Traditional File Processing

Excerpts from lecture notes by Greg Hey

TRADITIONAL FILE PROCESSING

The principal feature of file-processing applications is to allow each application/designer to create files that are independent of other file systems within the organization. Each application or project is responsible for designing the contents, capturing the original data, and maintaining the files throughout the application's lifespan.

This leads to non-standardization of file contents, formats, design, and security measures, and makes access (e.g., summary of organization-wide information) very difficult.

Disadvantages of File Processing

When only a few programs used a file, only those programs needed to “know” the layout (position and length of fields within a file). If a change to a file was needed (e.g. an extra field, or some fields grew in size, e.g. date) it was simple for the programmer to change the data, and modify the interpretation of that data in all programs affected.

However, as organizations grew, coordination among programs became complex and led to miscommunication (or no communication) among programmers and, subsequently, errors.

Limited Data Sharing

Since file formats were so prone to change and designers/programmers did not always provide a file design, cross-departmental reporting was problematical.

Limited Reuse of Program Code

Even when the data was semantically the same between two systems (i.e., it meant the same thing), the different formats could present problems. For example, a program that took customer data from within the *invoicing* system and produced mailing labels could not necessarily be used to perform the same function for delivery slips within the *order entry system.*

Development and Maintenance Costs

Whereas the user faces *accessibility* and *sharing* problems, development programmers confront a different but related problem. Because programs depended on their picture of the data formats—which changed between start of project and final test—reopening and updating of previously completed code became necessary.

Departmental Information "Silos"/Protectionism/Ownership

Since files were open to reading and writing (i.e., with no supervisory or governing system as gatekeeper), files could be altered inadvertently, sensitive data could be read by

unauthorized personnel, etc. For this reason and for the perennial political reasons within organizations, departments guarded their data files jealously. ("knowledge is power.")

DATA CONTROL AND AVAILABILITY

Data Duplication Rampant—Due to Undisciplined Development

Data duplication can be a major problem where applications are developed with little coordination or communication. Often this was caused by the following factors:

- incompatible/inappropriate media (some data on tape . . . required on disk)
- freedom of developers to "create" files
- generic "corporate" files being too expensive in media and/or in time to access (i.e., much larger than needed for the task at hand)

Data Integrity

Integrity is similar to *reliability* of data—the accuracy and completeness of its contents. With file processing, none of this was guaranteed.

Limited Security Possible

Since programs could read any file as long as the tape was mounted or the disk inserted, no protective measures were available in this environment.

THE VALUE OF DATA TO AN ORGANIZATION

Data as a Support to Business Operations

Traditionally, data was kept only briefly in an electronic state. Because of the costs associated with magnetic storage, data was quickly converted to other media such as paper or microfilm. Traditionally, companies did not depend on this data to keep their operations running. Early computer applications always had to have a "Plan B"—reverting to the manual way of doing things.

Information Flow as *the* Operation

The traditional perspective changed radically during the 1970s and 1980s with the advent of reliable computer systems (and database technology). New companies and applications arose that depended more on data (and the information represented) than on any physical movement of objects. Examples include insurance and banking applications—and probably the most valuable flow of information with the least amount of object flow—the stock exchanges.