Topic 1.3: Motivations for Using Databases

The development of a database application can be a time-consuming, costly, and disruptive process, but many organizations place their applications in databases. Why do they bother?

Benefits of Using Databases

The development of a database application can be a time-consuming, costly, and disruptive process, but many organizations place their applications in databases. Why do they bother?

There can be many advantages to using database technology, but the ultimate reason is that the benefits are expected to outweigh the costs.

All computer projects should begin with a cost-benefit analysis. It is wise to review the cost-benefit analysis from time to time during the development process in order to keep the project on track, particularly if requirements change.

It is not possible to cover all benefits that will be important to *every* organization. However, this topic includes many of the benefits important to most organizations.

Data Sharing

When applications are developed using flat files, each application's files are designed for that specific application. Files cannot be easily shared across applications not only because the file technology did not support sharing the files, but also because the files were not designed to include the data requirements of the other applications. When applications read flat files, the data can only be presented in the exact way the records are stored. Consider the following Order File design:

Order File

Order ID	Customer ID	Employee ID	Order Date	Shipped Date	Shipping Fee
51	26	9	4/05/2006	4/05/2006	$60.00
56	6	2	4/03/2006	4/03/2006	$0.00
79	6	2	6/23/2006	6/23/2006	$0.00

If another application needs access to the total shipping fees collected by state, it cannot use this order file because it does not include the state for each order. While adding the state code to the order file would seem a simple solution, doing so creates two new issues. First, because flat file processing requires each application program to read entire records, every single application that reads the file would have to be changed -- otherwise it would fail the next time it tried to read the order file because the records would be longer than expected. Second, the postal code is really a property of the customer who placed the order, and if a customer changed its location, the order file would be inconsistent with customer files that might appear in another application. Another alternative would be to create a file containing only the required information, but that file would have to updated or recreated as updates are applied to the order and customer files.

Here is a partial list of the Customer File.

Customer File

Customer ID	Company Name	...	State
6	Company F	...	WI
26	Company Z	...	FL

Database systems have the unique property of being able to present multiple views of the data to different users. (Users in this context can be individuals or application programs.) This ability is enabled by the layers of abstraction provided by the DBMS, specifically the logical and external layers presented in Topic 1.1. With database technology, it would be relatively easy to create a *view* of the data tailored to the needs of the application that required shipping fees by state (using the Customer ID in each order to match is with its corresponding customer in order to obtain the State).

State Shipping View

State	Shipping Fee
FL	$60.00
WI	$0.00

Controlled Redundancy

Another important benefit of using database technology is the ability to control redundancy. This is accomplished in two ways.

First, the ability to produce views of the data tailored to individual applications (as discussed above) eliminates the need for each application system to create master files tailored to its individual data needs. When I worked for a large manufacturing company, there were 17 different master files that held product data for various applications. It often took 7 or more calendar days for a simple update such as the addition of a new product to find its way to all 17 master files, so as you can imaging, the files were never in a consistent state. Using database technology, a small application team created a master Product table using a relational DBMS and within a few months, all the applications were using one consistent set of master data for products. This was possible with minimal disruption to all the existing applications because a database query (view) could be run for each application that would present the data exactly the way the application expected it. Needless to say, this was a big win, especially for one of the first database systems this organization had ever deployed.

Second, through the design process known as normalization, DBMS objects (tables in relational technology) can be designed in such a way that they hold data about a single business entity. (Normalization is presented in Module 3 of this course.) If, for example, we were to store all the information regarding a customer with each order placed by the customer, we end up storing the same customer information over and over again. Then when it comes time to update any of the customer's information, we must update many rows of data. The moment you store the same information in multiple places, you also introduce the potential for the data to become inconsistent over time.

Many people believe that one of the goals of database technology, particular relational database technology, is to eliminate redundancy. This is untrue. Recall from Topic 1.1 that relational databases use foreign keys to relate tables. For example, the Customer ID, which is the primary key of the Customer table, is stored in a column in the Order table (as a foreign key) so that we can associate each order with the customer who placed it. The Customer ID is therefore stored redundantly in the Order table. However, this is controlled redundancy. In fact, foreign keys should be the *only* redundant columns in a well-designed relational database.

Data Consistency

By controlling redundancy, we obtain data consistency. If a data item is stored only once, the any update of its data value is immediately available to all users. Note, however, in some circumstances, databases are redundantly stored for recovery purposes. However, this does not pose a consistency issues for two reasons. First, database backups taken for the purposes of fast recovery from failures should not be available to database users, so even if they are inconsistent for a short period of time, the database users will never see the inconsistent data. Second, using database replication technology, the backup copy can be automatically updated very shortly after updates take place. This is often known as a "warm backup" or "near real-time database replication".

Data Integrity Enforced

Recall that data integrity is the insurance of accurate data in the database. Unlike flat file technology, databases permit the definition of data integrity constraints that are automatically enforced by the DBMS. With file systems, it is up to the applications to enforce data integrity, and while application developers will tell you how reliable the data integrity they write into the application programs is, there are two reliability problems at play. First, application programs can contain errors that allow incorrect data to sneak through. Ask any business intelligence professional to list the top few issues they encounter on a daily basis and data quality is bound to be one of them. Second, it is possible to bypass the application programs when loading data into a database or flat file, and doing so also bypasses the integrity controls that the application developers so carefully placed in those application programs. For example, during acquisitions and mergers, the data from one organization must be loaded into the surviving organization's systems. Generally this is done by writing conversion programs, which often do not contain the exact same integrity controls that were in the original application programs that maintained the data.

While all database systems permit the inclusion of data integrity constrains in one form or another, relational databases systems provide the most comprehensive capability. These integrity constraints are included in Topic 1.2.

Data Security and Privacy

Data security is the protection of the database from unauthorized users. Modern DBMS products have a built-in security subsystem that allows administrators to control access to the data contained in the database, giving each database user exactly the data required for its work, nothing more, and nothing less. The very fact that databases consolidate so much data in one location makes security controls essential because different users require different levels of access to the consolidated database. Database security is presented in detail in Topic 6.4.

Along with security, the security subsystem assists with ***data privacy***, which is the protection of sensitive data, such as healthcare and financial data, from unauthorized disclosure. While the DBMS security subsystem provides controls to prevent unauthorized access, this is usually not enough to meet various regulatory requirements, and thus, DBMS security subsystems usually support data encryption, which scrambles the data into a for that makes it useless to individuals unless they have the key required to decrypt (unscramble) the data.

Other Benefit

Other benefits from using database systems include the following:

- **Intellectual investment:** Existing programs and logical data structures representing many work-years of development time will not have to be redone when changes are made to the database.
- **Ease of use:** Users can gain access to data in a simple fashion since complexity is hidden by the DBMS. Also, the storage of metadata (data that describes the data) within the DBMS makes the databases self-documenting.
- **Unanticipated requests: S**pontaneous requests for data can be handled by high-level query or report-generation languages, minimizing the need to write application programs to obtain answers. Also, new application needs can be met with existing data rather than creating new data files for each new application.
- **Change: T**he database can grow and change without interfering with established ways of using the data. This is possible because of the physical and logical data independence that databases offer when they provide multiple views of the stored data. When new data items are added to

support a new application, the changes to the database objects (tables in relational databases) have no adverse effect on existing processes and queries because the view they are using does not need to be changed to include the new data.

Cost of Using Databases

Now let's look at the costs of using databases that must be weighed against the benefits during the cost-benefit analysis of the database project.

It is not possible to cover a set of costs that will be important to *every* organization. However, this discussion includes the costs important to most organizations.

Complexity

Creating integrated database designs that can serve the needs of an entire organization while allowing for growth and change that does not disrupt existing uses of the database is a complex undertaking. At the very least, the design phase of new database projects take longer, which adds to the project cost.

The additional complexity of consolidated database systems requires that the designers, developers, and users of these databases be trained in the structure of the database and the capabilities of the DBMS. Training costs include not only what must be paid to vendors and instructors, but also the cost of taking people away from their normal duties while the training takes place.

Software Costs

The cost of DBMS software varies significantly depending on the required features, and maintenance costs. Even so-called freeware DBMS products are not completely free if you read the fine print regarding license restrictions for commercial use and pay-as-you-go technical support plans.

The DBMS is likely not the only software you will need. Tools for data modeling and design, interactive database queries (SQL clients), report-writing, and the like will likely also be required.

Hardware Costs

On the surface, it would seem that the the storage hardware costs for a DBMS implementation would be the same as a flat file system because, after all, the same data is being stored, In fact, you might be led to believe that the storage costs would be less because the DBMS would not be storing all the redundant data that would be contained in the flat file system. However, databases usually require higher speed hardware than their flat file system counterparts in order to process queries efficiently. Furthermore, databases generally require redundant hardware to mitigate the risk of a centralized storage system -- the last thing you need is one hard drive failure preventing your entire organization from accessing critical operational information.

DBMS software can require substantial amounts of computer memory and higher speed processors and networks than flat file systems. However, these costs are at least somewhat offset by the consolidation of many application-specific data file servers into a centralized database server.

Conclusion

These costs can be quite significant, and the database developer should be aware of their potential for causing difficulty.

Database projects rarely fail due to strictly technical problems. Most project failures are due to some combination of lack of planning, lack of risk management, organizational strife, departmental politics, and bad management.