systems. Although those suggestions are tailored to field-specific forms, the principles could be applied to other forms of paper information as well.

Table 2. Decision matrix by EDMS type

Decision Consideration	**Environmental Data Management System Type**					
	Paper Files	**Spreadsheets**	**Personal Database Software**	**Enterprise / Multiuser Database**	**COTS**	**SAAS**

Ease of use (minimal training required)	High	Moderate-High	Moderate	Moderate-High	Low	High
Financial cost (initial capital)	Low	Low	Moderate	Moderate-High	High	High
Financial cost (maintenance/annual fees)	Low	Low	Moderate	Low	Moderate-High	Moderate-High
Support for data integrity constraints	Low	Moderate	Moderate-High	High	High	High
Ability to manage relationships between data sets	Low	Low-Moderate	Moderate-High	High	High	High
Searchability of the data	Low	Moderate	High	High	High	High
Ability to integrate with other data management systems	Low	Low-Moderate	Low-Moderate	High	Moderate	High
Customizability allowed by system	High	Low-Moderate	Moderate	High	Low-Moderate	High
Ease of information access	Low	Low-Moderate	Low-Moderate	High	High	High
Disaster recovery/backup	Low	High	High	High	High	High

Also, it is worth noting that paper files may still be required by policy or legislation for archival and document retention purposes. Even if your organization implements a full-featured digital data management system, your agency requirements may still mean you're maintaining a filing system of paper files. Situations like this will likely necessitate strategies to integrate your filing system with your database(s), particularly if original documents need to be retrieved alongside digital data queries. One suggestion would be to find or create an appropriate field in your database that states the location of the original document and configure reports that list all the necessary documents to be physically retrieved with each query. This will require meticulous filing and organization of physical materials, which can be arduous and time-consuming, but that's one of the intangible costs of dealing with any paper-based system.

A hybrid approach is to scan and save digital archives of paper documents. This can offload a lot of work and physical space, and also allow for some rudimentary search capabilities if the proper metadata are captured upon scanning or via optical character recognition (OCR) software. There are commercial, off-the-shelf (COTS) document management solutions (sometimes referred to as enterprise content management [ECM] packages) that can assist organizations with this process, but regardless, special attention to filing and organization of these documents is as critical as it is for physical paper.

Physical Paper Pros:

- often can be implemented with very little effort or IT involvement
- low initial cost
- for small-scale data collection efforts, this may offer the optimal solution
- where regulations require retention of hard copy original documents, integrating paper files into an electronic EDMS may be required

Physical Paper Cons:

- Paper data storage lacks many of the benefits of electronic EDMS—searchability, ease of summarization and reporting of data, public accessibility, portability, data backup and redundancy, etc.
- While the initial cost to set up can be very low, efficiency costs when meaningful data analysis is required can be immense, and dealing with paper can be labor-intensive.

- As data volume grows, so must physical space for filing cabinets, which may not be available.
- **This approach is not recommended as a sole data management solution unless the project is extremely small in scope and size.** Even for a hybrid solution where documents must be maintained alongside electronic data, digitizing paper via scanning and OCR is highly recommended if budget allows.

4.1.1 Other Forms of Nonstructured Data

The focus of the rest of this document is on structured data—data that is organized in a defined schema—typically represented by an arrangement of rows and columns. Examples would include a laboratory EDD with analytical results arranged by sample identifier and analysis; a list of locations with latitude-longitude coordinates and various metadata such as location type, county code, etc.; or a field activity log with activity notes separated by time stamps. Each of these can be arranged in a rectangular table and easily placed in a database, spreadsheet, or other structured model. Data elements can be separated into rows and columns, and functions such as searchability and fast data analysis can be done with relative ease.

Contrast this with other important data types, such as photos, audio files, video files, scanned paper documents in a PDF or TIFF format, etc. These types of files are often collected as supporting material to accompany more typical structured environmental data, but also in some cases are the primary data element. Regardless of the purpose, these nonstructured data types are often inherently missing critical metadata that give the file context and meaning. This lack of metadata can leave these data types without the crucial functionality of being searchable or categorized without further work to add those metadata elements. Hence, these nonstructured data types can be particularly challenging, but some of the same principles described above for paper files can apply to these nonstructured digital data types.

For instance, great care must be taken to ensure that these files are properly named and stored so that any associations between structured and nonstructured data are established and maintained. Some of the data management system types discussed below will allow for embedding these nonstructured media files as a “blob” file type, or alternately, a link to a file location can be embedded in a data field. Additionally, one must carefully consider the metadata required for the nonstructured data and establish those fields in the EDMS in a way that makes those nonstructured data searchable and informative to the end users. An example might be a series of photos taken at a sampling site in each of the cardinal directions—north, east, south, and west. Key information might include:

- location or sample identifier
- geographic coordinates (these may be embedded in the photo by default if using a geo-enabled camera)
- compass direction or azimuth that the camera is pointing
- any other details important to the study

A deeper discussion on the challenges of nonstructured data is outside the scope of this document. However, when considering an EDMS, it is critically important to consider whether the project needs will require heavy use of nonstructured data and ensure that the system is capable of handling those data properly. Some COTS systems have very useful tools that allow for automatic metadata population; many do not. Often a workable solution is to integrate an EDMS designed for handling structured data with an ECM system designed to handle nonstructured media files. This hybrid approach can be tricky if the systems are not properly configured to handle external connections, but it has been used successfully in several organizations.

4.2 Spreadsheets

Spreadsheets and other tabular data files have great utility when exchanging data or performing cursory data analysis. Their ubiquity makes them easy to use and highly accessible. Many spreadsheet systems have embedded formulas and built-in tools for data validation and analysis.

However, great caution must be taken if attempting to use spreadsheets as a “complete” system. For example, spreadsheets generally have a limited capacity to handle large data sets and appending new data to a spreadsheet may be a manual process of copying and pasting data. The ability to create relationships between two data tables is limited to the built-in formulas, which often rely on a single field to connect the data sets, without the ability to propagate (or restrict) updates and deletions. Consequently, relational integrity is difficult to maintain, the ability to query the data is minimal, and integration with other systems is limited. Structured Query Language (SQL) cannot be used to query data in spreadsheet

tables. Locally stored spreadsheets are editable by only a single user, which may cause issues with version control if multiple users are attempting to use the file at the same time.

Unlike most databases, spreadsheets allow users to easily examine the data visually. Visual examination of data, however, is not sufficient to identify all types of errors that may be present in a spreadsheet. Surveys of error rates have found that between 10 and 100 percent of spreadsheets have errors of some kind (Panko 2005; Powell, Baker, and Lawson 2009). The following types and frequencies of spreadsheet errors have been observed (Powell, Baker, and Lawson 2009):

- hard-coded references: 11%
- erroneous references: 48.8%
- logic errors: 31.7%
- copy/paste errors: 5.0%
- omissions: 2.5%
- data entry errors: 1.1%

Note that data entry errors, which might be thought likely to be responsible for most errors, constitute only a very small fraction of errors. The other types of errors can be more difficult to find than data entry errors.

Some guidelines for managing data in spreadsheets, which incorporate recommendations from Broman and Woo (2018) and Dunn (2010), are as follows:

- When creating a data table, have a clear understanding of what the table, and each row of the table, represents. Generally, there should be a table for each entity in the data model.
- Choose good names for columns and variables. Spaces and special characters are often troublesome if data are to be used by other software; underscores and hyphens are preferable. Use consistent capitalization.
- Put just one thing in each cell. Combinations of variable names, or qualifiers appended to concentration values, should be avoided.
- Make rectangular data tables without subheadings, subtotals, or merged cells.
- Use a single row for column headers.
- Store data in a normalized table structure, not a cross-tabbed structure. Use cross-tab data only for reporting (that is, in derived tables, not base tables).
- If using certain spreadsheet software, particularly Microsoft Excel, take steps to avoid the automatic conversion of some text into dates—see https://support.microsoft.com/en-us/office/stop-automatically-changing-numbers-to-dates-452bd2db-cc96-47d1-81e4-72cec11c4ed8. To mitigate this problem when importing data from CSV files, change the file name extension to “txt,” and in the import dialog in the spreadsheet program, choose appropriate data types for each column.
- Separate data and calculations as much as possible. Data in a tabular format should be clearly distinguished from, and separable from, cells containing formulas.
- If using spreadsheet software that can use different date systems, establish a standard date system for all workbooks before entering any data, and verify that data received from other sources use the same date system. See, for example, https://docs.microsoft.com/en-US/office/troubleshoot/excel/1900-and-1904-date-system.
- Do not use colors, typeface changes, or other formatting features to represent information. Important information should be represented by data stored in rows or columns of the data table, not by formatting, because formatting is lost if data are extracted for use in another application, and may no longer apply if the data table is re-sorted.
- Do not leave cells empty, or document a standard for the meaning of empty cells. Spreadsheets containing empty cells, just hyphens in some cells, and values such as “ND,” “NA,” and “–999,” may be interpreted differently by different people, and software that processes the table should be able to clearly identify instances of missing data.
- Ensure that some combination of column values uniquely identifies every row in the data table. This is an important corollary to the first recommendation: these unique column values should characterize the *thing* that each table row represents.
- Use named ranges, not hard-coded cell references, in all formulas.

- Create a data dictionary (for example, in another workbook sheet) that documents the meaning of column headings and any categorical codes used in the table.
- Related formulas should be in physical proximity, and the spreadsheet should read from left to right and top to bottom.
- Use the formula auditing tools provided by the spreadsheet software, or third-party tools, to check the spreadsheet for errors. Do not rely entirely on these tools; they can help find errors but will not find all types of errors.
- Use extreme care when sorting. A spreadsheet does not necessarily take row-based integrity into consideration, so a casual sort on one or more columns may not affect every column in the table. This can be prevented by selecting all columns when performing a sort operation, but it is easy to neglect this and introduce serious errors in the process.

Spreadsheet Pros:

- Accessible to most users without additional investment in software. Inexpensive even if not already available.
- Not difficult to find staff with adequate skills in using spreadsheet software.
- With some work and care, can be used to create a robust system using embedded formulas and other tools for data validation. Editing access may need to be restricted to staff who understand the procedures necessary for maintaining data integrity.

Spreadsheet Cons:

- Requires extra care and diligence to maintain data quality and consistency.
- Does not scale well to large data sets.
- Desktop spreadsheet software is not multi-user: different users cannot edit the software simultaneously. Some cloud-based systems can be used to mitigate this drawback.
- Desktop spreadsheet files are not very suitable for use with software version control systems. Procedures for versioning or backup of superseded files should be established.
- Limited ability to integrate with other systems.
- Appending data is frequently manual copy/paste operation, decreasing efficiency and increasing potential error rates.

4.3 Personal Desktop Database Software

This category refers to desktop software that provides a graphical user interface (GUI) to a database system that is stored in files on an individual hard drive rather than a server. Examples are Microsoft Access, OpenOffice Base, LibreOffice Base, SQLite/SpatiaLite, Kexi, and Filemaker Pro. Although not as full-featured as a client-server database, these programs offer a user-friendly interface that can often serve an organization's needs for data management at a relatively small scale. Both versions of Base and Kexi can also be used as front-end programs for client-server databases as well as standalone desktop programs, affording users a standard interface in an environment that uses a hybrid of desktop and client-server systems.

These programs are more complex than spreadsheets, requiring additional knowledge and training to properly use them. However, this additional complexity brings with it the advantages of using a relational data model designed with some of the principles laid out in Section 3 above. With these desktop database products, a database can be set up with relationships between multiple tables (that is, location, samples, measurements) and checks for uniqueness to avoid duplicate records. Queries may be easily established off these tables and relationships to allow for efficient and repeatable data reporting.

Cost and accessibility are usually positive features in this category. Often these programs are included and bundled in with office suites (for example, LibreOffice and OpenOffice both include their version of their Base product, higher tiers of Microsoft Office packages include Access, and the Calligra suite includes Kexi.) This can also mean that these programs offer relatively tight integration with other products in the office suite, such as spreadsheet programs and word processing software, which can help with ease of use and adoption among a wider array of users.

Although generally similar in many ways, there are some differences in features among these systems that may be important for some users or uses. For example, of the systems listed in the first paragraph of this section, all but

SQLite/SpatiaLite include a visual query designer. Also, as noted previously, only Base and Kexi can also link to client-server databases. Of these systems, only SQLite/SpatiaLite supports a data type specifically for geographic data, and SQL functions that allow manipulation and analysis of geospatial data in the database; in addition, common commercial and open-source geographic information system (GIS) software can connect directly to SpatiaLite databases. This list of contrasts is not exhaustive, and users considering adoption of a desktop database management system (DBMS) should carefully evaluate the features of all candidate systems.

Desktop database software does have its drawbacks, however. As indicated by the name, these programs are designed for databases that live on a user's local hard drive and are typically available for single-user-only functionality. Even if database files are placed on a shared drive (on an intranet or cloud-based service), multiple users are not able to access the data at the same time and data editing may be restricted. Occasionally, these kinds of conflicts can lead to data corruption or data loss. These data locks can be challenging to work with if multiple users need to access the data. Depending on the software, there can also be limitations to table/file size, and performance is often not optimized for large or complex data sets. If your data are particularly complex or are expected to grow beyond approximately 1 million records, you may want to consider moving up to a client-server or COTS solution.

Desktop databases can ease the learning curve of database design fundamentals considerably by adding GUI elements to perform certain operations that require writing SQL code in an enterprise relational database system. For instance, there may be a button to designate that a field in a table is a primary key, or certain table joins can be represented by a line between tables in a visual query designer. However, there is not a magic button that will automatically design a database data model for you to handle the types of data your organization needs to manage. Those skills require some training in database fundamentals—and preferably experience—to craft an appropriate data model for your needs. This document provides some very helpful hints with respect to the specific needs of *environmental* data management, but it is out of the scope of this product to provide basic database fundamentals. Luckily, there are many resources available on the internet, in bookstores, and from vendors of these products to help with staff training.

Desktop Database Pros:

- Can perform many of the advanced operations of a relational database without deep knowledge of SQL.
- Easier to establish relationships between different data tables.
- The entire database can be easily copied or transferred to other parties or locations.
- Enforcing uniqueness and basic data integrity checks can be set up with some database design considerations.
- Data exports can be customized by setting up queries that are repeatable and can scale with your data.
- Cost of the software is relatively low, and this solution can be ideal for smaller data sets.
- While not as easy to use as spreadsheets, desktop databases generally have easier learning curves than a client-server database and do not require highly specialized skills to maintain.

Desktop Database Cons:

- Single user only.
- May run into file size limitations, which can lead to problems as data sets grow.
- Lack of features for automatic maintenance of data integrity that are found in client-server systems, such as a variety of cascading update and deletion options, check constraints, and triggers.
- Although they may integrate with other desktop software, desktop databases typically do not offer the data sharing and scalability features found in enterprise client-server database systems.
- Performance is typically less than client-server databases as a result of (typically) less powerful hardware and less sophisticated query planners.
- Although much of the SQL code is abstracted in a GUI, knowledge of the principles of database design and usage is still required.
- Usually, will need to design a schema that fits the needs of the data/project.

4.4 Client-Server Database Systems

Client-server database management systems allow many users to access the data simultaneously and provide the maximum customizability for the implementation of data models. These features set client-server DBMSs apart from other systems

described here. In addition, client-server DBMSs support the creation of primary keys; foreign keys with different ways of handling cascading updates and deletions; check constraints; and triggers. These features allow creation of a database in which data integrity is automatically maintained (see the Data Quality fact sheets, particularly Using Data Quality Dimensions to Assess and Manage Data Quality). Better features for maintenance of data integrity also sets client-server databases apart from other systems.

Another advantage of client-server DBMSs is that the server hardware may be much more powerful than the computers of individual users. Performance of data selection and summarization operations can therefore be considerably higher than those operations would be if they were performed on an individual user's hardware. The speed of transmittal of data from the server to a client computer over a local network or the internet may be the largest constraint on performance.

The server component of a client-server DBMS can be installed on an organization's internal network (on a physical or virtual machine) or on an external network (that is, in the cloud). The former approach provides greater security because the database is inside the organization's firewall. The latter approach provides greater accessibility to the organization's own staff, clients, contractors, and the public. Costs for these two approaches will differ depending on the support that can be provided by an organization's own IT staff and the features and tools provided by the cloud hosting service.

Client-server DBMSs do not necessarily need to be installed on separate server hardware. They can also be installed on a user's own machine and used as a personal database system. This approach loses the advantages of multi-user access, but also eliminates the effect of network transfer time on performance.

Several widely used client-server DBMSs are:

- PostgreSQL—open source (https://www.postgresql.org/)
- MariaDB—open source (https://mariadb.com/)
- MySQL—open source (https://www.mysql.com/)
- Microsoft SQL Server—commercial, but with a feature-limited version available at no cost (https://www.microsoft.com/en-us/sql-server)
- Oracle—commercial, but with a feature-limited version available at no cost (https://www.oracle.com/index.html)

Management and analysis of geospatial data within the database is supported by PostgreSQL (with the PostGIS extension), Microsoft SQL Server, and Oracle. Capabilities differ between these systems, so if geospatial data storage and analysis is important, the features of these systems should be assessed with respect to the program or project's specific requirements. This specialized feature is available only in enterprise-level DBMS packages such as those mentioned above. These features are not currently available in data system types mentioned previously in Sections 4.1–4.3.

These DBMSs also differ in the complexity of the data models that they can support. For example, some systems support multiple direct or indirect foreign key pathways between tables, and some do not. If a table contains one column for the units of a reference surface (soil, water, or sediment) elevation measurement, and a second column for the units of a sample depth measurement, both of those columns should have a foreign key to a table of valid values for units. This type of structure is not supported by all DBMSs.

Some types of client-server DBMSs have features that allow one database to directly access data in a different database, even a different database that is running under a different DBMS. These features support hybrid or transitional approaches to data management.

Many user interfaces (client programs) are available for client-server DBMSs. Some of these are specific to a particular type of DBMS, and some can be used with many different types of DBMSs. Different types of user interfaces are available, including:

- GUI programs for querying and database management. These ordinarily allow entry of SQL commands and display the output. Most include features to import data and to export data in various formats. Some include visual query builders with a drag-and-drop interface to select tables and columns to be included in the query.
- Script processors. These read SQL commands from text files and send them to the database. Most script processors include some specialized commands to import and export data and perform database management functions. Using script files and script processors allows database operations to be integrated into a tool chain for automation of complex data-intensive processes. Scripts also are themselves a record of operations

performed, can include documentation, are subject to software quality review, can be put under version control, and can be used as a basis for other scripts.
- Programming languages and programmable analytical software. Languages such as R and Python include libraries that allow direct connection to client-server databases, execution of SQL, and capturing of the output for analysis. Programs such as these have the same advantages as SQL script files, and enable more direct integration of data queries and data analyses.
- Spreadsheets. Some spreadsheet software can connect to client-server databases and import data.
- GIS software. Both open-source and commercial GIS software can connect to widely used client-server databases and recognize specialized data types used to store spatial data.
- GUI analytical software. Most interactive data exploration and analysis software can connect to widely used client-server databases. Examples of such commercial and open-source software include Spotfire (https://www.tibco.com/products/tibco-spotfire) and Orange Data Mining (https://orangedatamining.com/).

Depending on the requirements of the program or of users (see Section 2 above), additional software may need to be developed that is tailored specifically to the data model and to those requirements. Although high-quality open-source software is available for the DBMS(s) and for client programs, leading to relatively low costs, a need for custom software may substantially increase those costs in either the short or long term. Use of client-server systems may also require time from IT staff to maintain the hardware, a database administrator (DBA) to maintain the DBMS software, and a developer or application systems specialist to maintain or customize the data model implementation and related software.

Client-Server Database Pros:

- Multiple users can simultaneously access and edit data.
- Users can connect from multiple physical locations.
- Customizable.
- High performance.
- Scalable.
- Highest level of support for automatic maintenance of data integrity.
- Spatial data storage and analysis features are available in some DBMSs.
- Linking of multiple databases, with some DBMSs.
- May be low cost.
- Multiple general-purpose client programs are available.
- Most specialized analytical software can connect to client-server databases.

Client-Server Database Cons:

- A DBA and possibly other support staff will be needed.
- Specialized knowledge/training will be needed for implementation of a custom data model.
- Custom interfaces may need to be developed to meet some requirements.

4.5 Commercial Off-the-Shelf (COTS) Database Systems

COTS database systems are client-server database systems that have been developed by a software vendor for commercial sale. Although the COTS database systems use the same DBMSs for the backend as client-server database systems, they have vendor-developed data models and proprietary GUIs to standardize user interactions, with options for customization. However, because the underlying DBMSs are typically the same as the client-server database systems, the other user interfaces, such as programming languages for scripting and GIS software described above, can also be used if the vendor allows access to the database back end.

The physical server hosting the COTS can either be local, on the same network, or in the cloud—either a vendor-managed cloud or one set up by the purchasing organization. This flexibility allows users to choose the server option that works best for their potential implementation. Remote hosting of the server on a cloud environment, although incurring an additional cost for upkeep, can have several advantages over hosting the database locally or on a physical server within a network. The remote server is not impacted by local events, such as power outages or severe weather events, and is backed up off site to ensure the data are stored in multiple places. Remote hosting also simplifies the security associated with the

database and allows multiple users to log in from many different physical locations without worrying about internal firewall issues. However, there may be IT policies within your organization that dictate whether the cloud is an option at all. Please consult with your IT department closely while considering hosting solutions.

The commercial nature of the vendors that produce and manage COTS database systems means the upfront costs for purchasing and operating these systems can be significantly higher than the other types of data systems described previously in this document. There are not only upfront costs associated with purchasing the COTS database system, but also ongoing costs for user licenses, maintenance fees, and, potentially, hosting fees if a cloud-based remote hosted server is used. However, these expenses also include access to vendor-provided customer support and troubleshooting, including regular updates to the software to add features and respond to security threats, should they arise.

If deciding between client-server and COTS solution, be mindful of total return on investment (ROI) and hidden costs. Sometimes COTS can be more expensive up front but less expensive long term.

Involvement of the software vendor can also increase standardization of both the database schema and the methods for submitting data to the database. External parties submitting data to the COTS database system can use standardized database submittal formats (see also the Electronic Data Deliverables and Data Exchange Fact Sheet) that may be usable with any instance of the COTS database systems. The vendor can create custom submittal formats for individual databases as needed, and these are able to be shared with external data providers.

Many COTS database systems include not only the custom GUI front end for use but also a robust set of additional tools to help manage, store, and analyze the data. These additional tools can range from programs that ensure reference values are correct to GIS and graphics creation packages. The standardized nature of these systems also ensures that outputs from the COTS database systems can be used in other external programs without much customization or remapping of the output required.

These systems are readily expandable whether created internally or created by another organization; consequently, storage space is an issue that is readily fixed by the agency or the development contractor.

COTS Database Pros:

- Most of the advantages of client-server databases listed above also apply to COTS, since most commercially available systems use one of the major relational database management system (RDBMS) packages as a “back end.”
- COTS systems will have established schema and workflows built around a typical sampling and analysis model that most environmental organizations will use. This eliminates the need to develop a data model from scratch (provided that the COTS model fits the organization’s needs.)
- Vendor customer support and frequent updates.
- Technical support for bug fixes.
- Provides standardized inputs/outputs.
- Robust security if patched in a timely manner.
- Cloud servers (if applicable) typically more redundant than on-premise installations.

COTS Database Cons:

- Up-front purchase costs, and possible ongoing costs for support and hosting (if applicable).
- Limited customizability.
- New features are based on the vendor’s schedule and not necessarily the customer’s needs. Some vendors are more responsive to customer requests than others.
- A DBA may still be needed, if hosting on-premise.
- May require trained staff to act as “power users,” since these systems can still be quite complex to manage.

4.6 Software as a Service (SAAS)

Software as a service (SAAS) systems (for example, Salesforce and Microsoft Dynamics) are increasing in popularity due to the ease of configuring and making changes to the system without having dedicated IT staff. Major platforms also have

dedicated user groups that share custom developed configurations for rapidly implementing improvements. SAAS systems are low-code or no-code, meaning that they do not require any knowledge of computer programming or scripting languages to build, maintain, and use the system. The system flexibility does require higher up-front implementation costs and the cost of an ongoing subscription to use the platform. Data are hosted by the system provider but can be exported at any time using reporting tools built into the system. Migrating data from these systems can be challenging because the data are not stored in structured tables and can be difficult to map to another data structure. Most systems offer integration with other popular software packages (for example, Esri ArcGIS, Microsoft Office, Google Workspace).

SAAS Pros:

- Customer support.
- Technical support for bug fixes.
- Software updated several times a year.
- Robust security.
- Many users can simultaneously access and edit data.
- The system is expandable based on usage, although price is typically per user.
- Customizable. The data model does not need to be defined in advance.
- Easily shared with users outside the system such as the public.
- Supports many data types, including non-tabular data, audio/video.
- Easily integrated with email and other systems for notifications and transferring data.
- Does not require dedicated IT staff if support and development are outsourced to the vendor.
- Users can create forms and reports without vendor or IT staff assistance.

SAAS Cons:

- Expensive and requires ongoing subscription. Per-user cost, so cost increases with increased user base.
- Subscription-based. The system is available only while the paid subscription is active.
- Difficult to transfer to other systems if the subscription is terminated.
- Can be difficult to develop if data model is not defined.

4.7 Other Systems and Conclusion

In the sections above, the authors have considered six general types of EDMS that are most likely to be used by states or other medium to large organizations. These six types are not meant to be an exhaustive list, and there are certainly other ways an organization can successfully manage data. One example that quickly comes to mind is using a GIS to manage data. Often many GIS packages use geodatabases as a back end for mapping functions, and those can be leveraged for their ability to perform key database operations such as establishing primary key/foreign key relationships and SQL joins. What a GIS may lack as an out-of-the-box feature is automated loading of nongeographic data such as laboratory EDDs—something that a well-designed EDMS will typically do with relative ease. (For more information on handling geographic data in the context of an EDMS, please see the Geospatial Data work products.)

Regardless, the strengths and weaknesses of *any* system must be evaluated against the specific needs of the project to ensure that data integrity and quality are maintained. Does the system have the ability to input data from a variety of different sources—field, laboratory, third-party consultants? Can the system easily generate reports to meet the needs of project managers, decision makers, or the public at large? Finally, evaluate the totality of costs involved. While the up-front expense of a larger, more complex database system described in Sections 4.4–4.6 in this document may be high, one must also consider the opportunity and efficiency *losses* by continuing to use a less robust system such as spreadsheets or desktop database software. This decision is going to be individual to each organization and, with so many considerations, likely will not be easy. The authors hope that the information and discussion laid out in this fact sheet has helped to clarify some of the technical details around these systems and clarify the decision-making process.

5 REFERENCES

The references cited in this fact sheet, and the other ITRC EDM Best Practices fact sheets, are included in one combined list that is available on the ITRC web site. The combined acronyms list is also available on the ITRC web site.