Performance Control During System Design

The performance of a system is measured by the time and computer resources required to run it. These may be important for the following reasons:

- Some system functions may have to be completed within a specified time frame;
- The system may compete for computer resources with other systems that have more stringent time constraints.

Performance may not be the most important objective. Trade-offs often need to be made between performance and

- flexibility;
- data independence;
- accessibility of information;
- audit and security considerations;
- currency of information;
- ease of scheduling and impact on concurrent users of the database; or
- disk space.

In some cases, performance may be a constraint to be met rather than an objective to be optimized. If the system meets its time and volume requirements, attention may be turned from performance to other areas.

This chapter covers the following topics:

- Methodology for Performance Control in System Design

Methodology for Performance Control in System Design

The need to achieve satisfactory performance may affect

- the design of the database;
- the options defined when first loading the database;
- the logic of application functions (for example, whether to use direct access or a combination of sequential accesses and sorts); and/or
- operation procedures and scheduling.

Performance requirements must be considered early in the system design process. The following procedure may be used as a basis for controlling performance:

1. Obtain from the users the *time* constraints for each major system function. These requirements are likely to be absolute; that is, the system must meet them.

2. Obtain the computer *resource* constraints from both the users and operations personnel and use the most stringent set.

3. Describe each function in terms of the logical design model specifying the

   - manner in which each record type is processed;
   - access path and the sequence in which records are required;
   - frequency and volume of the run;
   - time available.

4. *Decide* which programs are most "performance critical". The choice may involve volumes, frequency, deadlines, and the effect on the performance or scheduling of other systems. Other programs may also have minimum performance requirements which may constrain the extent to which critical functions can be optimized.

5. *Optimize* the performance of critical functions by shortening their access paths, improving their logic, eliminating database features that increase overhead, and so on. On the first pass, an attempt should be made to optimize performance without sacrificing flexibility, accessibility of information, or other functional requirements of the system.

6. *Estimate* the performance of each critical function. If this does not give a satisfactory result, either a compromise between the time constraints and the functional requirements must be found or a hardware upgrade must be considered.

7. *Estimate* the performance of other system functions. Calculate the total cost and compare the cost and peak period resource requirements with the economic constraints. If the estimates do not meet the constraints, then a solution must be negotiated with the user, operations, or senior management.

8. If possible, *validate* the estimates by loading a test database and measuring the actual performance of various functions. The test database should be similar to the planned one in terms of the number of records contained in each file and the number of values for descriptors. In the test database, the size of each record is relatively unimportant except when testing sequential processing, and then only if records are to be processed in physical sequence.