Creating the Database Environment

One of the primary tasks associated with the job of DBA is the process of choosing and installing a DBMS. Choosing a suitable DBMS for enterprise database management is not as difficult as it used to be as the number of major DBMS vendors has dwindled due to industry consolidation and domination of the sector by a few very large players. However, establishing a usable database environment requires a great deal of skill, knowledge, and consideration.

Large and medium-size organizations typically run multiple DBMS products, from as few as two to as many as ten. For example, it is not uncommon for a large company to use IBM's DB2 on the mainframe, Oracle and MySQL on several different UNIX servers, Microsoft SQL Server on Windows servers, as well as other DBMS products such as PostgreSQL on various platforms, not to mention single-user PC DBMS products such as Microsoft Access.

In order not to complicate the DBA's job, the DBA group should be empowered to make the DBMS decisions for the organization. No business unit should be allowed to purchase a DBMS without the permission of the DBA group.

Choosing a DBMS

The DBA should set a policy regarding the DBMS products to be supported within the organization. Whenever possible, the policy should minimize the number of different DBMS products as most of the major DBMS products have similar features.

When choosing a DBMS, it is wise to select a product from one of the largest vendors having the most heavily implemented and supported products on the market. These include IBM's DB2 and Oracle. Both are popular and support just about any type of database. Another major player is Microsoft SQL Server, but only for Windows platforms. DB2 and Oracle run on multiple platforms ranging from mainframe to UNIX, as well as Windows and even handheld devices. Choosing a DBMS other than these three should be done only under specific circumstances.

If the company is heavily into the open source software movement, PostgreSQL, or MySQL might be viable options. And there are a variety of NoSQL DBMS offerings available, too, such as Hadoop, Cassandra, and MongoDB.

Factors to consider when choosing a DBMS

1. **Operating system support**

Does the DBMS support the operating systems in use at your organization, including the versions that you are currently using and plan on using?

2. **Type of organization**

Take into consideration the corporate philosophy when you choose a DBMS. Some organizations are very conservative and like to keep a tight rein on their environments; these organizations tend to gravitate toward traditional mainframe environments. Government operations, financial institutions, and insurance and health companies usually tend to be conservative. More-liberal organizations are often willing to consider alternative architectures. It is not uncommon for manufacturing companies, dot-coms, and universities to be less conservative. Finally, some companies just do not trust Windows as a mission-critical environment and prefer to use UNIX; this rules out some database vendors (Microsoft SQL Server, in particular).

3. **Benchmarks**

What performance benchmarks are available from the DBMS vendor and other users of the DBMS? The Transaction Processing Performance Council (TPC) publishes official database performance benchmarks that can be used as a guideline for the basic overall performance of many different types of database processing.

4. **Scalability**

Does the DBMS support the number of users and database sizes you intend to implement? How are large databases built, supported, and maintained—easily or with a lot of pain? Are there independent users who can confirm the DBMS vendor's scalability claims?

5. **Availability of supporting software tools**

Are the supporting tools you require available for the DBMS? These items may include query and analysis tools, data warehousing support tools, database administration tools, backup and recovery tools, performance-monitoring tools, capacity-planning tools, database utilities, and support for various programming languages.

6. **Technicians**

Is there a sufficient supply of skilled database professionals for the DBMS? Consider your needs in terms of DBAs, technical support personnel (system programmers and administrators, operations analysts, etc.), and application programmers.

7. **Cost of ownership**

What is the total cost of ownership of the DBMS? DBMS vendors charge wildly varying prices for their technology. Total cost of ownership should be calculated as a combination of the license cost of the DBMS; the license cost of any required supporting software; the cost of database professionals to program, support, and administer the DBMS; and the cost of the computing resources required to operate the DBMS.

8. **Release schedule**

How often does the DBMS vendor release a new version? Some vendors have rapid release cycles, with new releases coming out every 12 to 18 months. This can be good or bad, depending on your approach. If you want cutting-edge features, a rapid release cycle is good. However, if your shop is more conservative, a DBMS that changes frequently can be difficult to support. A rapid release cycle will cause conservative organizations either to upgrade more frequently than they would like or to live with outdated DBMS software that is unlikely to have the same level of support as the latest releases.

9. **Reference customers**

Will the DBMS vendor supply current user references? Can you find other users on your own who might provide more impartial answers? Speak with current users to elicit issues and concerns you may have overlooked. How is support? Does the vendor respond well to problems? Do things generally work as advertised? Are there a lot of bug fixes that must be applied continuously? What is the quality of new releases? These questions can be answered only by the folks in the trenches.

Database administration tools

Often, the DBMS software comes with certain tools to help DBAs manage the DBMS. Such tools are called native tools. For example, Microsoft SQL Server comes with SQL Server Management Studio and Oracle has tools such as SQL*Plus and Oracle Enterprise Manager/Grid Control. In addition, 3rd parties such as BMC, Quest Software, Embarcadero Technologies, and SQL Maestro Group offer GUI tools to

monitor the DBMS and help DBAs carry out certain functions inside the database more easily.
Another kind of database software exists to manage the provisioning of new databases and the management of existing databases and their related resources. The process of creating a new database can consist of hundreds or thousands of unique steps from satisfying prerequisites to configuring backups where each step must be successful before the next can start. A human cannot be expected to complete this procedure in the same exact way time after time - exactly the goal when multiple databases exist. As the number of DBAs grows, without automation the number of unique configurations frequently grows to be costly/difficult to support. All of these complicated procedures can be modeled by the best DBAs into database automation software and executed by the standard DBAs. Software has been created specifically to improve the reliability and repeatability of these procedures such as Stratavia's Data Palette and GridApp Systems Clarity.

DBMS Architectures

The supporting architecture for the DBMS environment is very critical to the success of the database applications. One wrong choice or poorly implemented component of the overall architecture can cause poor performance, downtime, or unstable applications.

Today the IT infrastructure is distributed and heterogeneous. The overall architecture will probably consist of multiple platforms and interoperating system software. The DBA has to make sure that the DBMS selected is appropriate for the nature and type of processing they plan to implement.

Levels of DBMS architecture

1. **Enterprise DBMS**

An enterprise DBMS is designed for scalability and high performance. An enterprise DBMS must be capable of supporting very large databases, a large number of concurrent users, and multiple types of applications. The enterprise DBMS runs on a large-scale machine, typically a mainframe or a high-end server running UNIX, Linux, or Windows Server. Furthermore, an enterprise DBMS offers all the “bells and whistles” available from the DBMS vendor.

Multiprocessor support, support for parallel queries, and other advanced DBMS features are core components of an enterprise DBMS.

2. **Departmental DBMS**

A departmental DBMS, sometimes referred to as a workgroup DBMS, serves the middle ground. The departmental DBMS supports small to medium-size workgroups within an organization; typically, it runs on a UNIX, Linux, or Windows server. The dividing line between a departmental database server and an enterprise database server is quite gray. Hardware and software upgrades can allow a departmental DBMS to tackle tasks that previously could be performed only by an enterprise DBMS. The steadily falling cost of departmental hardware and software components further contributes to lowering the total cost of operation and enabling a workgroup environment to scale up to serve the enterprise.

3. **Personal DBMS**

A personal DBMS is designed for a single user, typically on a low- to medium-powered PC platform. Microsoft Access and SQLite are examples of personal database software. Of course, the major DBMS vendors also market personal versions of their higher-powered solutions, such as Oracle Database Personal Edition and DB2 Personal Edition. Sometimes the low cost of a personal DBMS results in a misguided attempt to choose a personal DBMS for a departmental or enterprise solution. However, do not be lured by the low cost. A personal DBMS product is suitable only for very small-scale projects and should never be deployed for multi user applications.

4. **Mobile DBMS**

A mobile DBMS is a specialized version of a departmental or enterprise DBMS. It is designed for remote users who are not usually connected to the network. The mobile DBMS enables local database access and modification on a laptop or handheld device. Furthermore, the mobile DBMS provides a mechanism for synchronizing remote database changes to a centralized enterprise or departmental database server.

Hardware Issues

When establishing a database environment for application development, selecting the DBMS is only part of the equation. The hardware and operating system on which the DBMS will run will greatly impact the reliability, availability, and scalability of the database environment. That is not to say everything should run on a mainframe; other issues such as cost, experience, manageability, and the needs of the applications to be

developed must be considered. The bottom line is that you must be sure to factor hardware platform and operating system constraints into the DBMS selection criteria.

Cloud Database Systems

Cloud computing is increasing in usage, especially at small to medium-size businesses. A cloud implementation can be more cost effective than building an entire local computing infrastructure that requires management and support. A cloud database system delivers DBMS services over the Internet. The trade-off essentially comes down to trusting a cloud provider to store and manage your data in return for minimizing database administration and maintenance cost and effort.

There is no need to install, set up, patch, or manage the DBMS software because the cloud provider manages and cares for these administrative tasks. Of course, the downside is that your data is now stored and controlled by an external agent —the cloud provider. Another inherent risk of cloud computing is the possibility of malicious agents posing as legitimate customers.

An example of a cloud database platform is Microsoft SQL Azure. It is built on SQL Server technologies and is a component of the Windows Azure platform.

Installing the DBMS

Once the DBMS has been chosen, you will need to install it. 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.

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.

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 system catalogue, system databases, log files, start-up files, temporary databases, e.t.c.

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.

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.

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. Typical configuration procedures can include creating new parameter files to establish connections.

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.

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.

Upgrading DBMS Versions and Releases

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. 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.

Version or Release?

Vendors typically make a distinction between a version and a release of a software product. A new version of software is a major concern, with many changes and new features. A release is typically minor, with fewer changes and not as many new features. For example, moving from Version 10g of Oracle Database to Version 11g would be a major change—a version change. However, an in-between point such as Oracle Database 11g Release 2 would be considered a release—consisting of a smaller number of changes.

Upgrading to a new DBMS release offers both rewards and risks.

Benefits of moving to a new DBMS release

- Developers can avail themselves of new features and functionality delivered only in the new release. If development requires a new feature, or can simply benefit from a new feature, program development time can be reduced or made more cost-effective.
- For purchased applications, the application vendor may require a specific DBMS version or release for specific versions of its application to enable specific functionality within the application.
- New DBMS releases usually deliver enhanced performance and availability features that can optimize existing applications. Sometimes a new DBMS release is required to scale applications to support additional users or larger amounts of data.
- DBMS vendors often provide better support and respond to problems faster for a new release of their software. DBMS vendors would not like to allow bad publicity about bugs in a new and heavily promoted version of their products.

- Cost savings may accrue by upgrading to a new DBMS release. Some vendors charge additionally when a company uses multiple versions of a DBMS, such as the new version in a test environment and the old in production. When both are migrated to the same version, the price tag for the DBMS sometimes can be reduced.
- Production migration to a new DBMS release will align the test and production database environments, thereby providing a consistent environment for development and implementation. If a new release is running in the test environment for too long, database administration and application development tasks become more difficult because the test databases will operate differently from the production databases.

Risks of upgrading to a new DBMS release

- An upgrade to the DBMS usually involves some level of disruption to business operations. At a minimum, databases will not be available while the DBMS is being upgraded. This can result in downtime and lost business opportunities if the DBMS upgrade occurs during normal business hours (or if there is no planned downtime). Clustered database implementations may permit some database availability while individual database clusters are migrated to the new DBMS version.
- Other disruptions can occur, such as having to convert database structures or discovering that previously supported features were removed from the

new release (thereby causing application errors). Delays to application implementation timelines are another possibility.

- The cost of an upgrade can be a significant barrier to DBMS release migration. First, the cost of the new version or release must be budgeted for (price increases for a new DBMS version can amount to as much as 10 to 25 percent). The upgrade cost must also factor in the costs of planning, installing, testing, and deploying not just the DBMS but also any applications that use databases. Finally, be sure to include the cost of any new resources (such as memory, storage, additional CPUs) required to use the new features delivered by the new DBMS version.
- DBMS vendors usually tout the performance gains that can be achieved with a new release. However, when SQL optimization techniques change, it is possible that a new DBMS release will generate SQL access paths that perform worse than before. DBAs must implement a rigorous testing process to ensure that new access paths are helping, not harming, application performance. When performance suffers, application code may need to be changed—a very costly and time consuming endeavor. A rigorous test process should be able to catch most of the access path changes in the test environment.
- New DBMS releases may cause features and syntax that are being used in existing applications to be deprecated. When this occurs, the applications must be modified before migration to the new release can proceed.
- To take advantage of improvements implemented in a new DBMS release, the DBA may have to apply some invasive changes. For example, if the new version increases the maximum size for a database object, the DBA may have to drop and recreate that object to take advantage of the new maximum. This will be the case when the DBMS adds internal control structures to facilitate such changes.
- Supporting software products may lack immediate support for a new DBMS release. Supporting software includes the operating system, transaction processors, message queues, purchased applications, DBA tools, development tools, and query and reporting software.

Factors to consider for an effective DBMS upgrade strategy

1. **Features and Complexity**

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. If

DBMS functionality can minimize the cost and effort of application development, the DBA will feel pressure to migrate swiftly to the new release, especially if DBMS problems are fixed in the new release.

2. 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. Location of the database servers also affects the release upgrade strategy.

3. Reputation of the DBMS Vendor

DBMS vendors have different reputations for technical support, fixing bugs, and responding to problems, which is why customer references are so important when choosing a database. The better the reputation of the vendor, the greater the likelihood of organizations rapidly adopting a new release.

4. Support Policies of the DBMS

As new releases are introduced, DBMS vendors will retire older releases and no longer support them. The length of time that the DBMS vendor will support an old release must be factored into the DBMS release migration strategy. You should never run a DBMS release in production that is no longer supported by the vendor. If problems occur, the DBMS vendor will not be able to resolve them for you.

5. Organization Style

Every organization displays characteristics that reveal its style when it comes to adopting new products and technologies. Some organizations are technology driven and, as such, are more likely to risk using new and unproven technologies to try to gain a competitive advantage. Some organizations are less willing to take risks but will adopt new technologies once others have shaken out the bugs. However other organizations are very conscious of cost and averse to risk, and will lag behind the majority when it comes to migrating to new technology.

6. DBA Staff Skill Set

Upgrading the DBMS is easier if your DBA staff is highly skilled and/or experienced. The risk of an upgrade increases as the skills of the DBA staff decrease. If your DBAs are not highly skilled, or have never migrated a DBMS to a new release, consider

augmenting your DBA staff with consultants for the upgrade. Deploying an integrated team of internal DBAs and consultants will ensure that your upgrade goes as smoothly as possible. Furthermore, the DBA staff will be better prepared to handle the future upgrades alone.

7. Platform Support

When a DBMS vendor unleashes a new release of its product, not all platforms and operating systems are immediately supported. The DBMS vendor usually first supports the platforms and operating systems for which it has the most licensed customers. The order in which platforms are supported for a new release is likely to differ for each DBMS vendor.

8. Supporting Software

Carefully consider the impact of a DBMS upgrade on any supporting software. Supporting software includes purchased applications, DBA tools, reporting and analysis tools, and query tools. Each software vendor will have a different time frame for supporting and exploiting a new DBMS release.

9. Fallback Planning

Each new DBMS version or release should come with a manual that outlines the new features of the release and describes the fallback procedures to return to a prior release of the DBMS. Be sure to review the fallback procedures provided by the DBMS vendor in its release guide. You may need to return to a previous DBMS release if the upgrade contains a bug, performance problems ensue, or other problems arise during or immediately after migration. Keep in mind that fallback is not always an option for every new DBMS release.

Database Change Management

An ever-changing market causes businesses to have to continually adapt. Businesses are striving to meet constantly changing customer expectations while trying to sustain revenue growth and profitability at the same time. To keep pace, businesses must constantly update and enhance products and services to meet and exceed the offerings of competitors.

Change is inevitable but necessary for business survival and success.

Factors that lead to change in database structures

Many factors conspire to force us into changing our database structures.

- Changes to application programs that require additional or modified data elements
- Performance modifications and tweaks to make database applications run faster
- Regulatory changes that mandate storing new types of data, or the same data for longer periods of time
- Changes to business practices, requiring new types of data
- Technological changes that enable databases to store new types of data and more data than ever before

Change Management Requirements

To successfully implement effective change management, understanding a set of basic requirements is essential. To ensure success, the following factors need to be incorporated into your change management discipline.

Factors to incorporate into the change management discipline

1. **Proactivity**.Proactive change, which can eliminate future problems, is an organization's most valuable type of change. The earlier in the development cycle that required changes are identified and implemented, the lower the overall cost of the change will be.
2. **Intelligence**.When implementing a change, every aspect of the change needs to be examined, because it could result in an unanticipated cost to the company. The impact of each change must be examined and incorporated into the change process, because a simple change in one area may cause a complex change in another area. Intelligence in the change management process often requires a thorough analysis that includes an efficient and low-risk implementation plan. True intelligence also requires the development of a contingency plan, should the change or set of changes not perform as projected.
3. **Planning analysis**.Planning maximizes the effectiveness of change. A well-planned change saves time. It is always easier to do it right the first time than to do it again after the first change proves to be less than effective. An effective organization will have a thorough understanding of the impact of each change

before allocating resources to implement the change. A well-planned change saves time.

4. **Impact analysis**.Comprehensive impact and risk analyses allow the organization to examine the entire problem, and the risk involved, to determine the best course of action. A single change usually can be accomplished in many different ways. However, the impact of each change may be considerably different. Some changes involve more risks: failure, undue difficulty, need for additional changes, downtime, and so on. All considerations are important when determining the best approach to implementing change.
5. **Automation**.With limited resources and a growing workload, automating the change process serves to reduce human error and to eliminate more menial tasks from overburdened staff.
6. **Standardization of procedure**.Attrition, job promotions, and job changes require organizations to standardize processes to meet continued productivity levels. An organized and thoroughly documented approach to completing a task reduces the learning curve, as well as the training time.
7. **Reliable and predictable process**.When creating any deliverable, a business needs to know that none of the invested effort is wasted. Because time is valuable, a high level of predictability will help to ensure continued success and profitability. Reliability and predictability are key factors in producing a consistently high-quality product.
8. **Availability**.Most changes require downtime to implement the change. Applications must come down—the same is true of databases. However, high availability is required of most applications these days, especially for an e-business. This is fast becoming a requirement in the Internet age. Reducing the amount of downtime required to make a change will increase application availability.
9. **Quick and efficient delivery**.Consumers demand quick turnaround for most products and services. Profitability is at its best when a product is first to market. Conversely, the cost of slow or inefficient delivery of products can be enormous. So, when implementing change, faster is better. The shorter the duration of an outage to accomplish the change, the quicker the system can be brought to market.

Types of Changes

Managing change is a big component of the DBA's job. In fact, if systems and databases could be installed into an environment that never changed, most of the DBA's job would vanish. However, things change. Business changes usually necessitate a change to application code or to database structure. Less obvious business changes also impact the database—for example, when the business grows

and more users are added, when additional data is stored, or when transaction volume grows. Additionally, technological changes such as upgrades to the DBMS and changes to hardware components impact the functionality of database software and therefore require DBA involvement.

1. **DBMS Software**
   The DBA must be prepared to manage the migration to new DBMS versions and releases. The complexity involved in moving from one version of a DBMS to another depends on the new features and functions supported by the new version. Additional complexity will be introduced if features are removed from the DBMS in a later version, because databases and programs may need to change if the removed features were being used. Furthermore, as functionality is added to, and removed from, the DBMS, the DBA must create the policies and procedures for the proper use of each new DBMS feature.

2. **Hardware Configuration**
   The DBMS may require hardware upgrades or configuration changes. The DBA will be expected to work in conjunction with the system programmers and administrators responsible for setting up and maintaining the hardware. At times the DBMS may require a different configuration from the one that is commonly used, thereby requiring the DBA to communicate to the SA the reason why a nonstandard configuration is required.

3. **Logical and Physical Design**
   When the database changes, it is important that the blueprints that define the database also change. This means that you need to keep the conceptual and logical data models synchronized with the physical database.

4. **Applications**
   Application changes need to be synchronized with database changes; however, this is easier said than done. Whenever changes are made to a physical database structure, application changes usually accompany those changes. For example, simply adding a column to a database table requires application software to populate, modify, and report on the data in the new column.

5. **Physical Database Structures**

The most complicated and time-consuming type of change for DBAs is planning, analyzing, and implementing changes to physical database structures. But most databases change over time—indeed, the database that remains static once implemented is very rare. So DBAs must be prepared to make changes to the databases under their care. Some changes are simple to implement, but others are very complex, error prone, and time-consuming.

Impact of Change on Database Structures

Relational databases are created using Data Definition Language (DDL) statements. DDL consists of three SQL verbs: CREATE, DROP, and ALTER. The CREATE statement is used to create a database object initially, and the DROP statement is used to remove a database object from the system. The ALTER statement is used to make changes to database objects.

Not every aspect of a database object can be changed by using the ALTER statement. Some types of changes require the database object to be dropped and recreated with the new parameters. The exact specifications for what can, and cannot, be changed using ALTER differ from DBMS to DBMS.

DATABASE SYSTEMS ARCHITECTURES

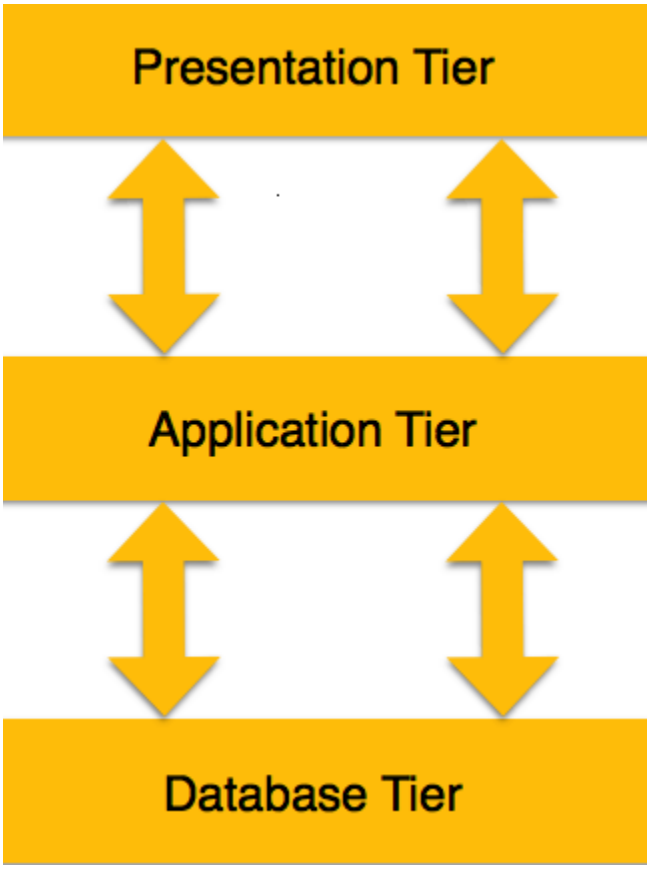
The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed, or replaced.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

3-tier Architecture

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.



Database (Data) Tier − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.

Application (Middle) Tier − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

User (Presentation) Tier − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

ORACLE DATABASE ARCHITECTURE

An Oracle database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management. In general, a server reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this is accomplished while delivering high performance. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

Oracle Database is the first database designed for enterprise grid computing, the most flexible and cost effective way to manage information and applications. Enterprise grid computing creates large pools of industry-standard, modular storage and servers. With this architecture, each new system can be rapidly provisioned from the pool of components. There is no need for peak workloads, because capacity can be easily added or reallocated from the resource pools as needed.

The database has **logical structures and physical structures**. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting the access to logical storage structures.

ORACLE PHYSICAL DATABASE STRUCTURES

The following sections explain the physical database structures of an Oracle database, including datafiles, redo log files, and control files.

1. Datafiles

Every Oracle database has one or more physical datafiles. The datafiles contain all the database data. The data of logical database structures, such as tables and indexes, is physically stored in the datafiles allocated for a database.

The characteristics of datafiles are:

A datafile can be associated with only one database.

Datafiles can have certain characteristics set to let them automatically extend when the database runs out of space.

One or more datafiles form a logical unit of database storage called a tablespace.

Data in a datafile is read, as needed, during normal database operation and stored in the memory cache of Oracle. For example, assume that a user wants to access some data in a table of a database. If the requested information is not already in the memory cache for the database, then it is read from the appropriate datafiles and stored in memory.

Modified or new data is not necessarily written to a datafile immediately. To reduce the amount of disk access and to increase performance, data is pooled in memory and written to the appropriate datafiles all at once, as determined by the database writer process (DBWn) background process.

2. Control Files

Every Oracle database has a control file. A control file contains entries that specify the physical structure of the database. For example, it contains the following information:

- Database name
- Names and locations of datafiles and redo log files
- Time stamp of database creation

When you mount and open the database instance, the control file is read. The entries in thecontrol file specify the physical files that make up the database. When you add additional filesto your database, the control file is automatically updated. The location of the control files isspecified in the CONTROL_FILES initialization parameter.

To protect against the failure of the database because of the loss of the control file, you mustmultiplex the control file on at least three different physical devices. By specifying multiplefiles through the initialization parameter, you enable the Oracle server to maintain multiplecopies of the control file.

Oracle can multiplex the control file, that is, simultaneously maintain a number of identical control file copies, to protect against a failure involving the control file.

Every time an instance of an Oracle database is started, its control file identifies the database and redo log files that must be opened for database operation to proceed. If the physical makeup of the database is altered (for example, if a new datafile or redo log file is created), then the control file is automatically modified by Oracle to reflect the change. A control file is also used in database recovery.

3. Redo Log Files

Every Oracle database has a set of two or more redo log files. The set of redo log files is collectively known as the redo log for the database. A redo log is made up of redo entries (also called redo records).

The primary function of the redo log is to record all changes made to data. If a failure prevents modified data from being permanently written to the datafiles, then the changes can be obtained from the redo log, so work is never lost.

To protect against a failure involving the redo log itself, Oracle allows a multiplexed redo log so that two or more copies of the redo log can be maintained on different disks.

The information in a redo log file is used only to recover the database from a system or media failure that prevents database data from being written to the datafiles. For example, if an unexpected power outage terminates database operation, then data in memory cannot be written to the datafiles, and the data is lost. However, lost data can be recovered when the database is opened, after power is restored. By applying the information in the most recent redo log files to the database datafiles, Oracle restores the database to the time at which the power failure occurred.

The process of applying the redo log during a recovery operation is called rolling forward.

4. Archive Log Files

You can enable automatic archiving of the redo log. Oracle automatically archives log files when the database is in ARCHIVELOG mode.

5. Parameter Files

Parameter files contain a list of configuration parameters for that instance and database.

Oracle recommends that you create a server parameter file (SPFILE) as a dynamic means of maintaining initialization parameters. A server parameter file lets you store and manage your initialization parameters persistently in a server-side disk file.

6. Alert and Trace Log Files

Each server and background process can write to an associated trace file. When an internal error is detected by a process, it dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database administrator, while other information is for Oracle Support Services. Trace file information is also used to tune applications and instances.

The alert file, or alert log, is a special trace file. The alert log of a database is a chronological log of messages and errors.

7. Backup Files

To restore a file is to replace it with a backup file. Typically, you restore a file when a media failure or user error has damaged or deleted the original file.

User-managed backup and recovery requires you to actually restore backup files before you can perform a trial recovery of the backups.

Server-managed backup and recovery manages the backup process, such as scheduling of backups, as well as the recovery process, such as applying the correct backup file when recovery is needed.

ORACLE LOGICAL DATABASE STRUCTURES

The logical storage structures, including data blocks, extents, and segments, enable Oracle to have fine-grained control of disk space use.

1. Tablespaces

A database is divided into logical storage units called tablespaces, which group related logical structures together. For example, tablespaces commonly group together all application objects to simplify some administrative operations.

Each database is logically divided into one or more tablespaces. One or more datafiles are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace. The combined size of the datafiles in a tablespace is the total storage capacity of the tablespace.

Every Oracle database contains a SYSTEM tablespace and a SYSAUX tablespace. Oracle creates them automatically when the database is created. The system default is to create a smallfile tablespace, which is the traditional type of Oracle tablespace. The SYSTEM and SYSAUX tablespaces are created as smallfile tablespaces.

Oracle also lets you create bigfile tablespaces. This allows Oracle Database to contain tablespaces made up of single large files rather than numerous smaller ones. This lets Oracle Database utilize the ability of 64-bit systems to create and manage ultralarge files. The consequence of this is that Oracle Database can now scale up to 8 exabytes in size. With Oracle-managed files, bigfile tablespaces make datafiles completely transparent for users. In other words, you can perform operations on tablespaces, rather than the underlying datafiles.

2. Online and Offline Tablespaces

A tablespace can be online (accessible) or offline (not accessible). A tablespace is generally online, so that users can access the information in the tablespace. However,

sometimes a tablespace is taken offline to make a portion of the database unavailable while allowing normal access to the remainder of the database. This makes many administrative tasks easier to perform.

3. Oracle Data Blocks

At the finest level of granularity, Oracle database data is stored in data blocks. One data block corresponds to a specific number of bytes of physical database space on disk. The standard block size is specified by the DB_BLOCK_SIZE initialization parameter. In addition, you can specify up to five other block sizes. A database uses and allocates free database space in Oracle data blocks.

4. Extents

The next level of logical database space is an extent. An extent is a specific number of contiguous data blocks, obtained in a single allocation, used to store a specific type of information.

5. Segments

Above extents, the level of logical database storage is a segment. A segment is a set of extents allocated for a certain logical structure. The following table describes the different types of segments.

Segment	Description
Data segment	Each nonclustered table has a data segment. All table data is stored in the extents of the data segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
Index segment	Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.
Temporary segment	Temporary segments are created by Oracle when a SQL statement needs a temporary database area to complete execution. When the statement finishes execution, the extents in the temporary segment are returned to the system for future use.
Rollback segment	If you are operating in automatic undo management mode, then the database server manages undo space using tablespaces. Oracle recommends that you use automatic undo management. Earlier releases of Oracle used rollback segments to store undo information. The information in a rollback segment was used during database recovery for generating read-consistent database information

Segment	Description
	and for rolling back uncommitted transactions for users. Space management for these rollback segments was complex, and Oracle has deprecated that method. This book discusses the undo tablespace method of managing undo; this eliminates the complexities of managing rollback segment space, and lets you exert control over how long undo is retained before being overwritten. Oracle does use a SYSTEM rollback segment for performing system transactions. There is only one SYSTEM rollback segment and it is created automatically at CREATE DATABASE time and is always brought online at instance startup. You are not required to perform any operations to manage the SYSTEM rollback segment.

Oracle dynamically allocates space when the existing extents of a segment become full. In other words, when the extents of a segment are full, Oracle allocates another extent for that segment. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.

Database Writer (DBW*n*)

The server process records changes to undo and data blocks in the database buffer cache.DBWn writes the dirty buffers from the database buffer cache to data files. It ensures that a sufficient number of free buffers (buffers that can be overwritten when server processes needto read blocks from the data files) are available in the database buffer cache. Server processes make changes only in the database buffer cache. This is a prerequisite for high databaseperformance.

Checkpoint (CKPT)

Every three seconds (or more frequently), the CKPT process stores data in the control file to document which modified data blocks DBWn has written from the SGA to disk. This is called a “checkpoint”. The purpose of a checkpoint is to identify that place in the online redo log file where instance recovery is to begin (which is called the “checkpoint position”). In the event of a log switch, the CKPT process also writes this checkpoint information to the headers of data files.

Checkpoints exist for the following reasons:

• To ensure that modified data blocks in memory are written to the disk regularly so that data is not lost in case of a system or database failure

• To reduce the time required for instance recovery. Only the online redo log file entries following the last checkpoint need to be processed for recovery.

• To ensure that all committed data has been written to data files during shutdown

The checkpoint information written by the CKPT process includes checkpoint position, system change number, location in the online redo log file to begin recovery, information about logs, and so on.

Redo Log Files and LogWriter

Redo log files record changes to the database as a result of transactions and internal Oracle server actions. (A transaction is a logical unit of work, consisting of one or more SQL statements run by a user). Redo log files protect the database from the loss of integrity
because of system failures caused by power outages, disk failures, and so on. Redo log files must be multiplexed to ensure that the information stored in them is not lost in the event of a disk failure. The redo log consists of groups of redo log files. A group consists of a redo log file and its multiplexed copies. Each identical copy is said to be a member of that group, and each group is identified by a number. The LogWriter (LGWR) process writes redo records from the redo log buffer to all members of a redo log group until the file is filled or a log switch operation is
requested. Then it switches and writes to the files in the next group. Redo log groups are used in a circular fashion.

Archiver (ARC*n*)

ARCn is an optional background process. However, it is crucial to recovering a database after the loss of a disk. As online redo log files get filled, the Oracle instance begins writing to the next online redo log file. The process of switching from one online redo log file to another is called a log switch. The ARCn process initiates backing up or archiving of the filled log group at every log switch. It automatically archives the online redo log file before the log can be reused, so all of the changes made to the database are preserved. This enables recovery of
the database to the point of failure even if a disk drive is damaged.

One of the important decisions that a DBA has to make is whether to configure the database to operate in ARCHIVELOG mode or in NOARCHIVELOG mode.

• In NOARCHIVELOG mode, the online redo log files are overwritten each time a log switch occurs.

• In ARCHIVELOG mode, inactive groups of filled online redo log files must be archived before they can be used again.

System Monitor (SMON)

The system monitor (SMON) process performs recovery, if necessary, at instance startup.SMON is also responsible for cleaning up temporary segments that are no longer in use and for coalescing contiguous free extents within dictionary-managed tablespaces. If any terminated transactions are skipped during instance recovery because of file-read or offline errors, SMON recovers them when the tablespace or file is brought back online. SMON checks regularly to see whether it is needed. Other processes can call SMON if they detect a need for it.

Process Monitor (PMON)

The process monitor (PMON) performs process recovery when a user process fails. PMON is responsible for cleaning up the database buffer cache and freeing resources that the user process has been using.

PMON periodically checks the status of server processes and restarts any processes that have stopped running (but not those that are intentionally terminated by the Oracle instance).

PMON checks regularly to see whether it is needed and can be called if another process detects the need for it.

INSTANCE MEMORY STRUCTURES

Oracle creates and uses memory structures to complete several jobs. For example, memory stores program code being run and data shared among users. Two basic memory structures are associated with Oracle: the system global area and the program global area. The following subsections explain each in detail.

a.System Global Area

The **System Global Area (SGA)** is a shared memory region that contains data and control information for one Oracle instance. Oracle allocates the SGA when an instance starts and deallocates it when the instance shuts down. Each instance has its own SGA.

Users currently connected to an Oracle database share the data in the SGA. For optimal performance, the entire SGA should be as large as possible (while still fitting in real memory) to store as much data in memory as possible and to minimize disk I/O.

The information stored in the SGA is divided into several types of memory structures, including the **database buffer**s, **redo log buffer**, and the **shared pool**.

i.Database Buffer Cache of the SGA

Database buffers store the most recently used blocks of data. The set of database buffers in an instance is the database **buffer cache**. The buffer cache contains modified as well as unmodified blocks. Because the most recently (and often, the most frequently) used data is kept in memory, less disk I/O is necessary, and performance is improved.

ii.Redo Log Buffer of the SGA

The **redo log buffer** stores **redo entries**—a log of changes made to the database. The redo entries stored in the redo log buffers are written to an **online redo log**, which is used if database recovery is necessary. The size of the redo log is static.

iii.Shared Pool of the SGA

The shared pool contains shared memory constructs, such as shared SQL areas. A shared SQL area is required to process every unique SQL statement submitted to a database. A shared SQL area contains information such as the parse tree and execution plan for the corresponding statement. A single shared SQL area is used by multiple applications that issue the same statement, leaving more shared memory for other uses.

Statement Handles or Cursors

A **cursor** is a handle or name for a private SQL area in which a parsed statement and other information for processing the statement are kept. (Oracle Call Interface, OCI, refers to these as **statement handles**.) Although most Oracle users rely on automatic cursor handling of Oracle utilities, the programmatic interfaces offer application designers more control over cursors.

For example, in precompiler application development, a cursor is a named resource available to a program and can be used specifically to parse SQL statements embedded within the application. Application developers can code an application so it controls the phases of SQL statement execution and thus improves application performance.

b.Program Global Area

The **Program Global Area (PGA)** is a memory buffer that contains data and control information for a server process. A PGA is created by Oracle when a server process is started. The information in a PGA depends on the Oracle configuration.