20.3: Planning the data flow

There are many advantages to collecting and storing research data electronically. Electronic storage of data facilitates easy retrieval, simpler generation of study reports, easy exportation to statistical packages, and rapid data sharing. The benefits of electronic storage of data can only be fully realized if the database storing the data is well designed. A poorly designed database leads to poor performance, inefficient data queries, inaccurate and unreliable data, and redundant data that are duplicated in many places, making it difficult to check and clean. This section focuses on the key aspects of the processes in the data flow.

3.1 Database design

Database design is the process of organizing data in such a way that it can be stored and retrieved efficiently. It involves making decisions on how best to model a real-world information system, such as a paper-based data collection system, into a database. It is very unlikely that an analyst can correctly design a system without a full understanding of the key processes and activities involved in the study. This requires researchers spending time with the system developers to ensure that the system developed is what is required.

It is good practice to use a structured approach, referred to as a system development life cycle, when undertaking a database development project. The choice of the methodology is usually influenced by factors such as the complexity of the proposed database, the size of the database and the programming team, cost, time, and criticality of the project. What is important is to get an approach that meets the needs of the project. An overview of the key phases involved in database development is presented, rather than focusing on a specific methodology. These procedures should not been as checklists, but rather key processes that can be incorporated in any methodology chosen. The key procedures are:

1. project specification
2. requirements gathering
3. programming and testing
4. database implementation (‘going live’)
5. database maintenance and change management.

The objective of the next phase is to transform the high-level requirements into more detailed manageable tasks and functions that can be programmed into a software system. The requirements can be gathered by interviewing end-users of the proposed new database, examining the current database, if any, and also looking at existing forms such as questionnaires and reports. It is important to think of what extra functionality is required in the database. Will the data be shared? If so, which specific data will be shared? What are the security and compliance requirements? Have risks been assessed, and the database designed to mitigate the risks? The output of this process is a detailed requirements specification document, describing all the functional and non-functional requirements of the database. It is imperative that requirements are specified correctly and as comprehensively as possible at this phase; otherwise, it could lead to a system that does not meet its intended goal and may necessitate major changes during programming and testing or after the database has gone live.

The third phase involves creating a conceptual design (logical diagram) that shows the different tables that will be required to store the data identified in the first two phases. A good place to start from when generating the list of possible tables and their attributes (data columns) is to look at the current process, if any, used to collect data. There are two possible scenarios—an existing computer system is being converted or modernized or a new system is being built from scratch. In the former scenario, the tables and data entry forms of the existing computer system should be used as a starting point. If there was never a computerized system in place, begin with the existing paper-based data collection forms. If there are none, sketch out the forms, based on the requirements specification document, and discuss the sketches with the research team, and refine them further. Note that, while some of these data entry forms are sketches of what will eventually become data entry screens, others will properly remain in the realms of paper forms and will not necessarily map directly into data entry screens. If new requirements arise, while the conceptual design is being created, add them to the list of requirements that was created in the earlier phase. While sketching out the tables, also review the list of existing and new reports to establish a reasonably definitive list of the reports that the system must produce if it is to satisfy the needs of the users. The objective of analysing the reports is to ensure that the tables sketched out will have all the attributes that are needed to generate the reports. If there are missing attributes or tables, they should be added now. The completeness is important, but it can sometimes be difficult to know if all the reports that are being, or will be, produced or used have been identified. The database developer can proceed to create the physical database when the team considers that the requirements are
sufficiently comprehensive.

After the database has been created, a programmer designs the data entry forms and links them to the tables in the database. There are various types of software that can be used to create the electronic data entry forms to capture the data. The choice of the tools and programming language will depend on the technical skills and preference of the team. When the programming phase is done, the database application should be tested. It is recommended that someone other than the programmer who developed it tests the application. Testing is an important phase, because it ensures that the system is validated and verified, a major requirement for GCP compliance. Users have to be trained on how to use the database, before deploying it for actual use. The database application should supplement user training by providing help features where users can access help through the application.

3.2 Data cleaning and integrity

Data cleaning should be an ongoing process, rather than something that is done at the end of the study. The process by which the data will be cleaned should be well thought out, planned, and documented at the beginning of the project, and certainly before any significant volume of data has been collected.

Double data entry is commonly used to minimize data entry errors. In this technique, two different people enter the same record independently, and the two entries are compared against each other. A validated data record is one where both entries are the same (see Section 5.1). It is important to remember, however, that no data entry system can avoid errors that were made by the interviewer using a paper-based questionnaire to record information in the field.

The database application can be programmed to flag inconsistencies in records, either during or after data entry. One approach is to categorize errors as being critical and non-critical. A critical error is one that is so important that systems are put in place to ensure that the data record cannot be saved into the database until the error has been fixed, for example, lack of the respondent’s identity code or this being out of the valid range, as this code will be needed to link information in the database. The programmer would write these as checks embedded in the database application, sometimes called ‘online checks’. The downside to having too many such checks embedded into the data entry screens is that the users cannot save the data until all the errors have been fixed, which can lead to back-logs. Decisions about what errors will be critical and non-critical should be made early enough, so that these are programmed in the system. Non-critical errors should not stop the user from saving the data record. They would instead be flagged up as data queries and reports for the data manager to follow up, rectify, and update the database.

Another approach is to incorporate data checks in a statistical program, for example, Stata, and run the checks against the data periodically. In the case of a paper-based collection system, periodic monitoring visits can be made to ensure that SOPs are being adhered to. Further checks can be done by taking a random sample of paper forms and comparing them against the corresponding electronic data records.

3.3 Programming issues

Computerized data collection systems are driven by computer programs written by system developers. The resources used to develop these systems can be made more effective if good programming practices are used. Computerized data systems should be documented to a sufficiently detailed level, so that any other system developer could quickly take
over the maintenance or extension of such a system. A poorly documented computerized data system makes it very
difficult to make changes to the existing system, and it will take a second person longer to figure out what needs to be
changed. Even the programmer who wrote the initial program may forget specific technical details after a few months.
Investing the effort to document programs, as they are developed, makes them easier to maintain, and changes can be
made much more quickly.

Prototyping is an iterative technique used in computer systems development where the programmer designs mock-ups
and asks the user to try them out and give feedback. The advantage of prototyping is that the users do not have to wait
until the system has been fully developed, before they can try it out.

3.4 Standard operating procedures

SOPs are a set of written instructions detailing how a particular process is carried out. Computer-based systems support
trial processes by providing a means of storing, modifying, and retrieving data. The process by which these computer
systems are developed and used should be documented and controlled by procedures (SOPs) that ensure that they are
adequate, and, where necessary, GCP-compliant. SOPs allow different people to check the procedures and ascertain
whether what is done corresponds to what should be done. SOPs are also invaluable for training different people in the
tasks that need to be undertaken in the study.

It is usually a good idea to split the SOPs into different categories such as database development, database validation
and testing, database implementation and site set-up, and database maintenance/backups/upgrades. SOPs should be
written by the person responsible for the task (who knows what should be done) and checked by the person who
supervises their work and finally approved by the study PI.

3.5 Version control

The purpose of version control is to keep track of changes made to a computerized system during its development and
after it has been implemented. The changes can come from various sources. For example, the users may find errors
that need to be fixed when they start using the live system or may request new features or improvements to the system.
Also, a change in environment may require a change in the computer system.

For example, a decision to move from Microsoft Access databases to SQL Server databases will require changes to the
data entry screens. Another example is a new compliance requirement that requires a certain type of report to be
generated by the system.

A requirement for GCP-compliant data management is the use of validated and verified computerized data collection
systems. Systems are validated and verified by thorough testing, comparing the database system against the user
requirements. A validated and verified system is one that meets its specifications and requirements and that fulfils the
purpose it was created for. Any change made to an already validated computer system may introduce new errors.
Hence, the process of making changes needs to be done in a controlled environment. Previous versions of the program
code are stored in a version control software, for example, Visual SourceSafe, Subversion. The system ought to be
tested after making changes to ensure it remains in a validated state, before deploying the updated version. The
detailed process for managing and maintaining changes to the system should be written in a version control and change
management SOP. It is important that the SOP is adhered to strictly.

### 3.6 Confidentiality

Information that can be used to identify a person should be stored in a secure database that allows only authorized persons to access the data. Any data that can be used to identify a person, for example, name, address, date of birth, should be kept out of the public domain. Sensitive information should be identified from the onset, so that appropriate controls are put in the database. If the data are to be shared, it is necessary to decide how this will be done and what kind of security checks will be put in place. Technical security mechanisms, such as audit trails, access control using user logins and passwords, and permissions should be supplemented by data-sharing contracts and user training. Encryption should be used when sharing or carrying data on portable devices to ensure that unauthorized users cannot read the data, even if they get hold of the portable device.

### 3.7 Training

However basic the database system may seem, users should be adequately trained and should fully understand what they are doing. This training may be in the form of professional and in-house training and may involve using a prototype of the database in a pilot scheme. User training logs should be kept as evidence of training.

### 3.8 Pilot testing and database testing

User acceptance testing and pilot testing are commonly used to verify that the database performs well. In user acceptance testing, the end-users test the new database by entering data, following the SOP, and trying out the functionality provided by the database. The end-users feedback comments to the programmers and study leaders, who can make the necessary changes to the database programs and to the SOP that define how the procedures work. This is very useful, since database issues are identified early and rectified, before data have started being captured. Pilot testing also helps to identify potential issues that may arise when the study systems go ‘live’.