Chapter 20. Database Administration and Tuning

Table of contents

- Objectives
- Introduction
- Functions of the DBA
  - Management of data activity
  - Management of database structure
  - Tables and tablespaces
  - Data fragmentation
  - Designing for the future
  - Information dissemination
  - Supporting application developers
  - Use of the data dictionary
    - The Oracle data dictionary
    - Core system tables
    - Data dictionary views
    - Application
  - Control of redundancy
  - Configuration control
  - Security
  - Summary of DBA functions
    - Documentation
    - Documentation for use throughout the organisation
    - Documentation for use within the DBA function
    - Qualities and roles of the DBA function
- Administration of client-server systems
  - Tools used in DBA administration
    - Data dictionary
    - Stored procedures
    - Data buffering
    - Asynchronous data writing
    - Data integrity enforcement on a server
    - Concurrency control features
    - Communications and connectivity
  - Client-server security
    - Server security
    - Client security
    - Network security
    - Checking of security violations
- Database tuning
  - Tuning SQL
    - Guidelines

  - Tuning disk I/O
  - Tuning memory
  - Tuning contention
  - Tools to assist performance tuning
    * Software monitors
    * Hardware monitors
  - Other performance tools
  - Concluding remarks on database tuning
- Discussion topics

Objectives

At the end of this chapter you should be able to:

- Describe the principle functions of a database administrator.
- Discuss how the role of the database administrator might be partitioned among a group of people in a larger organisation.
- Discuss the issues in tuning database systems, and describe examples of typical tools used in the process of database administration.

Introduction

In parallel with this chapter, you should read Chapter 9 of Thomas Connolly and Carolyn Begg, "Database Systems A Practical Approach to Design, Implementation, and Management", (5th edn.).

The functions of the database administrator (DBA) impact on every aspect of database usage within an organisation, and so relate closely to most of the chapters of this module. Database administrators often use the SQL language for the setting up of users, table indexes and other data structures required to support database applications. They will often work closely with analysts and programmers during the design and implementation phases of database applications, to ensure that the decisions made during these processes are appropriate in the context of the organisation-wide use of the database system. The DBA will work with management and key users in developing security policy, and will be responsible for the effective implementation of that policy in respect of the use of the databases in the organisation. In the common situation where client-server systems or a fully distributed database system are being used, the DBA is likely to be the one who determines important aspects relating to how the processing and/or data are distributed within the system, including, for example, the frequency with which replicas are updated in a DDBMS. This activity will be undertaken in close consultation with users and developers within the organisation, so that the strategy is informed by a good understanding of the priorities of the organisation as a whole. Furthermore, a good understanding of

the potential of new database technology, such as Object databases, is important to the DBA role, as it enables the DBA to advise budget holders within the organisation of the potential benefits and pitfalls of new database technology.

Modern database management systems (DBMSs) are complex software systems. When considered within the context of an organisational environment, they are usually key to the success of the organisation, enabling personnel at all levels of the organisation to perform essential tasks. In order for the database systems within an organisation to remain well aligned to requirements and provide a high degree of availability and efficiency, the system needs to be administrated effectively. The importance of the tasks of database administration was highlighted from the early days of database systems, in the writings of James Martin and others. With the increased complexity and frequent need in present systems to enable multiple databases to combine to provide an efficient overall service, the importance of database administration cannot be over-stressed. In this chapter we shall examine the tasks involved in the administration of database systems. We shall start by examining each of the major aspects of the job of a database administrator (DBA), and go on to explore the way in which the tasks involve the DBA in working with people from all levels of an organisation. In larger organisations, the functions of the DBA are sufficiently numerous to warrant being split among a number of different individuals who will form the members of a DBA group. We shall look at the ways that the job of the DBA might be partitioned, and explore the issue of what level within an organisation the DBA functions should be placed.

One of the most important and technically demanding aspects of database administration is the performance tuning for the database. This involves ensuring that the database is running efficiently, and providing a reliable and responsive service to the users of the system. In the later part of the chapter, we will discuss the major aspects of database tuning, using examples drawn from current systems.

Functions of the DBA

Management of data activity

The term 'data activity' refers to the way in which data is accessed and used within an organisation. Within medium to large organisations, formal policies are required to define which users should be authorised to access which data, and what levels of access they should be given; for example, query, update, copy, etc. Processes also need to be defined to enable users to request access to data which they would not normally be able to use. This will involve the specification of who has responsibility for authorising users to have data access. This might typically involve the line manager of the user requesting access, who should be in a position to verify that access is genuinely required, plus the authorisation of the user identified as the owner of the data, who should be in a position to

be aware of any implications of widening the access to the data. From time to time, conflicts may arise during the processing of requests for access to data. For example, a user may request access to particularly sensitive sales or personnel data, which the data owner refuses. In these circumstances, the DBA may be involved as one of those responsible for resolving the conflict in the best interests of the organisation as a whole.

Much of the DBA's activity will be focused on supporting the production environment of the organisation, where the operational programs and data are employed in the daily business of the organisation. However, in medium to large organisations, there will be significant activity to support developers in the processing of writing and/or testing new software.

Data activity policies formulated to address these issues need to be clearly documented and disseminated throughout the organisation. The policies might include agreements about the availability of data, specific times at which databases may be taken offline for maintenance, and the schedules for migrating new programs and data into the production environment.

A further aspect of data activity policy is the determination of procedures for the backup and recovery of data. The DBA will require a detailed knowledge of the options provided by the DBMSs being used regarding the backup of data, the logging of transactions, and the recovery procedures in light of a range of different types of failure.

Management of database structure

Another fundamental area of activity for the DBA concerns the definition of the structure of the databases within the organisation. The extent to which this is required will vary greatly depending on how mature the organisation is in terms of existing database technology. If the organisation is in the process of acquiring and setting up a database or databases for the first time, many fundamental decisions need to be taken regarding how best this should be done. Most organisations have progressed well beyond these fundamental issues today; however, even in such organisations, the effectiveness of those initial decisions needs to be monitored to ensure that the computer-based information within the organisation provides a good fit for the needs of its users. Among the key decisions to be taken during the process of establishing database systems within an organisation are the following:

- Which type of database systems best suit the organisation? There is no doubt that Relational database systems dominate the current marketplace, but for some organisations, particularly those specialising in engineering or design, there is certainly a case for examining the potential of Object-based systems, either instead of or running alongside a Relational system.
- How many databases are required? In the early days of database systems,

most organisations used one single database system, and the data on that system could be considered to provide the data model for the enterprise as a whole. Today, it is much more often the case that a number of database systems are employed, sometimes from different vendors, and sometimes with different underlying data models. This usually implies the need to transfer at least some of the data between these different systems.

- Is it necessary to establish a number of different database environments to support different types of activity within the organisation? This might, for example, involve the creation of development and production environments mentioned above, or may require a different organisation of work, such as, for example in a chemical company, the setting up of one environment to support online transaction processing, and another environment to enable the research and development of models of chemical and manufacturing processes.
- The interfaces between different database systems must be defined to allow transfer of data between them. Furthermore, interfaces between each database system and any other important software components, such as groupware and/or office information systems, need to be established.

The above decisions can be seen as arising at the strategic level of database administration, and will be considered in conjunction with senior members of the company and/or department in which the DBA is based. Nearer to the operational level, there is a range of important decisions concerning the structure of individual database systems within the organisation. Some of these are described below.

Tables and tablespaces

A 'tablespace' is a unit of disk space which contains a number of database tables. Usually each tablespace is allocated to data of a particular type; for example, there may be a tablespace established to contain user data, and another one to contain temporary data, i.e. data that is only in existence for a short time and is usually required in order to enable specific processes such as data sorting to take place. A further example might be in a university database, where separate tablespaces might be established respectively to support student, teaching staff and administrative users.

As an aside, the pattern of usage of tables in the student tablespace of a university database is very different to that typically seen in a commercial system. Very generally speaking, in a commercial system, data activity in a production environment might be characterised as comprising the following:

- Access to a relatively small number of fairly large tables.
- Regular executions of a predefined set of transactions.

- Several transactions may regularly scan large volumes of data.

In contrast, data activity in a student tablespace during a database class might be characterised as comprising:

- Many different tables, each of which contains only a few rows.
- Irregular and different types of transactions.
- Very small numbers of records processed by each transaction.

Tablespaces are an extremely important unit of access in database administration. In many systems they are a unit of recovery, and, for example, it may be necessary to take the whole tablespace offline in order to carry out recovery or data reorganisation operations to data in that tablespace.

A further important consideration is the way in which tables, indexes and tablespaces are allocated to physical storage media such as disks. This will be discussed further later in this chapter, in the section on database tuning. It is important here to point out that the DBA requires a good understanding of the volume of query and update transactions to be made on the tables in a tablespace, so that the allocation of physical storage space can be made in such a way that no physical device such as a disk becomes a major bottleneck in limiting the throughput of transactions.

Typically, database systems provide a number of parameters that can be used to specify the size of tablespaces and tables. These often include the following:

- Initial extents: The amount of disk space initially made available to the structure.
- Next extents: The value by which the amount of storage space available to the structure will be increased when it runs out of space.
- Max extents: The maximum amount of space to be made available to the structure, beyond which further growth can only be enabled by further intervention by the DBA.

Note: An extent is a term referring to a contiguous area of disk space. For any specific DBMS, it will consist of a number of data blocks which will be stored together in the same extent.

Data fragmentation

Over a period of time during which a database table is being used, it is likely to experience a significant number of inserts, updates and deletions of data. Because it is not always possible to fit new data into the gaps left by previously removed data, the net effect of these changes is that the storage area used to contain the table is left rather fragmented, containing a number of small areas which are hard to insert new data into them. This phenomenon is called ‘data

fragmentation'. The effect of data fragmentation in the long run is to slow down the performance of transactions accessing data in the fragmented table, as it takes longer to locate data in the fragmented storage space. Some database systems provide utilities (small programs) that can be used to remove these pockets of unusable space, consolidating the table once more into one contiguous storage structure — this is known as defragmentation. In other DBMSs, where no defragmentation utility is available, it may be necessary to export the table to an operating system file, and re-import it into the database, so that the pockets of unusable free space are removed.

Review question 1

- Describe the difference between production and development environments.
- Make a list of the range of different types of failure that the DBA needs to plan for in determining adequate backup and recovery strategies.
- Describe typical parameters that can be used to control the growth of tables and tablespaces.
- What is gained when a table is defragmented?

Designing for the future

An important overall principle to be applied when trying to estimate the requirements for storage and performance tuning is to design for the future. It will be part of the DBA's role to collect information from those responsible for the introduction of new database applications, about the volumes of data and processing involved. Producing such estimates can be extremely difficult, but even when these are correct, it is a mistake to plan on the basis of these figures alone. It is more realistic to base estimates of the volume of data and processing required on what they might be expected to have reached in a year's time, and to provide sufficient storage and processing capacity on that basis. This avoids being caught by surprise, should the volume of activity within the application grow much faster than was initially anticipated.

Information dissemination

Communication is an essential aspect of the role of the DBA. When new releases of DBMSs are introduced within the organisation, or new applications or upgrades come into use, it is important that the DBA is sensitive to the needs of the user population as a whole, and produces usable documentation to describe the changes that are taking place, including the possible side effects on users and developers. An equally important role in communicating information relates to the development and dissemination of information about programming and testing standards that may be required within an organisation. This may include

specifications of who is able to access which data structures, and standards for the use of query and update transactions throughout the organisation.

Supporting application developers

The DBA plays an important role in assisting those involved with the development and/or acquisition of software for the organisation. As well as being the gatekeeper of database resources, and monitoring future requirements for database processing (as exemplified in activity 1 further on), the DBA can provide ongoing advice to analysts and programmers about the best use of database resources. Typical issues that this may involve are the following:

- Information on the different types and instances of databases in the organisation, including the interfaces between them, any regular times when databases are unavailable, etc.
- Advice on the details of existing tables and tablespaces in the databases of the organisation.
- Advice on the use of indexes, different index types available and indexes currently set up.
- Details of security standards and procedures.
- Details of existing standards and procedures for the use of programming languages (including query languages).
- Details of migration procedures used for moving programs and data in and out of production.

Use of the data dictionary

The data dictionary is a key resource for database administration. It contains data about the tablespaces, tables, indexes, users and their privileges, database constraints and triggers, etc. Database administrators should develop a good knowledge of the most commonly used tables in the dictionary, and reference it regularly to retrieve information about the current state of the system. The way in which dictionaries are organised varies greatly between different DBMSs, and may change significantly with different releases of the same DBMS.

The Oracle data dictionary

The Oracle data dictionary is a set of tables that the Oracle DBMS uses to record information about the structure of the database. There is a set of core system tables, which are owned by the Oracle user present on all Oracle databases, the SYS user. SYS is rarely used, even by DBAs, for maintenance or enquiry work. Instead, another Oracle user with high-level system privileges is used.

The DBA does not usually use the SYSTEM user, which is also automatically defined when the database is created. This is because product-specific tables are installed in SYSTEM, and accidental modification or deletion of these tables can interfere with the proper functioning of some of the Oracle products.

Core system tables

The core data dictionary tables have short names, such as tab$, col$, ind$. These core system tables are rarely referenced directly, for the information is available in more easily usable forms in the data dictionary views defined when the database is created. To obtain a complete list of the data dictionary views, query the DICT view, as shown in the section on data dictionary views below.

Data dictionary views

The data dictionary views are based on the X$ and V$ tables. They make information available in a readable format. The names of these views are available by selecting from the DICT data dictionary view. Selecting all the rows from this view shows a complete list of the other accessible views.

SQL*DBA is an Oracle product from which many of the database administration tasks can be performed. However, SQL*Plus provides basic column formatting, whereas the SQL*DBA utility does not. Therefore, you use SQL*Plus for running queries on these views. If you are not sure which data dictionary view contains the information that you want, write a query on the DICT_COLUMNS view.

Most views used for day-to-day access begin with USER, ALL or DBA. The USER views show information on objects owned by the Oracle user running the query. The data dictionary views beginning with ALL show information on objects owned by the current Oracle user as well as objects to which the user has been given access. If you connect to a more privileged Oracle account such as SYS or SYSTEM, you can access the DBA data dictionary views. The DBA views are typically used only by the DBA. They show information for all the users of the database. The SELECT ANY TABLE system privilege enables other users to access these views. Querying the DBA_TABLES view shows the tables owned by all the Oracle user accounts on the database.

Application

Activity 1 - Capturing requirements for application migration

Imagine you are a database administrator working within a large organisation multinational. The Information Systems department is responsible for the in-house development and purchase of software used throughout the organisation. As part of your role, you require information about the planned introduction of software that will run in the production database environment. You are required

to develop a form which must be completed by systems analysts six weeks before their application is moved into production. Using a convenient word processing package, develop a form to capture the data that you think may be required in order for the DBA team to plan for any extra capacity that might be required for the new application.

Activity 2 - Examining dictionary views available from your user account

Query the DICT_COLUMNS table to examine the list of tables including the string USER visible from your own Oracle account. Remember to put USER in upper case in the query. The output should contain quite a lot of views. To avoid the information scrolling off the top of the screen, issue the command

Set pause on

prior to issuing the query. The information should then appear 24 lines at a time. You can view the next 24 lines by pressing the Enter key on your PC.

Activity 3 - Examining the details of available tables

Log into your Oracle account. Issue the following command to view the details of tables that you own:

SELECT * FROM USER_TABLES;

Examine the columns of the results that are returned.

To explore any further tables that are available to you, issue the following query:

SELECT * FROM ALL_TABLES;

To examine any of these tables further, use the DESCRIBE command, or issue further SELECT commands on the specific tables you find.

Control of redundancy

We have seen from the chapters on database design that a good overall database design principle is to store one fact in one place. This reduces the chances of data inconsistency, and avoids wasted space. There are, however, situations in which it can be appropriate to store redundant information. One major example of this we have seen in connection with the storage of replicas or copies of data at different sites in a truly distributed database system. Even within a non-distributed system, data may sometimes be duplicated, or derived data stored, in order to improve overall performance. An example of storing derived data to improve performance, might be where a value which normally could be calculated from base values in an SQL query is actually stored explicitly, avoiding the need for the calculation. For example, we might store net salary values, rather than relying on transactions to calculate the value by subtracting various deductions (for tax, national insurance, etc) from gross pay. This reduces

the processing load on the system, at the expense of the extra storage space required to hold the derived values. The issue of calculating derived values every time from the underlying base values is generally more of a significant problem than the provision of the extra storage space required. The DBA will play an important role in advising developers about the use of redundant and derived data, and will maintain control of these issues, as part of the overall responsibility for ensuring the database runs efficiently.

Configuration control

In a modern database environment, there are usually several products which are used to provide the range of functions required. These will typically include the database server, and various additional products such as a GUI forms-based interface, report writing tool, data graphing tool, software to assist the design of new applications, utilities for migrating programs and data in and out of production, etc. All of these different software components will be upgraded from time to time, and it is likely that some of them will come from different vendors. Many of the problems that occur in developing Information Systems arise when trying to enable two or more software components to communicate effectively. For this reason, it is essential that the DBA maintains a record of which versions of which software components have been and are currently running. In general, this area of work is known as configuration control. The information needed to be maintained to carry out successful configuration control is, as a minimum, the following:

- For currently running software components, a note of the name, exact release number, date put into production, vendor, and the details of any parameter values or activities that have been necessary in order to establish interfaces with other products, including the release numbers of those products and the dates during which these changes were effective.
- For software components that have been used previously: the name, exact release number, dates introduced and withdrawn from production, vendor, and as above, any details required of parameter values or other activities required to enable this component to communicate effectively with other products, including the release numbers of those products and the dates during which these values, etc, were in effect.

Even when all of the products have been purchased from the same vendor, proper configuration control is extremely important to ensure the correct running of the environment. Having this information available gives the option that, should a problem arise involving incompatibilities of software components, it may be possible to revert back to a previously stable configuration, maintaining a service to users while the incompatibility problem is resolved.

Security

Security is a major issue in database systems, and the DBA is the foremost person with responsibility for ensuring the day-to-day security of the stored data. This process begins with the DBA working in conjunction with managers, key users and owners of the organisations data, to establish appropriate security mechanisms and procedures. This will be a process of determining how to make the most appropriate use within the organisation of the security mechanisms provided by the DBMS and other software in use, plus a clear definition and documentation of supporting policies and processes to ensure that data and programs are properly protected. Typical issues that should be addressed include:

- Procedures for the allocation of passwords. Many database systems provide considerable flexibility in the use of passwords, enabling them to be set from the database level right down to the individual attribute level. Procedures need to be defined regarding how passwords are allocated, including any rules regarding the format of passwords; e.g. whether they should exceed a certain length, and how long they can be used for before they expire and need to be reset. The requirements for passwords may vary hugely, from a development environment in which a number of standard user accounts have been established requiring no password protection, right through to highly secure production situations such as records of bank account details, in which two passwords may be required to access a particularly sensitive attribute. In general, a very important consideration to keep in mind with all security mechanisms, is the set of procedures and practices that are used within the organisation for the use and communication of security information. For example, there is no point in having a very sophisticated software protection system to prevent people from discovering one another's passwords, if it is commonplace for people to write their passwords on their PCs or in other places in their work area, where they can be easily read by anybody.
- Procedures for the administration of database privileges, such as the granting and revocation of access to tables, query and update transactions.
- The use of encryption techniques for encoding data while it is being transmitted over networks, including any intranet and the Internet.
- As discussed earlier, the establishment of procedures for transaction logging and recovery from a range of different failures.

Summary of DBA functions

Documentation

We have seen that the job of the database administrator impacts on many aspects of an organisation's work. Depending on the geographical spread of an

organisation, and its size in terms of personnel, the person or people undertaking the DBA role may be required to produce a significant amount of documentation, in order to describe various aspects of database activity to users as a whole. This documentation is likely to include the following elements:

Documentation for use throughout the organisation

- Security standards and procedures.
- Details of forthcoming changes to DBMSs being used or of DBMSs to be introduced.
- Database change procedures for developers.
- Meeting documentation to clarify agreed developments and changes with users.
- Programming and query language standards.

Documentation for use within the DBA function

- Documentation for configuration control.
- Records of changes made to the database structure and system.
- Records of test procedures and test runs after changes.

Qualities and roles of the DBA function

The DBA is clearly a key player in the success or failure of a company. The role encompasses a wide range of technical and political/social skills. In medium to large organisations, it is extremely likely that the job of database administration will be split into a number of different parts, and be performed by a group of between 3-5 people. Each of these individuals will take responsibility for specific aspects of database administration. Among the qualities that would be required collectively of this group of people, we would expect to see the following:

- Good communications. The DBA needs to communicate effectively with people throughout the entire spectrum of the organisation. Communications with high-level management is required, in order that the DBA function can be sufficiently informed about strategic directions, so that the database strategy for the organisation can be closely aligned with the business strategy. Frequent communications will be needed with many other levels of the organisation, including Information Systems personnel, end-users and their managers.
- Technical knowledge. In addition to the need to have a sound knowledge of the various utilities and database languages being used to administer user privileges and monitor database activity, a detailed understanding of

the interfaces to other database systems is often required. The knowledge used to tune the performance of a database system ranges from the ability to spot individual trouble spots, such as an inefficiently coded SQL transaction, which can be sped up by recording or the use of indexes, through to variations of system parameters that might be used to make global performance improvements to the DBMS.

- Good understanding of the organisation and its priorities; ability to liaise with management.
- Good arbitrator, in situations where it is necessary to make decisions regarding disputes over access to data or processes.
- Trustworthy - clearly a major part of the security of the organisation is in the hands of the DBA.
- Respected by application development staff and management.
- Cool under pressure. Should disasters arise, for example at the database-application or whole-DBMS level, it is likely that the DBA will be involved in trying to resolve the situation, with minimum disruption to the users and clients of the organisation.
- Flexible. It is possible that years of hard-won knowledge relating to a specific DBMS may from time to time become obsolete, either because that system is replaced by a substantial new release, or because for business reasons, the organisation decides to migrate to a totally different DBMS.

In situations where the functions of database administration are to be organised among the members of a DBA group or team, the following roles might be identified. Depending on the level of work related to each role, one person may adopt more than one role within the overall context of the DBA group.

- Database project leader.
- Documentation and standards developer/disseminator.
- User representative.
- Database systems manager.
- Performance tuning expert.
- Research and development specialist, perhaps looking into database technologies which are new, or new to the organisation, such as replication, Object databases, data mining, etc.

Review question 2

- Describe an example of a situation other than that given in the content of the chapter, in which it may be desirable to store redundant or derived data.
- Describe the arguments for and against storing derived data.

- Explain what is meant by the term configuration control.

Administration of client-server systems

The job of the DBA is to decide what database system and architecture is suitable for the company. He/she needs to be well in tune with the business strategies and objectives, and how the database architecture impacts the development and priorities of the organisation's information systems. He/she is the person to establish policies for maintaining and dealing with database systems in the organisation. He/she is also responsible for ensuring that the system operates with adequate performance to meet the demands of the organisation. Faced with such great responsibility, the DBA needs to know the various issues of client-server architecture and what impact it has on the organisation.

The advantage of client-server architecture is its potential in portability, scalability and interconnectability of clients and servers using various network configurations. In addition, when evaluating Relational DBMS on client-server systems, the DBA must consider many factors besides the architectural model - transaction control, performance, security, integrity, procedure logic and other issues also figure prominently.

SQL has become a standard data access language between client and server machines. We expect to find a mechanism for the server to accept SQL statements and return data and status codes to the client. However, many vendors have added their own extra extensions to standard SQL (i.e. proprietary data-access languages). These extensions are advanced and change as strategy and technology develops.

However, though the extra SQL features can be attractive, the portability of the application in the future may be affected.

It should be noted that no theoretical rule says that only one server may access the database, and there are many reasons for wanting more than one in operation. Once a server has been activated, it is called an instance. A database without an attached instance is a lifeless collection of data and status information; likewise, an instance that is not attached to a database is useless.

Tools used in DBA administration

Tools that are of value to the DBA in supporting client-server systems are as follows:

Data dictionary

The data dictionary is, as we have seen, itself a set of tables and views. When responding to a client request, the server can find any data required about the

data it needs in the data dictionary. It can use the same mechanisms it employs on behalf of clients to read its own data. These are commonly known as recursive requests because the server generates them automatically.

Stored procedures

(These are used by the DBA and programmers, briefly encountered in the chapters on distributed database systems.) Stored procedures are groups of SQL statements which are stored in the server. They can be run like procedures written in standard programming languages, and allow portions of application code (normally commonly executed tasks or transactions) to move from client to server. The server checks the syntax of these stored procedures, The existence, accessibility and data type of each object mentioned in each statement must then be verified. The database engine's SQL optimiser is usually invoked at this point to choose the best access path to the referenced data. Advantages of stored procedures are that they reduce network traffic (fewer SQL statements are sent from client to server), and they improve server performance as they are compiled prior to execution on the server.

Data buffering

During execution of statements that query or update the database, certain data (sent from or requested by the client) is placed in memory. The server tries to keep the data in memory to save disk input/output (I/O) should the next request require data that's still in the buffer. The buffer size should be configured so the DBA can optimise the memory-versus-speed trade-off.

Most servers provide shared buffering, in which data brought into the buffer for one client will be later used by all others. When data is regularly shared by many users, server buffering is a necessity.

Asynchronous data writing

This feature is used to try to smooth out peaks in I/O that may arise during database processing. If I/O slows down, the server starts writing data blocks from the buffers to disk. Since this write activity is scheduled during periods of otherwise low I/O, there is no degradation in performance, even if the data in the buffer is changed later. If an urgent need for buffer space develops, the disk write is already done; the new request can be serviced without a time-consuming disk wait.

Data integrity enforcement on a server

With the client-server architecture, all database processing can be consolidated on a server machine. Such consolidation provides an opportunity to achieve

a high degree of data integrity. Since every database request is processed by a server, if database constraints are defined at the server level, they can be consistently applied. Server-based enforcement of data integrity guarantees data correctness and integrity by having the server enforce constraints on any changes or updates to the database. As such constraints are held centrally, they cannot be bypassed as the data can only be accessed through the server software.

Concurrency control features

One of the challenges in the development of client-server applications is to gain the maximum degree of parallelism on the client computers while providing protection against problems such as lost updates and inconsistent reads. Concurrency control allows multiple clients to access the server and still preserve the integrity of shared data. Updates by users are controlled and isolated to prevent one's changes disrupting or overwriting another's. This is usually enforced by using various automatic locking mechanisms, multiversioning techniques or rollback of partially completed transactions.

Some server implementations provide consistency by blocking writers from accessing the data being read. Others offer snapshots showing the state of data when the read began. Changes are ignored until the next read statement begins.

Communications and connectivity

A characteristic of client-server architecture is that a client application and server software are on different computers. The protocol used to pass messages, SQL and data between them is therefore of crucial importance. As there is no single standard protocol for computer networks, the server has to offer tools to handle the complexities of multivendor networks (i.e. to enable any application to be able to run on any network supported by the server without the application developer needing to be concerned with handling the hardware and network-specific communications issues). However, the connection between PCs and minicomputers is much more complex to implement than a conventional terminal-to-host implementation.

Client-server security

Server security

The built-in security mechanism of the database server provides central data access control, thereby reducing the need for security measures in the client applications. The server normally provides three basic levels of security:

1. User enrolment. This involves granting a user access to the database server itself.

2. Access privileges. This grants users access and privileges to individual database objects.
3. Resource allocation. This controls the amount of disk space allocated on the server to each user's database objects.

Client security

As the general administration of security in a client-server system is handled by the DBA at the server end, the client needs only to be concerned with errors returned from the server when an unauthorised operation is detected.

Network security

The distribution of a system, be it as a client-server or truly distributed database system, requires the additional issues associated with the protection of the data as it is transmitted across the network to be handled. This is most often dealt with by encryption algorithms, which encode the data, rendering it useless if it is intercepted during transmission. Following reception of encrypted data, the receiver of the data will run the decryption algorithm to re-establish the original data values.

Checking of security violations

Journal logging and other facilities are used to locate security breaches or violations in the server. The reason for a user failing action should also be logged so that the DBA can distinguish between a simple human error and attempted security violations.

Review question 3

Within the context of client-server systems, explain the meaning of the following terms:

1. Shared data buffering
2. Asynchronous data writing

Database tuning

Providing an efficient service to users of the database system is an ongoing responsibility of the DBA. In the following sections we shall examine some of the major considerations involved in the tuning of a database system. Most of the examples we shall give are somewhat specific to the Oracle DBMS, but certainly have analogies in any of the other major database systems of today.

Tuning SQL

The optimisation of SQL transactions is a major topic in its own right. There are a number of guidelines which can be followed, which in general will lead to more efficient SQL. In general, the DBA activity in tuning specific transactions should be focused on those which:

- are run sufficiently often to have a noticeable impact on performance;
- access sufficient numbers of records (including intermediate results obtained during the evaluation of the query) to provide scope for transaction tuning.

Guidelines

The following guidelines can be applied when tuning SQL transactions:

- Use indexes on primary and foreign keys, and consider their use on other attributes that are frequently referenced in the WHERE clause of queries.
- Corollary to the above, indexes improve performance for queries that return fewer than approximately 20% of the rows in a table; otherwise it is faster to use a full table scan. For update transactions, indexes can actually make things slower, because of the need to update the index structure.
- Unique indexes are faster than non-unique indexes.
- Several SQL constructs, such as use of the keywords like '%string', distinct, group by, order by, max, etc, lead a query not to use an index. The reason for this is that either the data to use the index is not available (as with '%string'), or the operation implies a sort of the data, in which case all of the rows need to be accessed.
- Compressed indexes save space, but do not provide as substantial an improvement in response time.
- In general, joins are executed more efficiently than nested queries. This may provide scope for recasting an existing nested query in the form of a join.
- Use of short table aliases in queries can improve performance.

Activity 4 - Examining the time response of queries

To examine the times associated with the execution of an SQL command in Oracle, we can use the sql*plus command

Set timing on

Log in to the Oracle system and issue the above command. Experiment then by running several queries to examine the time values returned. For example, experiment for queries that return a single row, that perform a full table scan,

and for JOIN queries. The value returned for each query appears after the rows displayed in the result of the query.

Tuning disk I/O

No matter how much money you spend on high-speed disk technology, there is still no getting away from the fact that disks are slow when compared with solid-state devices. For this reason, most data structures and design options are geared around minimising disk input and output. As long as we are limited to disks as the main medium for storing our data, then researchers will continue to search for methods to improve the efficiency of storing and retrieving data from these devices. The following guidelines are useful when trying to minimise the impact of relatively slow disk I/O processes:

- Reduce disk contention. Contention occurs when several users try to access the same disk, at the same time. If contention is noticeable on a particular disk and queues are visible, then distribute the I/O by moving heavily accessed files onto a less active disk. Distribute tables and indexes on different disks if resources are available.
- Allocate free space in data blocks (i.e. space in a block is used by INSERTS and also UPDATES which increase the size of a row).
- Allow for block chaining by using PCTFREE (Oracle specific, i.e. the percentage of blocks reserved for row expansion) parameter to control/limit chaining. Chaining can be examined using the ANALYZE command. (Chaining occurs when data in a block grows beyond the size that can be contained within the block, and so some of the data must be stored in a further block, to which the original block must have a pointer. We describe this dynamic expansion of data into additional blocks as chaining. Its overall effect is to reduce performance, as the system must follow the series of pointers to retrieve requested data.)
- Seek to minimise dynamic expansion. For example, with Oracle, set up storage parameters in the CREATE table and CREATE tablespace statements so that Oracle will allocate enough space for the maximum size of the object. Hence space will not need to be extended later.
- Tune the database writer DBWR (an Oracle-specific process which writes out data from the buffer cache to the database files).

Note: If the hardware is changed in the future, then it is quite likely that the system will need re-tuning, because many of the settings will be hardware specific.

Tuning memory

As previously mentioned, accessing disk is very expensive in terms of performance, whereas access to memory is much faster. Hence, we want to make the majority of accesses to be to memory rather than to disk. In an ideal world(!), it would be nice if we could load the whole of the database into memory so that all accesses are then to memory, rather than disk. Obviously this is not possible in the real world, so instead the system needs to be tuned to make the best use of the limited amount of memory available.

Oracle's memory can be broken down as follows:

- **System Global Area (SGA):** This block of memory is used for storing data for use by Oracle processes. It is a shared portion of memory for all Oracle processes.
- **Caches:** Blocks of memory used for keeping copies of data that is also on disk, but which can be accessed much quicker here. Hence it makes sense to keep frequently accessed data in cache. The two main caches of concern are:
- **Data dictionary cache:** Requires only a small amount of memory in comparison to the buffer cache, but can have a dramatic effect on performance. The actual size of this will depend on the different types of objects used by applications.
- **Buffer cache:** It is here where tables and indexes can be stored. The most frequently accessed tables and indexes should be stored here to minimise disk I/O as much as possible.
- **Program global area (PGA):** A PGA is a non-shared memory region that contains data and control information exclusively for use by an Oracle process. The PGA is created by Oracle Database when an Oracle process is started.
- **User Global Area (UGA):** The UGA is memory associated with a user session.
- **Software code areas:** Software code areas are portions of memory used to store code that is being run or can be run. Oracle Database code is stored in a software area that is typically at a different location from user programs — a more exclusive or protected location.

Note

Memory is also required for operating system use, hence other factors need to be taken into account, such as memory allocation for paging and virtual memory. For example, if the system is multi-user, then an increase in the number of users currently online could alter the performance on the machine quite dramatically.

Tuning contention

The term 'contention' refers to a problem which can arise in most areas of computing. It occurs when several processes make an attempt to gain access to the same resource at the same time. This will obviously result in a performance degradation, as one or more processes will need to wait until the resource is available. There are three main areas concerning memory contention in Oracle:

- Data blocks. Usually occurs when users attempt to update the same block.
- Rollback segments. All transactions use the rollback segments, so if there is only a small number of segments, contention is quite likely. A guideline given by Oracle is to use one rollback segment per five active users.
- Redo log buffers. Any block modification will write data to this buffer. The 'redo space waits' statistic can be used to provide information on contention for this buffer.

Tools to assist performance tuning

Having looked at the various factors that can affect performance in a system, what tools are available to aid the tuning process? Depending on the type of database, there will be a selection of tools available to monitor the system, allowing the DBA (or similar) to see the effects of tweaking the system.

Monitors can be broken down into two types:

- Software monitors
- Hardware monitors

Software monitors

These are programs which can be called up when necessary to provide statistics on the state of the system. These tools are flexible and may even be specially written by the DBA, although most vendors now supply these. Unfortunately, as these tools actually run on the system, they themselves apply an additional performance overhead, requiring CPU time, etc, in order to execute.

Hardware monitors

Hardware monitors consist of electronic devices which record data collected by probes, where each probe is connected to circuitry in the machine and/or peripherals. A major advantage of these, is that they do not interfere with the operation of the system.

As well as monitors, tools are available which aid database set up, loading, checking, back-up, and recovery and general database maintenance.

Other performance tools

An overview of some of the performance tools which are commercially available are as follows:

- Explain facility. This excellent facility allows the user to obtain information regarding the optimiser's choice of access strategy for a particular SQL statement. It is available for both Oracle and DB2. It works by examining the access paths that would have been chosen for the execution of a particular query. Note that it doesn't actually execute the query, which is a major bonus if trying to analyse the processing of a lengthy transaction.
- SQL*DBA monitor facilities. This Oracle utility allows the DBA to monitor database activity, which is grouped into areas such as Locking, File I/O, Statistics and Tables. This utility allows most bottlenecks which are causing performance degradation to be discovered.
- RUNSTATS utility. A DB2 utility which calculates statistics based on data stored. It is usually run after data is loaded or after a significant amount of updating has taken place.

Concluding remarks on database tuning

The subject of database performance tuning continues to receive a great deal of attention in both academic and industrial establishments. As databases become more complex, they place an even greater load on available resources, hence improved techniques for getting the best performance out of the system will always be in demand.

Discussion topics

1. Evaluating database systems

   Database administrators require from time to time to evaluate database systems in order to assess whether they will meet the needs of their organisation. This process will involve discussing requirements and features with the vendors of the database systems being considered, viewing of demonstrations, and finding out the opinions of other users of the systems. Some of the requirements for the system may be very specific to the organisation, but there are many characteristics which are generic to the evaluation of database systems. In discussion, identify the factors that you consider to be important in evaluating a DBMS. You may wish to start by making a personalised list of factors, and proceed by exchanging and discussing the lists prepared by other people on the course. In particular, given the fact that no DBMS is likely to fulfil all the requirements stated, it may

be interesting to seek agreement on a prioritised list of your top 10 or so requirements. There may be a few cases where it is not possible to place a higher priority on one factor than another, but these situation should be kept to a minimum, to sharpen the selectivity of your prioritised list.

2. Distribution of the DBA function

   In an organisation using a truly distributed database system, it would be possible to allocate aspects of the database administration function to individuals based at sites where significant amounts of data are to be stored. Consider in discussion what may be the possible benefits of distributing the database administration function in this way, and whether anything might be lost in distributing the function, when compared with a more centralised approach.