DATABASE SYSTEMS

Lecture 1 Additional Notes

Introduction to Databases

Historical Development of Database Systems

TIMEFRAME	DEVELOPMENT	COMMENTS
1960s (onwards)	File-based systems	Precursor to the database system. Decentralized approach: each department stored and controlled its own data.
Mid-1960s	Hierarchical and network data models	Represents first-generation DBMSs. Main hierarchical system is IMS from IBM and the main network system is IDMS/R from Computer Associates. Lacked data independence and required complex programs to be developed to process the data.
1970	Relational model proposed	Publication of E. F. Codd's seminal paper "A relational model of data for large shared data banks," which addresses the weaknesses of first-generation systems.
1970s	Prototype RDBMSs developed	During this period, two main prototypes emerged: the Ingres project at the University of California at Berkeley (started in 1970) and the System R project at IBM's San José Research Laboratory in California (started in 1974), which led to the development of SQL.
1976	ER model proposed	Publication of Chen's paper "The Entity-Relationship model—Toward a unified view of data." ER modeling becomes a significant component in methodologies for database design.
1979	Commercial RDBMSs appear	Commercial RDBMSs like Oracle, Ingres, and DB2 appear. These represent the second generation of DBMSs.
1987	ISO SQL standard	SQL is standardized by the ISO (International Standards Organization). There are subsequent releases of the standard in 1989, 1992 (SQL2), 1999 (SQL:1999), 2003 (SQL:2003), 2008 (SQL:2008), and 2011 (SQL:2011).
1990s	OODBMS and ORDBMSs appear	This period initially sees the emergence of OODBMSs and later ORDBMSs (Oracle 8, with object features released in 1997).
1990s	Data warehousing systems appear	This period also see releases from the major DBMS vendors of data warehousing systems and thereafter data mining products.
Mid-1990s	Web–database integration	The first Internet database applications appear. DBMS vendors and third-party vendors recognize the significance of the Internet and support web–database integration.
1998	XML	XML 1.0 ratified by the W3C. XML becomes integrated with DBMS products and native XML databases are developed.

Figure 1.10 Historical development of database systems.

[© Pearson - Database Systems: A Practical Approach to Design, Implementation, and Management - http://app.kortext.com/read/80118/?page=74]

Limitations of the File-Based Approach

1.2.2 Limitations of the File-Based Approach

This brief description of traditional file-based systems should be sufficient to discuss the limitations of this approach. We list five problems in Table 1.1.

Separation and isolation of data

When data is isolated in separate files, it is more difficult to access data that should be available. For example, if we want to produce a list of all houses that match the requirements of clients, we first need to create a temporary file of those clients who have "house" as the preferred type. We then search the PropertyForRent file for those properties where the property type is "house" and the rent is less than the client's maximum rent. With file systems, such processing is difficult. The application developer must synchronize the processing of two files to ensure that the correct data is extracted. This difficulty is compounded if we require data from more than two files.

10/10/2016 9:55:22 PM - 65371V - 86.184.39.189

Duplication of data

Owing to the decentralized approach taken by each department, the file-based approach encouraged, if not necessitated, the uncontrolled duplication of data. For example, in Figure 1.5 we can clearly see that there is duplication of both property and client details in the Sales and Contracts Departments. Uncontrolled duplication of data is undesirable for several reasons, including:

- Duplication is wasteful. It costs time and money to enter the data more than once.
- It takes up additional storage space, again with associated costs. Often, the duplication of data can be avoided by sharing data files.

TABLE 1.1 Limitations of file-based systems.

Separation and isolation of data
Duplication of data
Data dependence
Incompatible file formats
Fixed queries/proliferation of application programs

[© Pearson - Database Systems: A Practical Approach to Design, Implementation, and Management - http://app.kortext.com/read/80118/?page=60-62]

- Perhaps more importantly, duplication can lead to loss of data integrity; in other words, the data is no longer consistent. For example, consider the duplication of data between the Payroll and HR Departments described previously. If a member of staff moves and the change of address is communicated only to HR and not to Payroll, the person's paycheck will be sent to the wrong address. A more serious problem occurs if an employee is promoted with an associated increase in salary. Again, let's assume that the change is announced to HR, but the change does not filter through to Payroll. Now, the employee is receiving the wrong salary. When this error is detected, it will take time and effort to resolve. Both these examples illustrate inconsistencies that may result from the duplication of data. As there is no automatic way for HR to update the data in the Payroll files, it is not difficult to foresee such inconsistencies arising. Even if Payroll is notified of the changes, it is possible that the data will be entered incorrectly.

Data dependence

As we have already mentioned, the physical structure and storage of the data files and records are defined in the application code. This means that changes to an existing structure are difficult to make. For example, increasing the size of the PropertyForRent address field from 40 to 41 characters sounds like a simple change, but it requires the creation of a one-off program (that is, a program that is run only once and can then be discarded) that converts the PropertyForRent file to the new format. This program has to:

10/10/2016 9:59:09 PM - 65371V - 86.184.39.189

- Open the original PropertyForRent file for reading
- Open a temporary file with the new structure
- Read a record from the original file, convert the data to conform to the new structure, and write it to the temporary file, then repeat this step for all records in the original file
- Delete the original PropertyForRent file
- Rename the temporary file as PropertyForRent

In addition, all programs that access the PropertyForRent file must be modified to conform to the new file structure. There might be many such programs that access the PropertyForRent file. Thus, the programmer needs to identify all the affected programs, modify them, and then retest them. Note that a program does not even have to use the address field to be affected: it only has to use the PropertyForRent file. Clearly, this process could be very time-consuming and subject to error. This characteristic of file-based systems is known as **program–data dependence**.

10/10/2016 9:59:09 PM - 65371V - 86.184.39.189

Incompatible file formats

Because the structure of files is embedded in the application programs, the structures are dependent on the application programming language. For example, the structure of a file generated by a COBOL program may be different from the structure of a file generated by a C program. The direct incompatibility of such files makes them difficult to process jointly.

For example, suppose that the Contracts Department wants to find the names and addresses of all owners whose property is currently rented out. Unfortunately,

10/10/2016 9:59:46 PM - 65371V - 86.184.39.189

Contracts does not hold the details of property owners; only the Sales Department holds these. However, Contracts has the property number (propertyNo), which can be used to find the corresponding property number in the Sales Department's PropertyForRent file. This file holds the owner number (ownerNo), which can be used to find the owner details in the PrivateOwner file. The Contracts Department programs in COBOL and the Sales Department programs in C. Therefore, matching propertyNo fields in the two PropertyForRent files requires that an application developer write software to convert the files to some common format to facilitate processing. Again, this process can be time-consuming and expensive.

Fixed queries/proliferation of application programs

From the end-user's point of view, file-based systems were a great improvement over manual systems. Consequently, the requirement for new or modified queries grew. However, file-based systems are very dependent upon the application developer, who has to write any queries or reports that are required. As a result, two things happened. In some organizations, the type of query or report that could be produced was fixed. There was no facility for asking unplanned (that is, spur-of-the-moment or *ad hoc*) queries either about the data itself or about which types of data were available.

In other organizations, there was a proliferation of files and application programs. Eventually, this reached a point where the DP Department, with its existing resources, could not handle all the work. This put tremendous pressure on the DP staff, resulting in programs that were inadequate or inefficient in meeting the demands of the users, limited documentation, and difficult maintenance. Often, certain types of functionality were omitted:

- There was no provision for security or integrity.
- Recovery, in the event of a hardware or software failure, was limited or nonexistent.
- Access to the files was restricted to one user at a time—there was no provision for shared access by staff in the same department.

In either case, the outcome was unacceptable. Another solution was required.

1.3 Database Approach

All of the previously mentioned limitations of the file-based approach can be attributed to two factors:

(1) The definition of the data is embedded in the application programs, rather than being stored separately and independently.

(2) There is no control over the access and manipulation of data beyond that imposed by the application programs.

To become more effective, a new approach was required. What emerged were the database and the Database Management System (DBMS). In this section, we provide

Advantages and Disadvantages of DBMSs[1]

Advantages

Control of data redundancy

Traditional file-based systems waste space by storing the same information in more than one file. In contrast, the database approach attempts to eliminate redundancy by integrating the files so that multiple copies of the same data are not stored. The database approach, however, does not eliminate redundancy entirely, but controls the amount of redundancy inherent in the database.

Data consistency

By eliminating or controlling redundancy, the risk of inconsistencies occurring is reduced. If a data item is stored only once in the database, any update to its value need only be performed once and the new value is immediately available to all users.

More information from the same amount of data

With the integration of operational data, it is possible for the organisation to derive additional information from the same data.

Sharing of data

Typically, files are owned by the people or departments that use them. On the other hand, the database belongs to the entire organisation and can be shared by all authorised users.

Improved data integrity

Database integrity refers to the validity and consistency of stored data. Integrity is usually expressed in form of constraints, which are consistency rules that the database is not permitted to violate. Integration allows the DBA to define integrity constraints, and the DBMS to enforce them.

Enforcement of standards

Integration allows the DBA to define, and the DBMS to enforce the necessary standards. These may include departmental, organisational, national or international standards. For example, data formats to facilitate the exchange of data between systems, naming conventions, documentation standards, update procedures and access rules.

Economy of scale

Combining all the organisation's operational data into one database and creating a set of applications that work on this one source of data can result in cost savings. The budget that would normally be allocated to each department for the development and maintenance of its file-based system can be combined, possibly resulting in a lower total cost, leading to an economy of scale.

Balance of conflicting requirements

Each user or department has needs that may be in conflict with the needs of other users. Because the database is under the control of the DBA, the DBA can make decisions about the design and

[1] Source: Database Systems: A Practical Approach to Design, Implementation and Management, 6th Edition, Addison Wesley (2014), By T. Connolly, C. Begg, pp. 75-79

operational use of the database that provide the best use of resources for the organisation as a whole.

Improved data accessibility and responsiveness

As a result of integration, data that crosses departmental boundaries is directly accessible to end-users. This provides a system with potentially much more functionality that can, for example, be used to provide better services to the end-user or the organisation's clients. Many DBMSs also provide query languages and interfaces that allow users to construct ad hoc queries on the data in the database and obtain the information immediately at their terminals, without requiring a programmer to write some specialised software application to extract the information.

Increased productivity

The DBMS provides many of the standard functions that the programmer would normally have to write in a file-based application. The provision of these functions allows the programmer to concentrate on the specific functionality required by the users without having to worry about low-level implementation details.

Improved maintenance through data independence

In file-based systems, the descriptions of the data and the logic for accessing the data are built into each application program, making the programs dependent on the data. Any changes, for example increasing the number of characters for a data item, can require substantial changes to the programs affected by the change. In contrast, a DBMS separates the data descriptions from the applications, thereby making applications immune to changes in the data descriptions. This is known as **data independence**.

Increased concurrency

In some file-based systems, if two or more users are allowed to simultaneously access the same file, it is possible that the accesses will interfere with each other, resulting in loss of information or even loss of integrity. Many DBMSs manage concurrent database access and ensure that such problems cannot occur.

Improved backup and recovery services

Many file-based systems place the responsibility on the user to provide measures to protect the data from failures to the computer system or application program. This may typically involve nightly backups. However, if there is a failure the next day, the work that has taken since the last backup is lost and has to be re-entered. In contrast, modern DBMSs provide facilities to minimise the amount of processing that is lost following a failure.

Disadvantages

Complexity

The provision of the functionality that we expect of a good DBMS makes the DBMS an extremely complex piece of software. Database designers and developers and data and database administrators must understand this functionality to take full advantage of it. Failure to understand the system can result in bad design decisions, which can have serious consequences for an organisation.

Size

The complexity and breadth of the functionality makes the DBMS an extremely large piece of software, occupying many megabytes of disk space and requiring substantial amounts of memory to run efficiently.

Cost of DBMSs

The cost of DBMSs varies significantly, depending on the environment and functionality provided. For example, a single-user DBMS for a personal computer may only cost £100 or you may even get for free with open source software. However, a large multi-user DBMS environment servicing hundreds of users can be very expensive. There is also the recurrent annual maintenance cost.

Additional hardware costs

The disk storage requirements for the DBMS and the database may necessitate the purchase of additional storage space. Furthermore, to achieve the required performance, it may be necessary to have a machine(s) dedicated to running the DBMS. The procurement of additional hardware results in further expenditure.

Cost of conversion

In some situations, the cost of the DBMS and extra hardware may be relatively small compared with the cost of converting existing applications to run on the new DBMS and hardware. This cost may also include the cost of training staff to use these new systems. This cost is one of the main reasons why some organisations feel tied to their current systems and cannot switch to modern database technology.

Performance

Typically, a file-based system is written for a specific application. As a result, performance is generally very good. The DBMS, however, is written to be more general to cater for many applications rather than just one. The result is that some applications may not run as fast as they used to.

Greater impact of a failure

The centralisation of resources increases the vulnerability of the system. As all users and applications rely on the availability of the DBMS, the failure of certain components can bring operations to a halt.

Review Question

Compare and contrast the file-based system to a database management system.