(Database) Application Programs

Application program: A computer program that interacts with the database by issuing an appropriate request (typically an SQL statement) to the DBMS.
Users interact with the database through a number of **application programs** that are used to create and maintain the database and to generate information. These programs can be conventional batch applications or, more typically nowadays, they will be online applications. The application programs may be written in some programming language or in some higher-level fourth-generation language.

Views

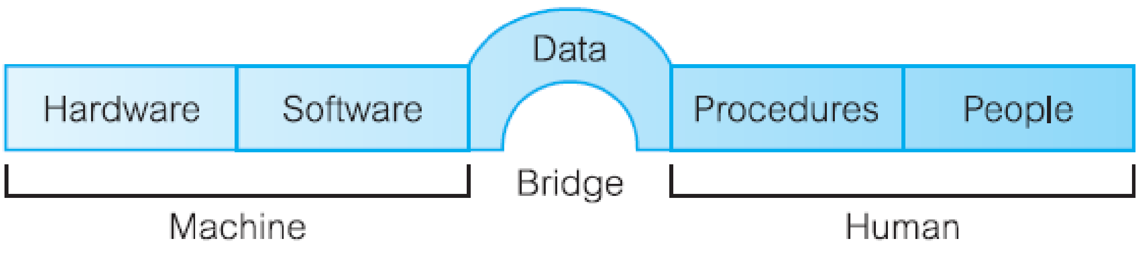
With this functionality, the DBMS is an extremely powerful and useful tool. However, as the end-users are not too interested in how complex or easy a task is for the system, it could be argued that the DBMS has made things more complex because they now see more data than they actually need or want.

In recognition of this problem, a DBMS provides another facility known as a **view mechanism**, which allows each user to have his or her own view of the database (a **view** is in essence some subset of the database). For example, we could set up a view that allows the Contracts Department to see only the data that they want to see for rental properties.
As well as reducing complexity by letting users see the data in the way they want to see it, views have several other benefits:

- *Views provide a level of security*. Views can be set up to exclude data that some users should not see. For example, we could create a view that allows a branch manager and the Payroll Department to see all staff data, including salary details, and we could create a second view that other staff would use that excludes salary details.
- *Views provide a mechanism to customize the appearance of the database*. For example, the Contracts Department may wish to call the monthly rent field (rent) by the more obvious name, Monthly Rent.
- *A view can present a consistent, unchanging picture of the structure of the database*, even if the underlying database is changed (for example, fields added or removed, relationships changed, files split, restructured, or renamed). If fields are added or removed from a file, and these fields are not required by the view, the view is not affected by this change. Thus, a view helps provide the program–data independence we mentioned earlier.

Components of the DBMS Environment

We can identify five major components in the DBMS environment: hardware, software, data, procedures, and people, as illustrated below



Hardware

The DBMS and the applications require hardware to run. The hardware can range from a single personal computer, to a single mainframe, to a network of computers. The particular hardware depends on the organization's requirements and the DBMS used. Some DBMSs run only on particular hardware or operating systems, while others run on a wide variety of hardware and operating systems. A DBMS requires a minimum amount of main memory and disk space to run, but this minimum configuration may not necessarily give acceptable performance.

Software
The software component comprises the DBMS software itself and the application programs, together with the operating system, including network software if the DBMS is being used over a network. Typically, application programs are written in a third-generation programming language (3GL), such as 'C', C□□, Java, Visual Basic, COBOL, Fortran, Ada, or Pascal, or using a fourth-generation language (4GL), such as SQL, embedded in a third-generation language.

Data
Perhaps the most important component of the DBMS environment, certainly from the end-users' point of view, is the data. From the figure, we observe that the data acts as a bridge between the machine components and the human components. The database contains both the operational data and the metadata, the 'data about data'. The structure of the database is called the **schema**.

Procedures
Procedures refer to the instructions and rules that govern the design and use of the database. The users of the system and the staff that manage the database require documented procedures on how to use or run the system. These may consist of instructions on how to:

- log on to the DBMS;
- use a particular DBMS facility or application program;
- start and stop the DBMS;
- make backup copies of the database;
- handle hardware or software failures. This may include procedures on how to identify the failed component, how to fix the failed component (for example, telephone the appropriate hardware engineer) and, following the repair of the fault, how to recover the database;
- change the structure of a table, reorganize the database across multiple disks, improve performance, or archive data to secondary storage.

People

- The final component is the people involved with the system. We discuss this component

Database Design: The Paradigm Shift

The structure of the database is determined during **database design**. However, carrying out database design can be extremely complex. To produce a system that will satisfy the organization's information needs requires a different approach from that of file-based systems, where the work was driven by the application needs of individual departments. For the database approach to succeed, the organization now has to think of the data first and the application second. This change in approach is sometimes referred to as a *paradigm shift*. For the system to be acceptable to the end-users, the database design activity is crucial. A poorly designed database will generate errors that may lead to bad decisions being made, which may have serious repercussions for the organization. On the other hand, a well-designed database produces a system that provides the correct information for the decision-making process to succeed in an efficient way.

Roles in the Database Environment

In this section, we examine what we listed in the previous section as the fifth component of the DBMS environment: the **people**. We can identify four distinct types of people that participate in the DBMS environment: data and database administrators, database designers, application developers, and the end-users.

Data and Database Administrators

The database and the DBMS are corporate resources that must be managed like any other resource. Data and database administration are the roles generally associated with the management and control of a DBMS and its data. The **Data Administrator** (DA) is responsible for the management of the data resource including database planning, development and maintenance of standards, policies and procedures, and conceptual/logical database design. The DA consults with and advises senior managers, ensuring that the direction of database development will ultimately support corporate objectives. The **Database Administrator** (DBA) is responsible for the physical realization of the database, including physical database design and implementation, security and integrity control, maintenance of the operational system, and ensuring satisfactory performance of the applications for users. The role of the DBA is more technically oriented than the role of the DA, requiring detailed knowledge of the target DBMS and the system environment. In some organizations there is no distinction between these two roles; in others, the importance of the corporate resources is reflected in the allocation of teams of staff dedicated to each of these roles.

Database Designers

In large database design projects, we can distinguish between two types of designer: logical database designers and physical database designers. The **logical database designer** is concerned with identifying the data (that is, the entities and attributes), the relationships between the data, and the constraints on the data that is to be stored in the database. The logical database designer must have a thorough and complete understanding of the organization’s data and any constraints on this data (the constraints are sometimes called **business rules**). These constraints describe the main characteristics of the data as viewed by the organization.

Application Developers

Once the database has been implemented, the application programs that provide the required functionality for the end-users must be implemented. This is the responsibility of the **application developers**. Typically, the application developers work from a specification produced by systems analysts. Each program contains statements that request the DBMS to perform some operation on the database. This includes retrieving data, inserting, updating, and deleting data.

End-Users

The end-users are the ‘clients’ for the database, which has been designed and implemented, and is being maintained to serve their information needs. End-users can be classified according to the way they use the system:

- **Naïve users** are typically unaware of the DBMS. They access the database through specially written application programs that attempt to make the operations as simple as possible. They invoke database operations by entering simple commands or choosing options from a menu. This means that they do not need to know anything about the database or the DBMS.

- **Sophisticated users**. At the other end of the spectrum, the sophisticated end-user is familiar with the structure of the database and the facilities offered by the DBMS. Sophisticated end-users may use a high-level query language such as SQL to perform the required operations. Some sophisticated end-users may even write application programs for their own use.

Advantages and Disadvantages of DBMSs
The database management system has promising potential advantages. Unfortunately, there are also disadvantages.

Advantages

1. **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 the redundancy by integrating the files so that multiple copies of the same data are not stored. However, the database approach does not eliminate redundancy entirely, but controls the amount of redundancy inherent in the database. Sometimes, it is necessary to duplicate key data items to model relationships. At other times, it is desirable to duplicate some data items to improve performance.
2. **Data consistency**
   By eliminating or controlling redundancy, we reduce the risk of inconsistencies occurring. If a data item is stored only once in the database, any update to its value has to be performed only once and the new value is available immediately to all users. If a data item is stored more than once and the system is aware of this, the system can ensure that all copies of the item are kept consistent.
3. **More information from the same amount of data**
   With the integration of the operational data, it may be possible for the organization to derive additional information from the same data.
4. **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 organization and can be shared by all authorized users. In this way, more users share more of the data. Furthermore, new applications can build on the existing data in the database and add only data that is not currently stored, rather than having to define all data requirements again. The new applications can also rely on the functions provided by the DBMS, such as data definition and manipulation, and concurrency and recovery control, rather than having to provide these functions themselves.
5. **Improved data integrity**
   Database integrity refers to the validity and consistency of stored data. Integrity is usually expressed in terms of **constraints**, which are consistency rules that the database is not permitted to violate. Constraints may apply to data items within a single record or they may apply to relationships between records.
6. **Improved security**
   Database security is the protection of the database from unauthorized users. Without suitable security measures, integration makes the data more vulnerable than file-based

systems. However, integration allows the DBA to define, and the DBMS to enforce, database security. This may take the form of user names and passwords to identify people authorized to use the database. The access that an authorized user is allowed on the data may be restricted by the operation type (retrieval, insert, update, delete). For example, the DBA has access to all the data in the database; a branch manager may have access to all data that relates to his or her branch office; and a sales assistant may have access to all data relating to properties but no access to sensitive data such as staff salary details.

7. **Enforcement of standards**
   Again, integration allows the DBA to define and enforce the necessary standards. These may include departmental, organizational, national, or international standards for such things as data formats to facilitate exchange of data between systems, naming conventions, documentation standards, update procedures, and access rules.
8. **Economy of scale**
   Combining all the organization'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. In this case, 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. The combined budget can be used to buy a system configuration that is more suited to the organization's needs. This may consist of one large, powerful computer or a network of smaller computers.
9. **Balance of conflicting requirements**
   Each user or department has needs that may be in conflict with the needs of other users. Since the database is under the control of the DBA, the DBA can make decisions about the design and operational use of the database that provide the best use of resources for the organization as a whole. These decisions will provide optimal performance for important applications, possibly at the expense of less critical ones.
10. **Improved data accessibility and responsiveness**
    Again, as a result of integration, data that crosses departmental boundaries is directly accessible to the 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 organization's clients. Many DBMSs provide query languages or report writers that allow users to ask *ad hoc* questions and to obtain the required information almost immediately at their terminal, without requiring a programmer to write some software to extract this information from the database.
11. **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. A change to the structure of the data, for example making an address 41 characters instead of 40 characters, or a change to the way the data is stored on disk, can require substantial alterations to the programs that are 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** and is discussed further in Section 2.1.5. The provision of data independence simplifies database application maintenance.
12. **Increased concurrency**
    In some file-based systems, if two or more users are allowed to access the same file simultaneously, 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 such problems cannot occur.
13. **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 involve taking a nightly backup of the data. In the event of a failure during the next day, the backup is restored and the work that has taken place since this backup is lost and has to be re-entered. In contrast, modern DBMSs provide facilities to minimize the amount of processing that is lost following a failure.

Disadvantages

The disadvantages of the database approach are.

1. **Complexity**
   The provision of the functionality we expect of a good DBMS makes the DBMS an extremely complex piece of software. Database designers and developers, the data and database administrators, and end-users must understand this functionality to take full advantage of it. Failure to understand the system can lead to bad design decisions, which can have serious consequences for an organization.
2. **Size**
   The complexity and breadth of 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.
3. **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 Ksh100,000. However, a large mainframe multi-user DBMS servicing hundreds of users can be extremely expensive, perhaps Ksh 10,000,000 or even Ksh100,000,000. There is also the recurrent annual maintenance cost, which is typically a percentage of the list price.
4. **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 purchase a larger machine, perhaps even a machine dedicated to running the DBMS. The procurement of additional hardware results in further expenditure.
5. **Cost of conversion**
   In some situations, the cost of the DBMS and extra hardware may be insignificant compared with the cost of converting existing applications to run on the new DBMS and hardware. This cost also includes the cost of training staff to use these new systems, and possibly the employment of specialist staff to help with the conversion and running of the system. This cost is one of the main reasons why some organizations feel tied to their current systems and cannot switch to more modern database technology. The term **legacy system** is sometimes used to refer to an older, and usually inferior, system.
6. **Performance**
   Typically, a file-based system is written for a specific application, such as invoicing. As a result, performance is generally very good. However, the DBMS is written to be more general, to cater for many applications rather than just one. The effect is that some applications may not run as fast as they used to.
7. **Higher impact of a failure**
   The centralization of resources increases the vulnerability of the system. Since all users and applications rely on the availability of the DBMS, the failure of certain components can bring operations to a halt.

<u>Continuation of lecture notes on 28 march 2017</u>

<u>Function of a DBMS</u>

In this section we look at the types of function and service we would expect a DBMS to provide.

(1) Data storage, retrieval, and update

A DBMS must furnish users with the ability to store, retrieve, and update data in the database. This is the fundamental function of a DBMS. From earlier discussions, clearly in providing this functionality the DBMS should hide the internal physical implementation details (such as file organization and storage structures) from the user.

(2) A user-accessible catalog

A DBMS must furnish a catalog in which descriptions of data items are stored and which is accessible to users.

A key feature of the ANSI-SPARC architecture is the recognition of an integrated **system catalog** to hold data about the schemas, users, applications, and so on. The catalog is expected to be accessible to users as well as to the DBMS. A system catalog, or data dictionary, is a repository of information describing the data in the database: it is, the 'data about the data' or **metadata**. The amount of information and the way the information is used vary with the DBMS. Typically, the system catalog stores:

- names, types, and sizes of data items;
- names of relationships;
- integrity constraints on the data;
- names of authorized users who have access to the data;
- the data items that each user can access and the types of access allowed; for example, insert, update, delete, or read access;
- external, conceptual, and internal schemas and the mappings between the schemas;
- usage statistics, such as the frequencies of transactions and counts on the number of accesses made to objects in the database.

The DBMS system catalog is one of the fundamental components of the system. Many of the software components that we describe in the next section rely on the system catalog for information. Some benefits of a system catalog are:

- Information about data can be collected and stored centrally. This helps to maintain control over the data as a resource.
- The meaning of data can be defined, which will help other users understand the purpose of the data.
- Communication is simplified, since exact meanings are stored. The system catalog may also identify the user or users who own or access the data.
- Redundancy and inconsistencies can be identified more easily since the data is centralized.
- Changes to the database can be recorded.
- The impact of a change can be determined before it is implemented, since the system catalog records each data item, all its relationships, and all its users.
- Security can be enforced.
- Integrity can be ensured.
- Audit information can be provided.

(3) Transaction support

A DBMS must furnish a mechanism which will ensure either that all the updates corresponding to a given transaction are made or that none of them is made. A transaction is a series of actions, carried out by a single user or application program, which accesses or changes the contents of the database. For example if you want to delete a member of staff from the database *and* to reassign the properties that he or she managed to another member of staff. In this case, there is more than one change to be made to the database. If the transaction fails during execution, perhaps because of a computer crash, the database will be in an **inconsistent** state: some changes will have been made and others not. Consequently, the changes that have been made will have to be undone to return the database to a consistent state again.

(4) Concurrency control services

A DBMS must furnish a mechanism to ensure that the database is updated correctly when multiple users are updating the database concurrently.

One major objective in using a DBMS is to enable many users to access shared data concurrently.

Concurrent access is relatively easy if all users are only reading data, as there is no way that they can interfere with one another. However, when two or more users are accessing the database simultaneously and at least one of them is updating data, there may be interference that can result in inconsistencies.

The DBMS must ensure that, when multiple users are accessing the database, interference cannot occur.

(5) Recovery services

A DBMS must furnish a mechanism for recovering the database in the event that the database is damaged in any way.

When discussing transaction support, we mentioned that if the transaction fails then the database has to be returned to a consistent state. This may be a result of a system crash, media failure, a hardware or software error causing the DBMS to stop, or it may be the result of the user detecting an error during the transaction and aborting the transaction before it completes. In all these cases, the DBMS must provide a mechanism to recover the database to a consistent state.

(6) Authorization services

A DBMS must furnish a mechanism to ensure that only authorized users can access the database.

It is not difficult to envisage instances where we would want to prevent some of the data stored in the database from being seen by all users. For example, we may want only branch managers to see salary-related information for staff and prevent all other users from seeing this data. Additionally, we may want to protect the database from unauthorized access. The term **security** refers to the protection of the database against unauthorized access, either intentional or accidental. We expect the DBMS to provide mechanisms to ensure the data is secure.

(7) Support for data communication

A DBMS must be capable of integrating with communication software.

Most users access the database from workstations. Sometimes these workstations are connected directly to the computer hosting the DBMS. In other cases, the workstations are at remote locations and communicate with the computer hosting the DBMS over a network. In either case, the DBMS receives requests as **communications messages** and responds in a similar way. All such transmissions are handled by a Data Communication Manager (DCM). Although the DCM is not part of the DBMS, it is necessary for the DBMS to be capable of being integrated with a variety of DCMs if the system is to be commercially viable. Even DBMSs for personal computers should be capable of being run on a local area network so that one centralized database can be established for users to share, rather than having a series of disparate databases, one for each user. This does not imply that the database has to be distributed across the network; rather that users should be able to access a centralized database from remote locations. We refer to this type of topology as *distributed processing.*

(8) Integrity services

A DBMS must furnish a means to ensure that both the data in the database and changes to the data follow certain rules.

Database integrity refers to the correctness and consistency of stored data: it can be considered as another type of database protection. While integrity is related to security, it has wider implications: integrity is concerned with the quality of data itself. Integrity is usually expressed in terms of *constraints*, which are consistency rules that the database is not permitted to violate. For example, we may want to specify a constraint that no member of staff can manage more than 100 properties at any one time. Here, we would want the DBMS to check when we assign a property to a member of staff that this limit would not be exceeded and to prevent the assignment from occurring if the limit has been reached.

(9) Services to promote data independence

A DBMS must include facilities to support the independence of programs from the actual structure of the database.

We have already discussed the concept of **data independence**. Data independence is normally achieved through a view or subschema mechanism. Physical data independence is easier to achieve: there are usually several types of change that can be made to the physical characteristics of the database without affecting the views. However, complete logical data independence is more difficult to achieve. The addition of a new entity, attribute, or relationship can usually be accommodated, but not their removal. In some systems, any type of change to an existing component in the logical structure is prohibited.

(10) Utility services

A DBMS should provide a set of utility services.

Utility programs help the DBA to administer the database effectively. Some utilities work at the external level, and consequently can be produced by the DBA. Other utilities work at the internal level and can be provided only by the DBMS vendor. Examples of utilities of the latter kind are:

- import facilities, to load the database from flat files, and export facilities, to unload the database to flat files;
- monitoring facilities, to monitor database usage and operation;
- statistical analysis programs, to examine performance or usage statistics;
- index reorganization facilities, to reorganize indexes and their overflows;
- garbage collection and reallocation, to remove deleted records physically from the storage devices, to consolidate the space released, and to reallocate it where it is needed.