# Installing the DBMS

Once the DBMS has been chosen, you will need to install it. Installing a DBMS is not as simple as popping a CD into a drive and letting the software install itself (or, for you mainframe folks, just using IEBGENER to copy it from a tape).

A DBMS is a complex piece of software that requires up-front planning for installation to be successful. You will need to understand the DBMS requirements and prepare the environment for the new DBMS.

## DBMS Installation Basics

The very first thing to do when you install a DBMS for the first time is to understand the prerequisites. Every DBMS comes with an installation manual or guide containing a list of the operating requirements that must be met for the DBMS to function properly.

Examples of prerequisites include ensuring that an appropriate version of the operating system is being used, verifying that there is sufficient memory to support the DBMS, and ensuring that any related software to be used with the DBMS is the proper version and maintenance level.

Once the basics are covered, read the installation guide from cover to cover. Make sure that you understand the process before you even begin to install the DBMS.

Quite a few preparations need to be made before installing a DBMS, and reading about them before you start will ensure a successful installation. Review how the installation program or routine for the DBMS operates, and follow the explicit instructions in the installation guide provided with the DBMS software.

You additionally might want to work closely with the DBMS vendor during an initial installation to ensure

that your plans are sound. In some cases, working with a local, experienced vendor or consultant can be beneficial to avoid installation and configuration errors.

The remainder of this section will discuss some of the common preparations that are required before a DBMS can be installed. If the DBMS is already operational and you are planning to migrate to a new DBMS release,

refer to the section “Upgrading DBMS Versions and Releases.”

## Hardware Requirements

Every DBMS has a basic CPU requirement, meaning a CPU version and minimum processor speed required for the DBMS to operate. Additionally, some DBMSs specify hardware models that are required or unsupported.

Usually the CPU criterion will suffice for an Intel environment, but in a mainframe or enterprise server environment the machine model can make a difference with regard to the DBMS features supported. For example, certain machines have built-in firmware that can be exploited by the DBMS if the firmware is available.

Furthermore, each DBMS offers different “flavors” of its software for specific needs. (I use “flavor” as opposed to “version” or “release,” which specify different iterations of the same DBMS.) Different flavors of the DBMS (at the same release level) are available for specific environments such as parallel processing, pervasive computing (such as handheld devices), data warehousing, and/or mobile computing. Be sure to choose the correct DBMS for your needs and to match your hardware to the requirements of the DBMS.

## Storage Requirements

A DBMS requires disk storage to run. And not just for the obvious reason to create databases that store data. A DBMS will use disk storage for the indexes to be defined on the databases as well as for the following items:

• The system catalog or data dictionary used by the DBMS to manage and track databases and related information. The more database objects you plan to create, the larger the amount of storage required by the system catalog.

• Any other system databases required by the DBMS, for example, to support distributed connections or management tools.

• Log files that record all changes made to every database. These include active logs, archive logs, rollback segments, and any other type of change log required by the DBMS.

• Start-up or control files that must be accessed by the DBMS when it is started or initialized.

• Work files used by the DBMS to sort data or for other processing needs.

• Default databases used by the DBMS for system structures or as a default catchall for new database objects as they are created.

• Temporary database structures used by the DBMS (or by applications accessing databases) for transient data that is not required to be persistent but needs reserved storage during operations (such as rebuilding clustered indexes on Microsoft SQL Server).

• System dump and error-processing files.

• DBA databases used for administration, monitoring, and tuning—for example, DBA databases used for testing new releases, migration scripts, and so on. Be sure to factor in every storage requirement of the DBMS and reserve the appropriate storage. Also, be aware that the DBMS will use many of these databases and file structures concurrently.

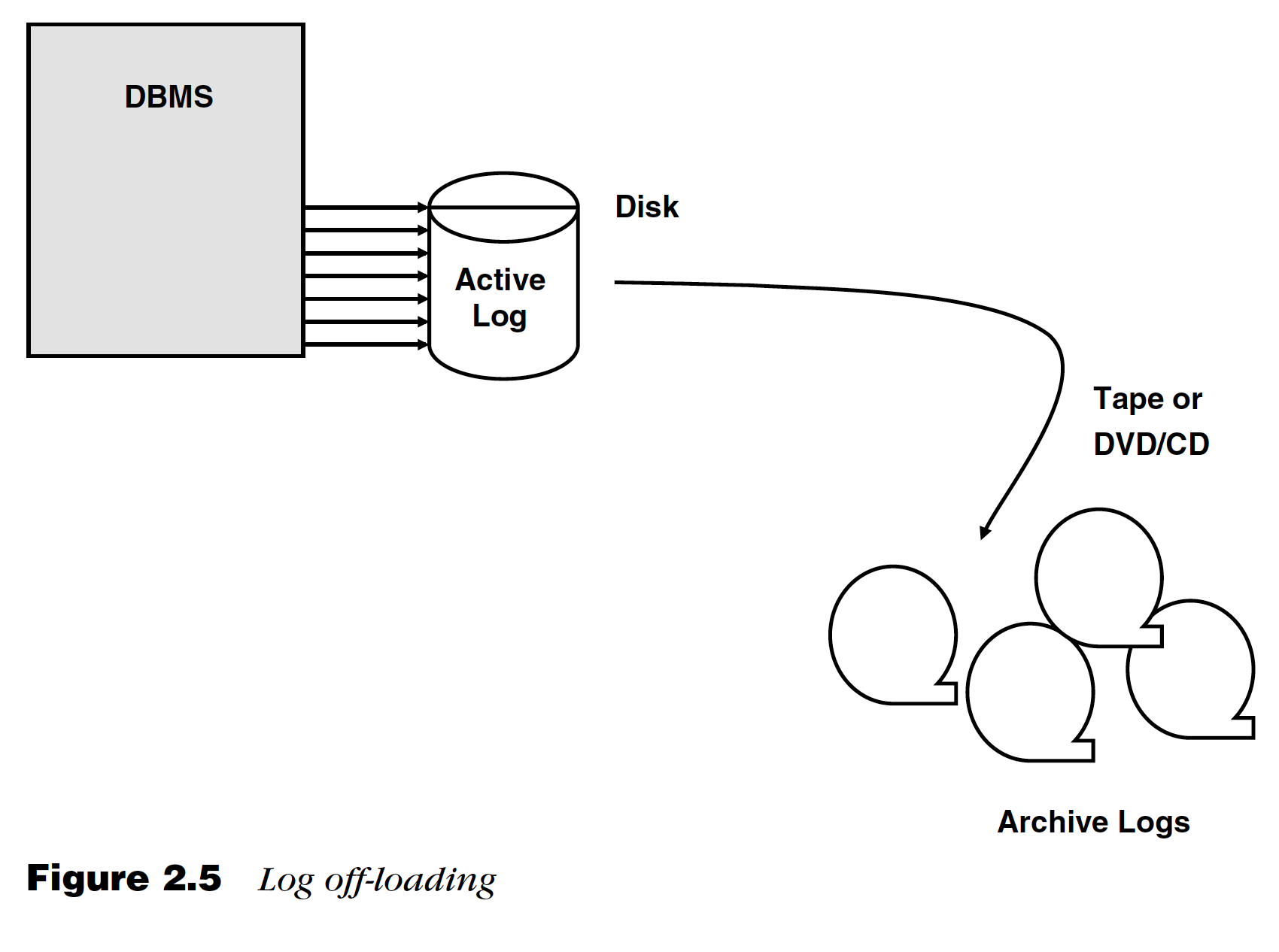
Therefore, it is a good idea to plan on using multiple storage devices even if you will not fill them to capacity.

Proper database and file placement will enable the DBMS to operate more efficiently because concurrent activities will not be constrained by the physical disk as data is accessed.

Disk storage is not the only requirement of a DBMS. Tape or optical discs (such as DVDs and CDs) are also required for tasks such as database backups and log off-loading. When the active log file fills up, the log records must be off-loaded to an archive log either on disk or on tape, as shown in Figure 2.5. Depending on the DBMS being used and the features that have been activated, this process may be automatic or manual.

**The archive log** files must be retained for recovery purposes, and even if originally storedon disk, they must eventually be migrated to an external storage mechanismfor safekeeping.

Plan on maintaining multiple tape or CD/DVD drives to enable the DBMS to run concurrent multiple processes that require external storage, such as concurrent database backups. Database outages can occur if you single-thread your database backup jobs using a single drive.



## Memory Requirements

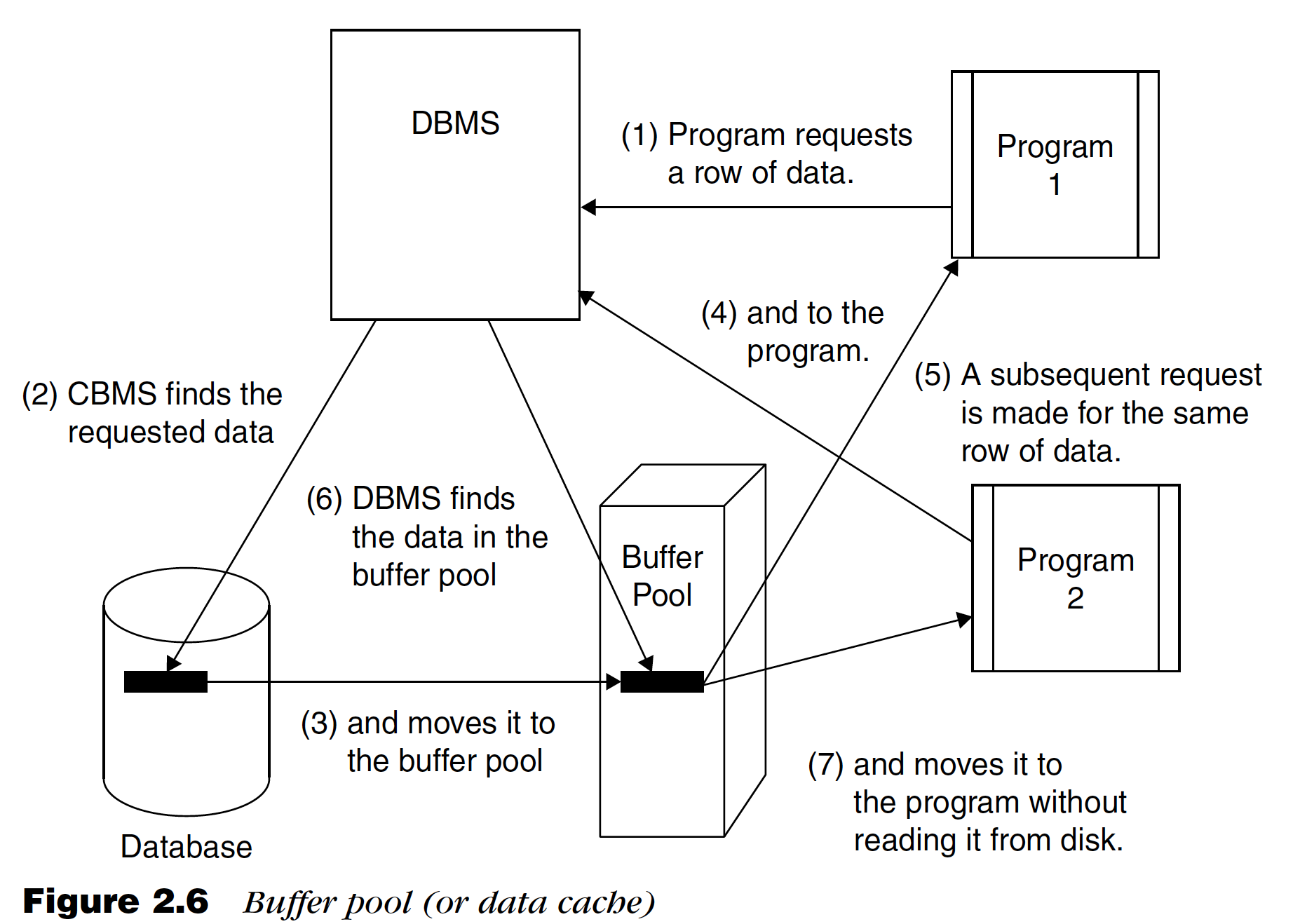
Relational DBMSs, as well as their databases and applications, love memory. A DBMS requires memory for basic functionality and will use it for most internal processes such as maintaining the system global area and performing many DBMS tasks.

A DBMS requires a significant amount of memory to cache data in memory structures in order to avoid I/O. Reading data from a disk storage device is always more expensive and slower than moving the data around in memory.

Figure 2.6 shows how the DBMS uses a memory structure called a buffer pool or data cache to reduce physical I/O requests. By caching data that is read into a buffer pool, the DBMS can avoid I/O for subsequent requests for the same data, as long as it remains in the buffer pool. In general, the larger the buffer pool, the longer the data can remain in memory and the better overall database processing will perform.

Besides data, the DBMS will cache other structures in memory. Most DBMSs set aside memory to store program structures required by the DBMS to process database requests.3 The program cache stores things like “compiled” SQL statements, database authorizations, and database structure blocks that are used by programs as they are executed.

When these structures are cached, database processing can be optimized because additional I/O requests to access them from a physical storage device are avoided. Memory is typically required by the DBMS to support other features such as handling lock requests, facilitating distributed data requests, sorting data, optimizing processes, and processing SQL. Ensure that the DBMS has a more-than-adequate supply of memory at its disposal. This will help to optimize database processing and minimize potential problems.



# Configuring the DBMS

Configuring the system parameters of the DBMS controls the manner in which the DBMS functions and the resources made available to it.

Each DBMS allows its system parameters to be modified in different ways, but the installation process usually sets the DBMS system parameters by means of radio buttons, menus, or panel selections. During the installation process, the input provided to the installation script will be used to establish the initial settings of the system parameters.

Each DBMS also provides a method to change the system parameters once the DBMS is operational. Sometimes you can use DBMS commands to set the system’s parameters; sometimes you must edit a file that contains the current system parameter settings. If you must edit a file, be very careful: An erroneous system parameter setting can be fatal to the operational status of the DBMS.

What do the system parameters control? Well, for example, system parameters control DBA authorization to the DBMS and the number of active database logs; system parameters set the amount of memory used for

data and program caching and turn DBMS features on or off.

Although every DBMS has system parameters that control its functionality, each DBMS has a different method of setting and changing the values. And, indeed, each DBMS has different specifications that can be set using system parameters. Beware of simply using default system parameters when installing the database system software. Although using defaults can save time and make for an easier installation, it can also result in subsequent problems.

Most DBMSs are poorly served, in the long run, by default settings and, in some cases, can experience worsening performance over time because resources were not preallocated during installation or setup.

Be sure to understand fully the parameters used by your DBMS. Failure to do so can result in an incorrectly configured database environment, which can cause performance problems, data integrity problems, or even

DBMS failure.

## Connecting the DBMS to Supporting Infrastructure Software.

Part of the DBMS installation process is the connection of the DBMS to other system software components that must interact with the DBMS. Typical infrastructure software that may need to be configured to work with the DBMS includes networks, transaction processing monitors, message queues, other types of middleware, programming languages, systems management software, operations and job control software, Web servers, and application servers.

Each piece of supporting infrastructure software will have different requirements for interfacing with the DBMS. Typical configuration procedures can include installing DLL files, creating new parameter files to establish connections, and possibly revisiting the installation procedures for the supporting software to install components required to interact with the DBMS.

## Installation Verification

After installing the DBMS, you should run a battery of tests to verify that the DBMS has been properly installed and configured. Most DBMS vendors supply sample programs and installation verification procedures for this purpose. Additionally, you can ensure proper installation by testing the standard interfaces to the DBMS. One standard interface supported by most DBMSs is an interactive SQL interface where you can submit SQL statements directly to the DBMS.

Create a set of SQL code that comprises SELECT, INSERT, UPDATE, and DELETE statements issued against sample databases. Running such a script after installation helps you to verify that the DBMS is installed correctly and operating as expected.

Furthermore, be sure to verify that all required connections to supporting software are operational and functioning properly. If the DBMS vendor does not supply sample programs, you may need to create and run simple test programs for each environment to ensure that the supporting software connections are functioning correctly with the DBMS.

## DBMS Environments

Generally, installing a DBMS involves more than simply installing one instance or subsystem. To support database development, the DBA needs to create multiple DBMS environments to support, for example, testing, quality assurance, integration, and production work. Of course, it is possible to support multiple environments in a single DBMS instance, but it is not prudent. Multiple DBMS installations are preferable to support multiple development environments for a single database. This minimizes migration issues and won’t require complex database naming conventions to support. Furthermore, segregating database instances makes testing, tuning, and monitoring easier.

## Upgrading DBMS Versions and Releases

Change is a fact of life, and each of the major DBMS products changes quite rapidly. A typical release cycle for DBMS software is 18 to 24 months for major releases, with constant bug fixes and maintenance updates delivered between major releases. Indeed, keeping DBMS software up-to-date can be a full-time job.

The DBA must develop an approach to upgrading DBMS software that conforms to the organization’s needs and minimizes business disruptions due to outages and database unavailability. You may have noticed that I use the terms version and release somewhat interchangeably. That is fine for a broad discussion of DBMS upgrades, but a more precise definition is warranted. For a better discussion of the differences between a version and a release, please refer to the sidebar.

A DBMS version upgrade can be thought of as a special case of a new installation. All the procedures required of a new installation apply to an upgrade: You must plan for appropriate resources, reconsider all system parameters, and ensure that all supporting software is appropriately connected.

However, another serious issue must be planned for: existing users and applications. An upgrade needs to be planned to cause as little disruption to the existing users as possible. Furthermore, any additional software

that works with the DBMS (such as purchased applications, DBA tools, utilities, and so on) must be verified to be compatible with the new DBMS version. Therefore, upgrading can be a tricky and difficult task.

## Features and Complexity

Perhaps the biggest factor in determining when and how to upgrade to a new DBMS release is the functionality supported by the new release. Tightly coupled to functionality is the inherent complexity involved in supporting and administering new features.

It is more difficult to delay an upgrade if application developers are clamoring for new DBMS features. If DBMS functionality can minimize the cost and effort of application development, the DBA group will feel pressure to migrate swiftly to the new release. An additional factor that will coerce rapid adoption of a new release is when DBMS problems are fixed in the new release (instead of through regular maintenance fixes).

Regardless of a new release’s “bells and whistles,” certain administration and implementation details must be addressed before upgrading. The DBA group must ensure that standards are modified to include the new

features, educate developers and users as to how new features work and should be used, and prepare the infrastructure to support the new DBMS functionality.

The types of changes required to support the new functionality must be factored into the upgrade strategy. When the DBMS vendor makes changes to internal structures, data page layouts, or address spaces, the risks of upgrading are greater. Additional testing is warranted in these situations to ensure that database utilities, DBA tools, and data extraction and movement tools still work with the revised internal structures.

## Complexity of the DBMS Environment

The more complex your database environment is, the more difficult it will be to upgrade to a new DBMS release. The first complexity issue is the size of the environment. The greater the number of database servers, instances, applications, and users, the greater the complexity. Additional concerns include the types of applications being supported. A DBMS upgrade is easier to implement if only simple, batch-oriented applications are involved. As the complexity and availability requirements of the applications increase, the difficulty of upgrading also increases.

Location of the database servers also affects the release upgrade strategy. Effectively planning and deploying a DBMS upgrade across multiple database servers at various locations supporting different lines of business

is difficult. It is likely that an upgrade strategy will involve periods of supporting multiple versions of the DBMS at different locations and for different applications. Supporting different versions in production should be avoided, but that is not always possible.

Finally, the complexity of the applications that access your databases must be considered. The more complex your applications are, the more difficult it will be to ensure their continuing uninterrupted functionality when the DBMS is modified.

Complexity issues include the following:

• Usage of stored procedures and user-defined functions.

• Complexity of the SQL—the more tables involved in the SQL and the more complex the SQL features, the more difficult it becomes to ensure that access path changes do not impact performance.

• Client/server processing—network usage and usage of multiple tiers complicates testing the new DBMS release.

• Applications that are designed, coded, and generated by a framework or an IDE (for example, Hibernate) may have additional components that need to be tested with a new DBMS release.

• Integration with other infrastructure software such as message queues and transaction processors can complicate migration because new versions of these products may be required to support

the new DBMS release.

• The language used by the programs might also impact DBMS release migration due to different support for compiler versions, changes to APIs (application programming interfaces), or new ways of embedding SQL into application programs.

## Database Standards and Procedures

Before a newly installed DBMS can be used effectively, standards and procedures must be developed for database usage. Studies have shown that companies with high levels of standardization reduce the cost of supporting end users by as much as 35 percent or more as compared to companies with low levels of standardization.

Standards are common practices that ensure the consistency and effectiveness of the database environment, such as database naming conventions. Procedures are defined, step-by-step instructions that direct the processes required for handling specific events, such as a disaster recovery plan.

Failure to implement database standards and procedures will result in a database environment that is confusing and difficult to manage. The DBA should develop database standards and procedures as a component of corporate-wide IT standards and procedures. They should be stored together in a central location as a printed document, in an online format, or as both. Several vendors offer “canned” standards and procedures that can be purchased for specific DBMS products.

# Summary

Give summary about all the above topics.