This page is intentionally left blank.

Chapter 15

Database Operation and Management

Donagon wiped his damp hands, opened the journal and began . . . .
We decided not to abandon the attempt after all; to try once more to store a man digitally. The last obstacle had been removed, i.e., storage. Previously we had estimated that many thousands of miles of magnetic tape would be required, with complex retrieval problems. The multiple storage paired redundancy tapes, developed by Müller-Fokker in Vienna. . .

Every datum will be recorded many times, to reduce error. At present surgeons are removing tissue samples from the subject and determining cell structure data. We have already encoded a DNA map, photographs, holographs, x-rays, resin casts, EKG's and so on. There remains but one step, the mapping of all electrical and chemical activity of the subject's brain. The press will be invited to this session . . . .

From *The Müller-Fokker Effect*, a science fiction story
By *John Sladek*

The benefits of database systems are obtained only after the design and the implementation are completed, and after sufficient data has been collected so that the user of the database can receive usable information. A large commitment of effort is required to attain that stage. The remainder of the database cycle will involve continuing assurance of reliability and quality, periodic adaptation to changing requirements, and eventually termination of the operation with transfer of valuable data and procedures to a new system.

This final chapter will consider these issues and in the process summarize some of the observations from earlier chapters.

15-1 DEVELOPMENT OF THE DATABASE

The development of a substantial database operation is invariably a task which requires the cooperation of a number of diverse individuals. The collection will include a large number of potential users, data-analysis specialists, information scientists, experts in various aspects of computing, communication specialists, the future database administrator, and representatives from management, who will be investing resources of the enterprise in the effort.

It is not unusual for potential vendors to be included in the initial discussions. They can bring considerable experience to bear on problems, but their opinions have to be validated at sites of prior implementations. A good deal of knowledge but relatively little experience is currently available in the academic world. Experimental research on large databases is difficult because of the large cost and long time scales associated with database system development.

Objective Setting The determining components of a database operation are the users and the data. The users determine the objectives of their applications and the semantics of the data determines how the objectives can be satisfied. The type, activity, and quantity of users and data have to be quantified before any specific system design can be done. The determination of these parameters is a management responsibility. A group associated with a specific system is apt to view users and data needs according to its capabilities, rather than according to the underlying user objectives. Documented feedback to management is required when imposed service demands are excessively costly. A constantly recurring problem is that programming staff venture opinions about cost and effectiveness of alternatives without analysis, often using a peremptory approach, sometimes basing opinions on personal interest.

In order to get a development group composed of diverse individuals moving in the same direction, a definition of the goal objectives is required. Both lower and upper levels of the operational parameters to be achieved have to be specified. Some quantified limits of system performance objectives for an application may be:

Response Time: 90% of single-element queries are to take less than 1 s between query entry-completion and begin of response.

Backup: Backup is to be available for all source data which have been entered more than 3 h ago. Backup is to be retained for all data which were deleted less than 3 months ago.

Deadlock: Less than one deadlock per year should occur.

Cost: A single element query should cost less than $1.00. Successive data elements to be obtained at less than $0.05 each.

Size: Capability to access 100 000 to 2 000 000 records of size as defined.

Objectives defined and understood at this level form the basis for mutual decision making and have to precede the exploration of the system alternatives. Objectives stated without restraint, as *instantaneous response, absolute reliability, perfect protection of privacy*, can lead to an unbalanced effort in system implementation.

Binding A decision criterion which appears throughout this book is the concept of binding. Binding measures the extent to which flexibility has been traded for efficiency. We have identified binding choices in file structures, in database models, in schema management, in information retrieval, and in the maintenance of data and integrity protection.

If much flexibility is required, the system will be loosely bound at many levels: the record structure will be variable, the access paths will be easily augmented or deleted, the schema will be able to accept changes, and the query-processing system will be able to interpret changes made at any time. If performance is important, constraints will be imposed on some of the levels, but at some other levels flexibility should be retained.

The concept of binding can promote a common understanding among the people involved in the system development process. A general problem in database design is that it involves the bringing together of people from many disciplines, both as users and as implementers.

Documentation In a database the most important documentation is the database model. The database model will determine the processes required for file creation, file maintenance, and information retrieval. The schema, expanded with notes about real-world relationships, connection constraints, and the definitions of the variable domains and their representation becomes the formal repository for the database model documentation as the design is refined.

The technical aspects of the design process were described in Chap. 5. When the basic system choices have been formulated, more detailed specifications have to be developed. There will be many interfaces among the system components (see, for instance, the ANSI SPARC model of Fig. 8-16) which all require careful documentation. Human interfaces are difficult to visualize by documentation alone, and examples or pilot operations may be desirable. Particular care is required for interfaces using visual display units or graphics if their full power is to be exploited.

The extent and detail of program documentation required will vary according to scope and complexity of the system. If higher-level languages are used, an external description of the function of a program module, and detailed documentation of all variables used, is often adequate. Flowcharts have been of value where multiple processes interact.

A common vocabulary is important. In the beginning of a project, especially if multiple outside vendors are involved, terms will have a variety of meanings. People who recognize the problems will begin to talk in noun pairs: CODASYL-sets, LEAP-relationships, etc.; those who have read less will be confused. As the specific system design stabilizes, so does the vocabulary. During the development of this book much effort has been made to assure that definitions are consistent throughout, and most of the terms have been acceptable to the readers of the first edition; some definitions have been revised. Vocabularies adapted to a particular system will continue to be used, but a table of definitions, which may cross reference other work, such as CODASYL, will be helpful to outsiders.

The design of a *generalized database management system* is made much more complicated because of the absence of user objectives and specific database mod-

els. In practice certain types of users are postulated and artificial models are constructed. As was discussed in Chap. 10, there are some underlying concepts of an information methodology which can be applied when a new system is being considered. Since no system can be everything to all users an explicit boundary of the problem area to be addressed is required if conflicts during the design and the complexity of implementation of a generalized DBMS are to be kept tolerably low.

Programming Techniques of structured programming promise to aid in the construction of large programs. It is interesting to note that seminal work in this area was applied to information systems: the *chief-programmer* approach at the New York Times Information System [Baker[72]] and *formal modularization* for a KeyWord-In-Context generator [Parnas[72]].

The management of a critical section of code requires tools to manage locking mechanisms. The fact that locks transcend structural levels (Sec. 13-2-2) makes locks error-prone. A large amount of programming effort can be wasted if these issues are not addressed prior to implementation. The trade-offs among integrity control, performance, and flexibility in the applications are not well understood by most programmers. Papers that promise that these problems have been solved have misled many uncritical readers.

The separation of data structures from programs is another tool which is important in the development and maintenance of systems. The use of schemas provides this facility for the database, but similar facilities are also useful for access to other shared system resources.

The use of a top-down approach to program design is feasible when there are reasonable expectations about capabilities at lower levels. This book presented the material of the database world in a bottom-up fashion, so that at each step up it was possible to verify that the ground below was solid. Abstraction of alternatives at the lower level are the building blocks of the higher levels; these abstraction have to be based on realizable constructs.

15-2 MAINTENANCE OF A DATABASE

When a system is completed, adequately debugged, and populated with data, the real work begins. We stress again that the value of a database to the user is not in the database system but the data content and particularily, the results from queries entered by users and users' programs. Of course, a bad system can frustrate all other positive aspects.

There will be the need for modifications due to changes which came along while the system was being implemented. To avoid without serious disturbances changes are best not considered during the latter part of a development effort. Changes will be tested on a copy of the database so that ongoing use will not be affected until the changes are well checked out.

Then there will be the need for additional facilities or data, as users find that they cannot quite do everything they had expected. The gap between expectations and delivered product can be especially great for users that did not participate in the development effort. If participation in development is not possible, it is wise

to warn the user that the installation of a database system will not provide all the possible benefits. The delays which a user can expect will depend on the divergence of the needs with the architecture of the sytem.

Stages, seen once a database system has been developed, include:

Operational shakedown of the database system The hardware and software is tested using pilot data.

Functional shakedown of data entry procedures It is difficult to have reliable data entry while the system is not perceived to be itself reliable, but at that point we have to assure ourselves that the input flows smoothly and error free into the database.

Verification of data content The benefits of integration of data can be proved only if the quality of the data stored is perceived by the users to be as good as their local data collections.

Report generation Use of the database to generate reports suitable for the environment provides initial operational experience and feedback of data.

Statistical analysis of data content Where data represents observations, descriptive statistics and graphs can present the contents in a form which leads to questioning and further inquiry.

Model building As the contents of the database matures, it becomes finally the intended resource. Hypotheses can be tested and projections made which are based on past data. This is the stage where the system actually generates information data useful to decision making and planning.

The time required to achieve the final operational stage can be many years. The various stages require increasingly sophisticated personnel and tools. The investment in these resources will be made only if earlier stages have provided the confidence that the effort is worthwhile. Disappointment due to expectations of rapid or massive benefits can cause a loss of confidence, and may make a system, which works technically reasonably well, a failure.

15-2-1 Tuning and Monitoring

An adequate performance-to-cost ratio will be of constant concern to the system administrator. An administrator will be developing continuously tools to measure system productivity. It may be difficult for users to formulate reasons for dissatisfaction with aspects of the systems performance, so that lack of use can be a signal to look for a problem. Sensitivity to the needs of the users is important.

Nearly all performance improvements increase the redundancy and the binding of the system. Access paths are generally easier to manipulate than replication of actual data. A system which allows the creation of new access paths can be tuned quickly to improve fetch behavior. Improving transactions which access many elements tends to be more difficult and may require considerable data redundancy. Increased redundancy, of course, decreases update performance. Increased duration and complexity of updates increases the probability of deadlock.

It is clear that the solutions which seem expedient are not always desirable. A good understanding of which system resources are heavily used and which are lightly used is necessary to make the correct decision.

Among the most difficult decisions to make is the unbinding of a previously bound relationship. This can involve the splitting of a collection of tuples into two separate relations, either by attribute or by ruling part. In Sec. 7-5-2 we combined a `highschool_record` and a `college_record` into a single relation `education`. This may have made the model simpler and more consistent. Unfortunately, as a file gets bigger, the time to process the file increases more than linearly. If we take that processing time as generally proportional to $n(\log n)$, we find that processing the same file split in two requires less time, namely, $2(\frac{1}{2}n \log \frac{1}{2}n)$. This example of a case where bigger is not better can be demonstrated in other facets of database systems. In general, database fragments that appear not strongly related to each other can be beneficially separated. Such a divorce, if it takes place after the data have been living together for a long time, is apt to be traumatic, since it may reveal the existence of relationships which were not documented.

Distribution of databases is best based on a partitioning into fragments that are not strongly bound, since this will minimize autonomy and integrity problems. There are cases where a single relation in a model may be split into *horizontal fragments*, namely tuples that are used mainly at different nodes. An example is where the employees work at at different locations and their data are kept on the local node. `Vertical fragments` are created when attribute usage differs among nodes. The employees' skill data may be kept on the `Personnel` computer, while salary data are kept on the `Payroll` machine.

Monitoring Measures of system utilization that can be profitably monitored include:

Statistics of device utilization. The percentage activity times of processors, channels, controllers, and disks. It is desirable to have both average values as well as values which pertain to the busy periods of system operation.

Statistics of file utilization. A matrix of user process versus file activities is the basic measure. The relative ratios of type of access such as fetch, get next, and update are important for decisions about the file organization. File density (space used versus space allocated) is another measure.

Statistics of record utilization. The frequency with which records are accessed for read or update provides a measure which can be used to provide optimization in systems where records are chained. The dates and times of last access and update are important for the maintenance of integrity, as is the identification of the process which updated the records.

Statistics of attribute utilization. The frequency with which attribute values are requested, updated, used as keys, or used as further search arguments provides the data for optimal tuple membership selection.

The measures can be obtained by continuous monitoring or by selective sampling. The best repository for data obtained from monitoring depends on the system design. It can be difficult to bring all the measurements together for automated analysis. A regular logging tape provides the most obvious repository.

Data on device utilization may be provided by the operating system. File utilization data can be saved in a file directory when a file is closed. Record activity logging was essential for comprehensive backup, but many systems do not provide this service. Data on record activity can be collected with the record when records are written or updated,

but the cost of rewriting a record which has been read in order to collect access monitoring data is excessive. Attribute activity can be recorded with the schema, and recorded when the associated file is closed or when a process terminates.

The monitoring data listed above do not provide information regarding individual data elements but do provide activity data for the record and the attribute, which together describe the activity of any element in terms of averages.

The availability of monitoring information not only provides data for tuning. A data administrator can now also justify expansion when needed, and can provide data when there are service conflicts among user groups. The job of a database administrator is already difficult. Without tools it is impossible. A person willing to undertake database administration without tools will be self-selected on the basis of political adroitness rather than on the basis of technical competence.

15-2-2 The Lifetime of Data and Database Systems

Both data and the systems lose value with age. It is hard to predict what becomes obsolete first, so that both effects have to be considered.

Data Archiving As data ages it becomes of less value. Eventually its value will not warrant continued on-line storage. The low cost of dense off-line tape storage makes it possible to retain old data as long as there is any probability that the data will be of eventual interest. Optical disks promise new alternatives for archiving old versions.

The procedures used during transaction logging have produced extensive back-up capability, but the contents of the logs and checkpoints is closely related to the status of the system when the backup was generated, so that these files may be difficult to utilize. For long-term archival storage, it is best to generate tapes in an output format, which can be read by input procedures when needed. If the schema entries for the attributes saved on the files are also written on the archival tapes, much of the file documentation will be retained with the files. In large and very dynamic systems, i.e., very large databases, it may not be possible to generate adequate archival tapes from the database, since the database as a whole is never consistent. Then the archival tapes can be generated from the logging files.

If preservation of auditability is important, archive files can be generated continuously in a database system. A `database_procedure`, specified in the schema, can state

```
ON DELETE, UPDATE CALL archival_recording
```

so that archiving responsibility is assumed by the manager of the schema.

It is important to read archival files, first immediately after creation, and later at least on a random sample basis. This avoids the embarrassment of finding that one's archives are useless, either due to an error when they were generated or due to incompatibilities introduced as the system was further developed.

System Life Cycle It is very hard for a developer to visualize the point where the operation of a particular system will be terminated. Even where the database

remains of value, there will be a time when it is better to transfer the data and programs than to continue an operation which has become obsolete.

Obsolescence is often associated with hardware which is no longer up to date, but more serious is inadequate software. The maintenance cost of hardware and software tends to decrease initially as the bugs are shaken out, but eventually begins to increase again if an effort is made to keep old hardware and software facilities compatible with new developments.

External developments which require adaptation may be technical or organizational; for instance, changes in communication facilities or a company reorganization can force a database change. While a number of adaptations can be made, especially in systems which are not rigorously bound, eventually a required adaptation will cost more than a reimplementation, which also can take advantage of improved technology.

We can hence speak of a system *life cycle*, beginning from design, development, implemenation, loading with data, operation and enhancement, operation and maintenance only, terminating with a transfer of services to new a new system. Once the expected *life cycle* of a system is determined, other decisions can be made objectively. Investments in system improvements which will not realize adequate benefits over the remaining lifetime can be rejected.

15-3 THE DATABASE ADMINISTRATOR

The final word in this book belongs, quite properly, to the database administrator. The role of a database administrator (DBA) is not yet well defined, and the exact definition varies from one operation to the other. A possible role for the database administrator is the position sketched in Fig. 15-1.

The position of database administrator may be taken by several individuals or may be considered a part-time duty. The choice depends on the number of functions which are assigned to the DBA.

Function of the Database Administrator The DBA acts formally as the control mechanism of a system which manages information loops of the users. The inputs to the control function are the results of the monitoring activity and requests by unsatisfied users. In order to carry out the control function, the DBA can restructure the database or obtain additional facilities for analysis, processing, or communication. The inertia in the various control activities can differ considerably. Additional resources require allocation of investment capital which the DBA has to obtain from management.

The database administrator controls a resource which is important to a variety of users. The users will look toward the DBA whenever they wish to obtain more services from this resource. In order to provide the services, two components of the database operation have to be considered:

1 The database system

2 The database content

Given technical resources and time, it should be possible to satisfy all reasonable technical requests. If the DBA cannot satisfy a user, the user may go to management to request that allocations be increased so that the needs may be satisfied.

If the data contents of the database is inadequate, the DBA has to find the potential source of good data. Source data are often in the domain of other users. The DBA may be able to obtain the data freely or may have to provide a reimbursement for data-collection costs. There will be users who feel that they are already providing more than their share of data. A DBA may have to go to management to present the need if the data is not easily obtained. Management may itself be a major contributor and user of the database system.

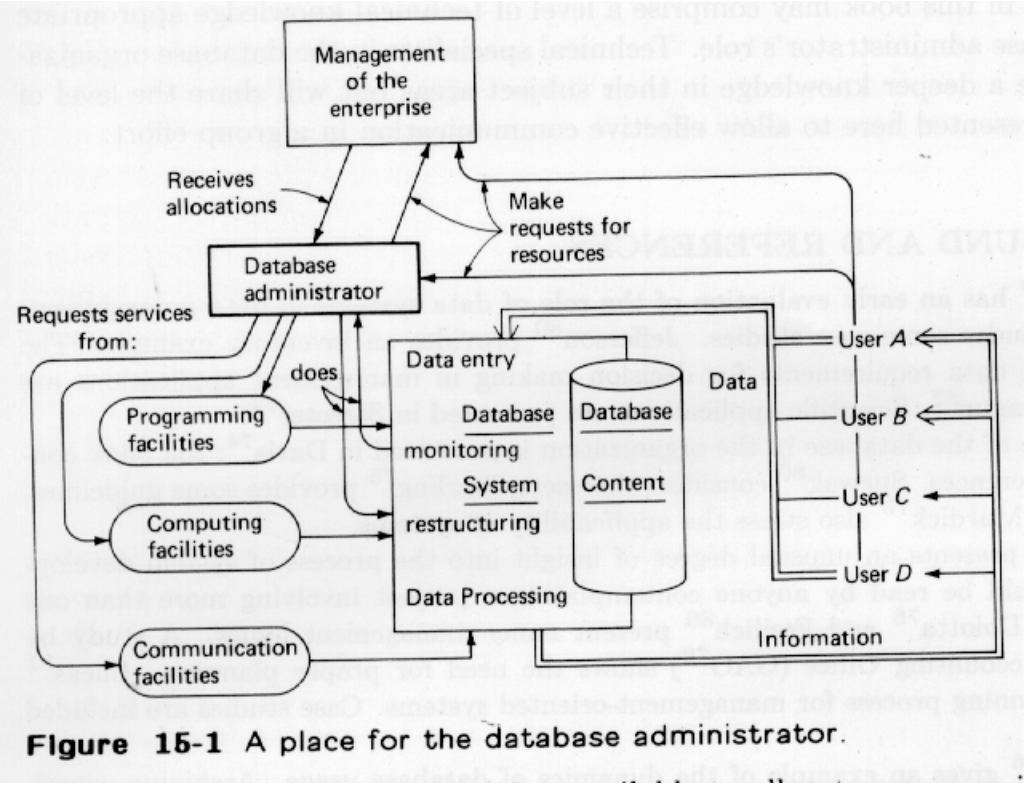

Figure 15-1 A place for the database administrator.

The principal tool that the DBA has available to allocate resources is the schema. The internal schema provides control over the efficiency and responsiveness of processes and the associated protection matrix controls access. The selection of the external schema determines the operational view that a user has of the database model. The conceptual schema, the description of the database model, is constrained by reality and not easily modifiable because of specific needs.

In addition the DBA will have monitoring tools and means to restructure the database. Automatic restructuring is an attractive concept which has up to now been applied only in limited and experimental situations.

Qualifications for a DBA The role of the DBA presented here requires primarily an understanding of the user's needs and a capability to deal with these needs. A DBA who cannot effectively deal with the majority of user requests does not fulfill the responsibility of the position since too much detail will require attention of management. The DBA needs sufficient authority to deal with the majority

of user's needs. This implies a capability to obtain or allocate data collection, programming, computing, and communication resources. This capability implies the presence of two conditions: the command over adequate resources and the competence to manage and balance the use of these resources.

The fact that there are only a small number of true database administrators in industry today may be traced to the fact that people who have an adequate background are not easy to find. The technical competence of people engaged in database administration has to be at a level which enables them to judge the efforts of programmers. They may not be directly managing any programming projects but will depend on programming talent for system maintenance and tool development.

The persons who take on the functions of a DBA have to educate themselves continuously so that they will remain organizationally and technically competent. The material in this book may comprise a level of technical knowledge appropriate to the database administrator's role. Technical specialists in the database organization will have a deeper knowledge in their subject areas but will share the level of knowledge presented here to allow effective communication in a group effort.

BACKGROUND AND REFERENCES

Gruenberger[69] has an early evaluation of the role of data systems in data management; Sanders[74] includes some case studies. Jefferson[80] provides an inventory example. The objectives and data requirements for decision making in management applications are defined by Sprague[82]. Scientific applications are presented in Streeter[74].

The place of the database in the organization is presented in Davis[74]; the book contains many references. Spewak[80] considers the users; Sterling[75] provides some guidelines. The papers in Murdick[75] also stress the applicability of systems.

Brooks[75] presents an unusual degree of insight into the process of system development and should be read by anyone contemplating a project involving more than one programmer. Dolotta[76] and Freilich[80] present many management issues. A study by the General Accounting Office (GAO[79]) shows the need for proper planning. Lucas[81] defines the planning process for management-oriented systems. Case studies are included in Riley[81].

Donovan[76] gives an example of the dynamics of database usage. Archiving experiments are described by Smith[81] and Lawrie[82]. Data are given by Satyanarayanan[81].

Jardine[74] contains a discussion about the role of the database administrator, and an attempt to define the role is made in Steel[75]. Some theoretic approaches (Sundgren[75]) assign a very powerful role to the administrator, included is the design of the database system. Experience with the concept (DeBlasis in Lowenthal[78]) gives cause to consider a more limited function. Formal design techniques and system improvements (as Arditi[78]) will also affect the role of the DBA. Many other articles in the trade literature reflect on the function of the database administrator and the background required to manage database operations. A self-assessment procedure of the ACM in the area was developed by Scheuermann[78] .