UNIVERSIDAD AUTONOMA DE QUERETARO

FACULTAD DE INFORMATICA

ENTORNO Y COMUNICACIÓN DE BASE DE DATOS ORACLE 8i(NET8)

TESINA

Que para obtener el Título de:

LICENCIADO EN INFORMATICA

Presenta:

Migdalia Marcela García Alamilla

ASESOR:

I.S.C. JABEL RESENDIZ GONZALEZ

Santiago, de Querétaro, Qro.,Octubre de 2003

AGRADECIMIENTOS

A mis padres Francisco García Fernández y Leidi Alamilla Alejandro; gracias por brindarme todo el amor, apoyo y confianza, ya que todo lo que he logrado en la vida ha sido gracias a ustedes.

A mis hermanos Gabriela, Francisco y Mariana, por el apoyo y comprensión que siempre me hicieron sentir en el transcurso de mi carrera.

A mis tíos Carlos Barraza y Elsa García, por su ayuda y comprensión durante los primeros años.

Con cariño a Norberto Vega, por su disposición y apoyo en los últimos meses de mi carrera.

Y a todas esas personas que de una u otra forma han contribuido al logro de uno de mis más grandes anhelos.

En especial a DIOS, por haber iluminado mi camino y haberme llevado hasta donde estoy ahora.

INDICE

1. Conceptos Básicos 3
1.1 ¿Qué es un sistema de base de datos? 3
1.1.1 Datos 4
1.1.2 Hardware 6
1.1.3 Software 6
1.1.4 Usuarios 6
1.2 Datos de la operación 7
1.3 ¿Por qué utilizar bases de datos? 11
1.3.1 Puede reducirse la redundancia 12
1.3.2 Puede evitarse la inconsistencia 12
1.3.3 Los datos pueden compartirse 13
1.3.4 Pueden hacerse cumplir las normas establecidas. 13
1.3.5 Pueden aplicarse restricciones de seguridad. 13
1.3.6 Puede conservarse la integridad 14
1.4 Independencia de los datos 15
1.4.1 Campo almacenado 17
1.4.2 Registro almacenado 17
1.4.3 Archivo almacenado 18

2. Arquitectura de un sistema de base de datos 19
2.1 Introducción 19
2.2 Los tres niveles de la arquitectura 19
2.3 El nivel externo 25
2.4 El nivel conceptual 28
2.5 El nivel interno 29
2.6 Transformaciones 30
2.7 El administrador de base de datos 31
2.8 El sistema de administración de base de datos 35
2.9 El administrador de comunicaciones de datos 41
2.10 Arquitectura cliente/servidor 41
2.11 Utilerías 45
2.12 El procesamiento distribuido 46

3. Administración de una base de datos local 49

3.1 Conexión de una red local 49
3.1.1 Terminología 50
3.2 Asignación de direcciones y enrutamiento 50
3.3 Eligiendo una estructura de direcciones 53
3.3.1 Subredes y múltiples números de red 55
3.3.2 Como asignar las subredes o los números de red 56
3.3.3 Trabajar con múltiples subredes "virtuales" en una red 58
3.3.4. Múltiples subredes: consecuencias en el broadcasting 59
3.3.5 Eligiendo una clase de dirección 59
3.3.6 Líneas IP y microgateways 61
3.3.6.1 Líneas IP 61
3.3.6.2 Microgateways 64
3.4 Servicios a nivel de red, nombres 66
3.5 Puentes y Gateways 68
3.5.1 Diseños alternativos 69
3.5.1.1 Una red de líneas punto a punto 69
3.5.1.2 Tecnología de los circuitos de conmutación 71
3.5.1.3 Redes de un solo nivel 71
3.5.1.4 Diseños mixtos 72
3.5.2 Introducción a las tecnologías de conmutación 73
3.5.2.1 Repetidores 73

4. Administración y configuración de net8 75
4.1 Configuración de net8 75
4.1.1 Configuración desde el asistente 78
4.2 Asignación de direcciones y enrutamiento 82
4.2.1 Subredes y múltiples números de red 85

Conclusiones 86

Bibliografía 87

1. CONCEPTOS BÁSICOS

1.1. ¿QUÉ ES UN SISTEMA DE BASE DE DATOS?

La tecnología de las bases de datos se ha descrito como "una de las áreas de la ciencia de la computación y la información de más rápido desarrollo". Como campo comercial, aún es relativamente nueva; los fabricantes y vendedores no empezaron a ofrecer sistemas de administración de bases de datos hasta mediados de la década de 1960 (aunque es verdad que ciertos paquetes de software antiguos incluían algunas de las funciones que ahora se asocian con tales sistemas. Pese a su calidad de innovación sin embargo, el campo rápidamente ha cobrado importancia práctica y teórica. La cantidad total de datos encomendados a las bases de datos se mide, sin exagerar, en varios miles de millones de bytes; la inversión financiera al respecto alcanza una cifra igualmente enorme y no es exagerado afirmar que muchos miles de organizaciones dependen de la operación continuada y eficaz de un sistema de base de datos.

¿Qué es exactamente un sistema de base de datos? En esencia, no es más que un sistema de mantenimiento de registros basados en computadores, es decir, un sistema cuyo propósito general es registrar y mantener información[1]. Tal información debe estar relacionada con cualquier cosa que sea significativa para la organización donde el sistema opera, en otras palabras, cualquier dato necesario para los procesos de toma de decisiones inherentes a la

[1] Los términos datos e información se utilizan como sinónimos en este texto. Algunos autores hacen distinción entre ellos, emplean datos para referirse a los valores registrados físicamente en la base de datos e información par aludir al significado de esos valores según el sentido que les dé un usuario. Esta distinción es muy importante, hasta el punto de que parece preferible hacerla explícita donde sea pertinente, en lugar de atenerse a una diferenciación algo arbitraria entre dos términos en esencia similares.

administración de esa organización. En la figura 1.1 se muestra una representación muy simplificada de un sistema de base de datos. En ella se pretende indicar que un sistema de bases de datos incluye cuatro componentes principales: datos, hardware, software y usuarios. En seguida, se presenta un análisis breve de cada uno de ellos.

1.1.1. Datos

Los datos almacenados en el sistema se dividen en una o más bases de datos. Desde el punto de vista didáctico es más conveniente suponer que solo hay una base de datos almacenados en el sistema. De modo general se usará esta suposición simplificativa, ya que en esencia no invalida nada del análisis siguiente. Sin embargo, como se explica más adelante, existen razones para no aplicar tal restricción en la práctica.

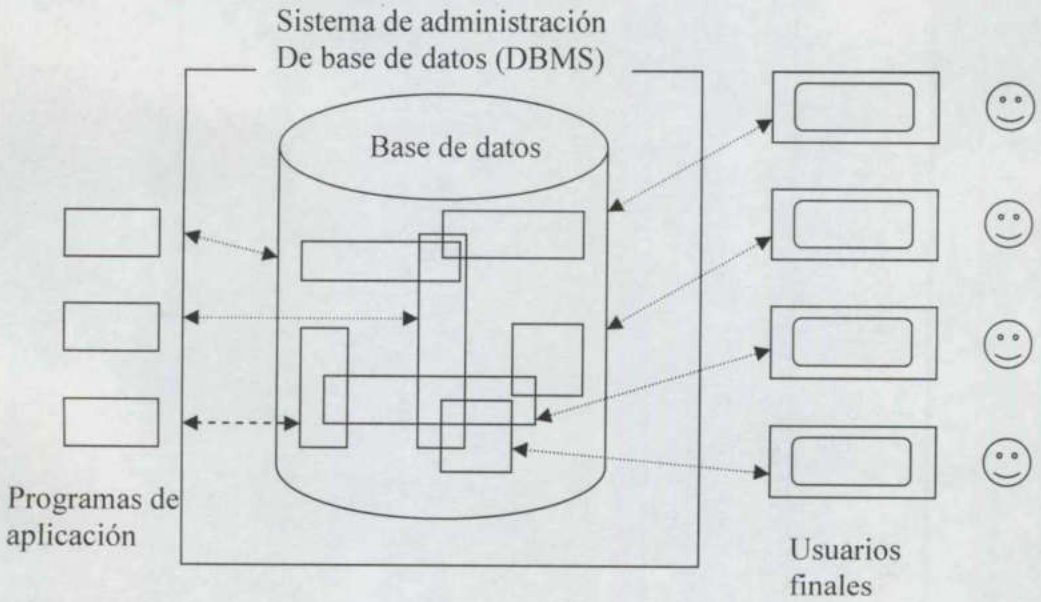


Fig. 1 Representación simplificada de un sistema de base de datos

Una base de datos, pues, en un repositorio de datos almacenados, y, en general, es tanto integrada como compartida.

Por integrada se entiende que la base de datos puede considerarse como una unificación de varios archivos de datos independientes, donde se elimina parcial o totalmente cualquier redundancia entre los mismos; por ejemplo, una base de datos específica puede contener registros de EMPLEADO, que incluyen el nombre, la dirección, el departamento, el salario, etc., y, registros de INSCRIPCIÓN, que representan inscripciones de empleados en cursos de capacitación. Supóngase que para llevar a cabo el proceso de administración de los cursos se necesita conocer el departamento de cada estudiante inscrito. Desde luego, no hay necesidad de incluir esta información redundante en los registros de INSCRIPCIÓN, porque siempre se puede obtener recurriendo a los registros de EMPLEADO correspondientes.

Por compartida se entiende que partes individuales de la base de datos pueden compartirse entre varios usuarios distintos, en el sentido de que cada uno de ellos puede tener acceso a la misma parte de la base de datos (y utilizarla con propósitos diferentes). Tal compartimiento es en verdad consecuencia del hecho de que la base de datos es integrada; en el ejemplo de los registros EMPLEADO/INSCRIPCIÓN citado antes, la información sobre los departamentos en los registros de EMPLEADO es compartida por usuarios del departamento de personal y del departamento de educación. Otra consecuencia del mismo hecho (es decir, que la base de datos es integrada) se advierte en que cualquier usuario específico, por lo general, tendrá acceso tan solo a algún subconjunto de la base de datos completa; además, subconjuntos de diferentes usuarios se trasladarán de muy diversas maneras. En otras palabras, diferentes usuarios percibirán de modos muy distintos una base de datos específica. (Aunque dos usuarios compartan el mismo subconjunto de la base de datos, sus percepciones o vistas de ese subconjunto pueden diferir mucho a nivel de detalle).

La palabra compartida a menudo se amplia para abarcar no sólo al compartimiento antes descrito, sino también al compartimiento concurrente: es decir, la oportunidad de que diversos usuarios accesen en realidad la base de datos, talvez la misma parte de la base de datos inclusive, al mismo tiempo. (Un sistema de base de datos que admite esta forma dee compartimiento en ocasiones se llama sistema de usuarios múltiples).

1.1.2. Hardware

El hardware se compone de los volúmenes de almacenamiento secundario, discos, tambores, etc., donde reside la base de datos, junto con dispositivos asociados como las unidades de control, los canales, etc. (Se supone que la base de datos es demasiado grande para caber en su totalidad en la memoria principal de la computadora)

1.1.3. Software

Entre la base de datos física en sí (es decir, el almacenamiento real de los datos) y los usuarios del sistema existe un nivel de software, que a menudo recibe el nombre de sistema de administración de base de datos o DBMS. Este maneja todas las solicitudes de acceso a la base de datos formuladas por los usuarios. Una función general del DBMS, por tanto, es proteger a los usuarios de la base de datos contra los detalles a nivel de hardware (casi de la misma manera en que los sistemas de lenguajes de programación protegen a los usuarios programadores contra los detalles a nivel de hardware). En otras palabras, el DBMS ofrece una vista de la base de datos que está por encima del nivel de hardware. En otras palabras, el DBMS ofrece una vista de la base de datos que está por encima del nivel de hardware y apoya las operaciones del usuario (tales como obtenga el registro EMPLEADO del empleado Pérez) que se expresan en términos de esa vista de nivel superior.

1.1.4. Usuarios

Se consideran tres clases generales de usuarios.

La primera representa el programador de aplicaciones, encargado de escribir programas de aplicación que utilicen bases de datos. Estos programas de aplicación operan con los datos de tosas las maneras usuales: recuperan información, crean información nueva, suprimen o cambian información existente, etc.[2]

Los programas en sí pueden ser aplicaciones convencionales de procesamiento por lotes o programas en línea diseñados para apoyar a un usuario final que interactúa con el sistema desde una terminal en línea.

La segunda clase de usuario es, entonces, el usuario final que accesa la base de datos desde una terminal. Un usuario final puede emplear un lenguaje de consulta proporcionando como parte integral del sistema o recurrir a un programa de aplicación escrito por un usuario programador que acepte órdenes desde la terminal y a su vez formule solicitudes al DBMS en nombre del usuario final. De cualquier manera, el usuario final puede realizar, en general, todas las funciones de recuperación, creación, supresión y modificación, aunque tal vez no sea falso afirmar que la recuperación es la función mas común de esta clase de usuario.

La tercera clase de usuario la representa el administrador de bases de datos o DBA. El estudio de la función de DBA se presentará mas adelante en la sección 1.5.

Esto completa la descripción preliminar de los aspectos principales de un sistema de base de datos.

1.2. DATOS DE OPERACIÓN

En uno de los primeros materiales de instrucción sobre el tema, Engles se refiere a los datos de una base de datos como datos de operación, distinguiéndolos de los datos de entrada, de

[2] Todas estas funciones se realizan formulando las solicitudes adecuadas al DBMS.

salida y de otras clases. Se ofrece a continuación una versión modificada de la definición original de base de datos formulada por Engles:

> Una base de datos es un conjunto de datos de operación almacenados y utilizados por los sistemas de aplicación de una empresa específica.

Es conveniente explicar esta definición: empresa tan solo es un término genérico que se emplea para designar cualquier organización comercial, científica, técnica o de otra clase que posee un nivel razonable de autosuficiencia: por ejemplo:

Una compañía manufacturera
Un banco
Un hospital
Una universidad
Una dependencia gubernamental

Cualquier empresa necesita disponer de una gran cantidad de datos acerca de su funcionamiento. Estos constituyen sus datos de operación. Los datos de operación de las empresas citadas probablemente incluirían lo siguiente:

Datos de productos
Datos de cuentas
Datos de pacientes
Datos de estudiantes
Datos de planeación

Como se mencionó, los datos de operación no incluyen datos de entrada o de salida, colas de espera de trabajo ni cualquier otra información de índole transitoria. Los datos de entrada se refieren a la información que entra al sistema desde el exterior (casi siempre en tarjetas o desde una terminal); tal información puede ocasionar un cambio en los datos de

operación pero en si misma no forma parte de la base de datos. Del mismo modo, los datos de salida conciernen a mensajes e informes que proceden del sistema (ya sean impresos o desplegados en una terminal); de nuevo, esos datos contienen información derivada de los datos de operación, pero no constituyen parte de la base de datos.

Consideremos con mayor detalle el caso de una compañía manufacturera para explicar el concepto de datos de operación. Tal empresa desea almacenar información sobre los proyectos que tiene en desarrollo; las partes utilizadas en esos proyectos; los proveedores que suministran las partes; los depósitos donde éstas se almacenan; los empleados que trabajan en los proyectos, etc. Estas son las entidades[3] básicas acerca de las cuales se registran datos en la base de datos. Figura 1.2

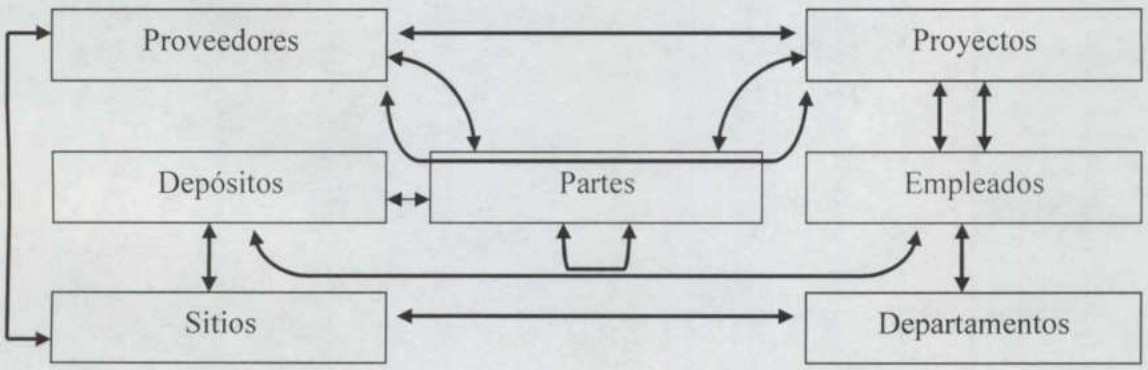


Fig. 1.2 Ejemplo de datos de operación

Es importante señalar que, en general, existen asociaciones que vinculan a las entidades básicas. Estas se representan por flechas de conexión en la figura 1.2, por ejemplo, hay una asociación entre proveedores y partes: cada proveedor suministra ciertas partes, y, por el contrario, cada parte es abastecida por ciertos proveedores. De modo similar, las partes se utilizan en proyectos y, a la inversa, los proyectos emplean partes; las partes se almacenan en depósitos, y éstos almacenan partes, etc. Nótese que todas estas asociaciones son

[3] El término entidad se usa ampliamente en los sistemas de bases de datos para designar cualquier objeto distinguible que pueda representarse en la base de datos.

bidireccionales, es decir pueden considerarse en cualquiera de las dos direcciones. Por ejemplo, la asociación entre empleados y departamentos puede usarse para resolver los dos problemas siguientes: (a) dado un empleado, hallar el departamento correspondiente; (b) dado un departamento, identificar a todos los empleados que le corresponden.

El punto clave de las asociaciones como las que se describen en la figura 1.2 es que forman parte de los datos de operación tanto como la entidades asociadas. Deben, por consiguiente, representarse en la base de datos.

En la figura 1.2 se describen también otros puntos.

1. Aunque la mayoría de las asociaciones del diagrama relacionan a dos tipos de entidad, esto no siempre ocurre así. En el ejemplo aparece una flecha que conecta a tres tipos de entidad (proveedores, partes y proyectos). Esto podría representar el hecho de que ciertos proveedores suministran ciertas partes para ciertos proyectos, lo que en general no equivale a la combinación de la asociación entre proveedores y partes y la asociación entre parts y proyectos. Por ejemplo la información de que el proveedor S2 suministra la parte P4 al proyecto J3, expresa algo mas que la combinación: el proveedor S2 abastece la parte P4 y la parte P4 se utiliza en el proyecto J3; no se puede deducir la primera de estas res asociaciones conociendo únicamente la segunda y la tercera (pero se pueden deducir estas dos últimas si se conoce la primera. En términos más explícitos, si se sabe que S2 suministra P4 y que P4 se emplea en J3, entonces se puede deducir que S2 abastece P4 para algún proyecto Jx, y que algún proveedor Sy suministra P3 para J3, pero no se puede inferir que Jx sea J3 ni que Sy sea S2. Inferencias falsas como éstas son ejemplos de lo que a veces se denomina *trampa de conexión*.

2. El ejemplo también muestra una flecha que solo atañe a un tipo de entidad (las partes). Esto representa una asociación entre una parte y otra: por ejemplo, el hecho de que algunas partes sean componentes de otras (un tornillo es componente de un juego de bisagras, que también se considera una parte).

3. En general, los mismos tipos de entidad pueden relacionarse en varias asociaciones. En el ejemplo citado, los proyectos y los empleados mantienen dos asociaciones. Una podría representar la asociación "trabaja en" (el empleado trabaja en el proyecto), y la otra la asociación "es el administrador de" (el empleado es el administrador del proyecto).

Muchos libros sobre base de datos y sobre sistemas consideran a las entidades y a las asociaciones como dos tipos de objetos en esencia diferentes; sin embargo, una asociación entre entidades puede considerarse como una entidad en sí misma. Si se acepta como definición de entidad "un objeto del que se desea registrar información", entonces una asociación corresponde en realidad a esa definición. Por ejemplo, "la parte P4 esta almacenada en el depósito W8" es una entidad de la que puede desearse registrar información a saber, la cantidad almacenada. Así pues, aquí en este texto se conviene concebir a las asociaciones tan solo como tipos especiales de entidades.

1.3. ¿POR QUÉ UTILIZAR BASES DE DATOS?

¿Por qué una empresa debe optar por almacenar sus datos de operación en una base de datos integrada? La respuesta general a esta pregunta es que un sistema de bases de datos proporciona a la empresa un control centralizado de sus datos de operación que, como el lector ahora podrá advertir, constituyen uno de sus activos más valiosos. Esto contrasta de manera aguda con la situación que prevalece actualmente en muchas empresas, donde a menudo cada aplicación tiene sus propios archivos, y muchas veces sus propias cintas y paquetes de discos particulares, de modo que los datos de operación se hallan muy dispersos y, por tanto, es probable que sean difíciles de controlar.

Lo anterior implica que en una empresa que utilice un sistema de bases de datos debe existir una persona específica cuya responsabilidad central sea controlar los datos de operación. Esta persona es el administrador de bases de datos (DBA), mencionado en la

sección 1.1. En esta sección señalaremos que su trabajo requiere un elevado nivel de destreza técnica y capacidad de entender los requerimientos administrativos a nivel gerencial. En la practica, el DBA puede estar representado por varias personas en lugar de una sola. Es importante advertir que el DBA ocupa un puesto de gran importancia dentro de la empresa.

Consideremos ahora algunas de las ventajas de tener un control centralizado de los datos, según se señalo antes.

1.3.1. Puede reducirse la redundancia.

En sistemas que no usan bases de datos, cada aplicación tiene sus propios archivos privados. Esto a menudo origina enorme redundancia en los datos almacenados, así como desperdicio resultante del espacio de almacenamiento, por ejemplo, una aplicación de personal y otra de registros educativos pueden poseer cada una un archivo que contenga información de departamentos de los empleados. Como se indico en la sección 1.1. estos dos archivos pueden integrarse si el DBA esta consciente de los requerimientos de información para ambas aplicaciones, es decir, si el DBA tiene el control global necesario.

1.3.2. Puede evitarse la inconsistencia

Esto en realidad es corolario del punto anterior. Supóngase que un hecho específico del mundo real, por ejemplo, el hecho de que el empleado E3 trabaja en el departamento D8, se representa por dos entradas distintas en la base de datos, y que el sistema no está al tanto de esta duplicación, habrá entonces algunas ocasiones en que las dos entradas no concuerden, es decir cuando una y solo una de ellas se haya actualizado. En tales circunstancias se dice que la base de datos es inconsistente. Desde luego, una base de datos que se halle en estado de inconsistencia puede suministrar información incorrecta o contradictoria.

No hay duda de que si el hecho específico se representa por una sola entrada (es decir, si la redundancia se elimina), tal inconsistencia no puede ocurrir. Por otra parte, si la redundancia no se suprime, pero se controla (enterando de esto al sistema), entonces éste puede garantizar que la base de datos nunca sea inconsistente para el usuario al asegurar que cualquier cambio hecho a una de las dos entradas se efectué de manera automática en la otra. Este proceso se denomina propagación de actualizaciones donde el término actualización se usa para abarcar todas las operaciones de creación, supresión y modificación. (Adviértase , sin embargo, que pocos sistemas actuales son capaces de propagar las actualizaciones de modo automático, es decir, la mayoría de los sistemas modernos de ninguna manera admiten redundancia controlada).

1.3.3. Los datos pueden compartirse

No solo significa que las aplicaciones existentes pueden compartir los datos de la base de datos, sino también que es factible desarrollar nuevas aplicaciones que operen con los mismos datos almacenados. En otras palabras, las necesidades de datos de las nuevas aplicaciones pueden atenderse sin tener que crear nuevos archivos almacenados.

1.3.4. Pueden hacerse cumplir las normas establecidas

Con un control central de la base de datos, el DBA puede garantizar que se cumplan todas las formas aplicables a la representación de los datos. Las normas aplicables pueden comprender la totalidad o parte de lo siguiente: normas de la compañía, de instalación, departamentales, industriales, nacionales o internacionales. Es muy deseable unificar los formatos de los datos almacenados como ayuda para el intercambio o migración de datos entre sistemas.

1.3.5. Pueden aplicarse restricciones de seguridad.

Al tener jurisdicción completa sobre los datos de operación, el DBA puede,

(a) Asegurar que el único medio de accesar la base de datos sea a través de los canales establecidos y, por tanto,

(b) Definir controles de autorización para que se apliquen cada vez que se intente el acceso a datos sensibles. Diferentes controles pueden establecerse para cada tipo de acceso (recuperación, modificación, supresión, etc.) a cada parte de la información de la base de datos.

1.3.6. Puede conservarse la integridad.

El problema de la integridad es garantizar que los datos de la base de datos sean exactos. La inconsistencia entre dos entradas que representan al mismo hecho es un ejemplo de falta de integridad, que por supuesto, solo ocurre si existe redundancia en los datos almacenados. Aún cuando la redundancia se elimine, empero, la base de datos puede contener aún datos incorrectos, por ejemplo, un empleado puede aparecer como si hubiera trabajado 200 horas semanales, o la lista de números de empleados de un departamento específico puede incluir el número de un empleado inexistente. El control centralizado de la base de datos ayuda a evitar estas situaciones en la medida de lo posible, pues permite al DBA definir procedimientos de validación que habrán de ejecutarse cada vez que se intente una operación de actualización. Es conveniente señalar que la integridad de los datos es m´s importante en un sistema de bases de datos en un sistema de archivos privados, precisamente porque el primero se comparte y porque sin procedimientos de validación adecuados es posible que un programa con errores genere datos incorrectos que afecten a otros programas que utilicen esa información.

1.3.7. Pueden equilibrarse los requerimientos contradictorios.

Cuando conoce los requerimientos globales de la empresa en contraste con los requerimientos de cualquier usuario individual, el DBA puede estructural el sistema de base de datos para brindar un servicio que sea el mejor para la empresa en términos globales. Por ejemplo, puede elegirse una representación de los datos almacenados que ofrezca rápido

acceso a las aplicaciones más importantes a costa de un desempeño de menor calidad en algunas otras aplicaciones.

La mayoría de las ventajas recién enumeradas son muy claras, sin embargo, otro aspecto que no es tan evidentes la provisión de independencia de los datos.

1.4 INDEPENDENCIA DE LOS DATOS

La independencia de los datos puede entenderse con mayor facilidad si primero se analiza el fenómeno opuesto. La mayoría de las aplicaciones actuales son dependientes de los datos. Esto significa que la manera como los datos se organizan en almacenamiento secundario y la manera como se accesan depende de los requerimientos de la aplicación y además que el conocimiento de la organización de los datos y de la técnica de acceso forma parte de la lógica de la aplicación, por ejemplo, puede decidirse que un archivo particular se almacene en forma secuencial con índices. La aplicación entonces, debe saber que el índice existe y conocer la secuencia del archivo, de modo que la estructura interna de la aplicación se construirá con ase en este conocimiento. Además la forma precisa de los diversos procedimientos de acceso y de control de excepciones dentro de la aplicación dependerá ante todo, de los detalles de la interfaz presentada por el software secuencial indicado.

Se dice que una aplicación como ésta es dependiente de los datos porque es imposible cambiar la estructura de almacenamiento o la estrategia de acceso sin afectar la aplicación, tal vez en forma acentuada; por ejemplo, no sería posible reemplazar el archivo secuencial indicado antes mencionado por un archivo direccionado por dispersión sin haber modificaciones importantes a la aplicación. Es interesante advertir por otra parte, que los aspectos de la aplicación que requieren cambios en tal caso son aquellos que se comunican con el software de manejo de archivo y que las dificultades implícitas casi no tienen relación con el problema cuya solución se buscaba al escribir la aplicación , es decir, son dificultades introducidas por la estructura de la interfaz con el software de manejo de archivos.

En un sistema de base de datos, sin embargo, sería muy indeseable permitir que las aplicaciones fueran dependientes de los datos, al menos por las dos razones siguientes:

1. Aplicaciones diferentes requerirán vistas diferentes de los mismos datos. Por ejemplo, supóngase que antes de que la empresa introduzca su base de datos integrada, se tienen dos aplicaciones, A y B, cada una con un archivo propio que contiene el campo "saldo del cliente". Supóngase que la aplicación A registra este valor en notación decimal, mientras que la aplicación B lo hace en sistema binario. Aún será posible integrar los dos archivos y eliminar las redundancias si el DBMS ejecutas todas las conversiones necesarias entre la representación almacenada escogida y la forma en que cada aplicación desee verla, por ejemplo, si la decisión es conservar el valor en notación decimal, entonces todo acceso desde B necesitará una conversión a notación binaria o viceversa.

2. El DBA debe tener libertad de modificar la estructura de almacenamiento o la estrategia de acceso (o ambas) en respuesta al cambio de necesidades sin tener que alterar las aplicaciones existentes; por ejemplo, la empresa puede adoptar nuevas normas; las prioridades de las aplicaciones pueden cambiar, nuevos tipos de dispositivos de almacenamiento pueden aparecer en el mercado, etc. Si las aplicaciones dependen de los datos, esos cambios implican modificaciones correspondientes en los programas, lo que requiere esfuerzos de programación que, de lo contrario, podrían utilizarse para crear aplicaciones nuevas. Por ejemplo, en una instalación grande, cerca del 25% del esfuerzo de programación se dedica a esta clase de actividades de mantenimiento, lo que implica un desperdicio de un recurso muy valioso.

Puede concluirse que la provisión de independencia de los datos es un objetivo esencial de los sistemas de base de datos. Es posible definir la independencia de los datos como loa inmunidad de las aplicaciones a los cambios de la estructura de almacenamiento y de la

estrategia de acceso, lo que implica desde luego, que las aplicaciones no dependen de ninguna estructura de almacenamiento o estrategia de acceso especial.

Empecemos con algunas definiciones.

1.4.1. Campo almacenado

Un campo almacenado es la unidad de datos con nombre mas pequeña que se halla almacenada en la base de datos. Esta contendrá en general muchas ocurrencias o instancias de cada uno de los diversos tipos de campo almacenado, por ejemplo, una base de datos que contenga información sobre las partes tal vez incluya un tipo de campo almacenado llamado numero de partes y habría una ocurrencia de ese campo para cada parte distinta.

1.4.2. Registro almacenado.

Un registro almacenado es un conjunto con nombre de campos almacenados asociados. Una vez mas se hace la distinción entre tipo y ocurrencia. Una ocurrencia o instancia de un registro almacenado se compone de un grupo de ocurrencias de campos almacenados relacionados; por ejemplo, una ocurrencia de registros almacenado puede consistir en una ocurrencia de cada uno de los campos almacenados siguientes: numero de parte, nombre de la parte, color de la parte y peso de la parte. La asociación entre ellos, desde luego, consisten en que todos los datos representan propiedades de alguna parte específica. Se dice que la base de datos contiene múltiples ocurrencias del tipo de registro almacenado "parte". En la mayoría de los sistemas, la ocurrencia de registro almacenado es la unidad de acceso a la base de datos, es decir, la unidad que el DBMS puede recuperar o almacenar en un acceso.

1.4.3. Archivo almacenado

Un archivo almacenado es el conjunto (con nombre) de todas las ocurrencias de un tipo de registro almacenado.[4]

En la mayoría de los sistemas actuales, un registro lógico de una aplicación es idéntico a un registro almacenado, sin embargo como se ha señalado, esto no siempre se observa en un sistema de bases de datos, porque el DBA puede efectuar cambios en la estructura de almacenamiento, es decir, en los campos y registros almacenados, en tanto que los campos y registros lógicos correspondientes no cambian.

[4] Para simplificar la exposición se ignora la posibilidad de un archivo almacenado que contenga mas de un tipo de registro almacenado. Esta suposición simplificadora no afecta en esencia ninguna parte del análisis ulterior.

2 ARQUITECTURA DE UN SISTEMA DE BASE DE DATOS

2.1 *INTRODUCCION*

Ahora estamos en condiciones de presentar la arquitectura para un sistema de base de datos. Nuestro objetivo al presentar esta arquitectura es ofrecer una infraestructura en la que puedan basarse. Dicha infraestructura resulta útil para describir los conceptos generales de las bases de datos y para explicar la estructura de sistemas de bases de datos específicos; pero no afirmamos que todo sistema pueda coincidir enteramente con esta infraestructura en particular, ni queremos sugerir que esta arquitectura represente la única infraestructura posible. En particular, es probable que los sistemas "pequeños" no manejen todos los aspectos de la arquitectura. Sin embargo, la arquitectura parece ajustarse bastante bien a la mayoría de los sistemas, además, es prácticamente idéntica a la arquitectura propuesta por el Grupo de estudio en Sistemas de Administración de Bases de datos de ANSI/SPARC, sin embargo, decidimos no seguirla en todos sus detalles.

2.2 LOS TRES NIVELES DE LA ARQUITECTURA.

La arquitectura ANSI/SPARC se divide en tres niveles, conocidos como interno, conceptual y externo, respectivamente (Vea la figura 2.1). Hablando en términos generales:

- El **nivel interno** (también conocido como el nivel físico) es el que está más cerca del almacenamiento físico; es decir, es el que tiene que ver con la forma en que los datos están almacenados físicamente.

- El **nivel externo** (también conocido como el nivel lógico de usuario) es el más próximo a los usuarios; es decir, el que tiene que ver con la forma en que los usuarios individuales ven los datos.

- El **nivel conceptual** (también conocido como el nivel lógico de la comunidad, o en ocasiones sólo como el nivel lógico, sin calificar) es un nivel de indirección entre los otros dos.

Observe que el nivel externo tiene que ver con las percepciones de usuarios individuales, mientras que el nivel conceptual tiene que ver con la percepción de una comunidad de usuarios.

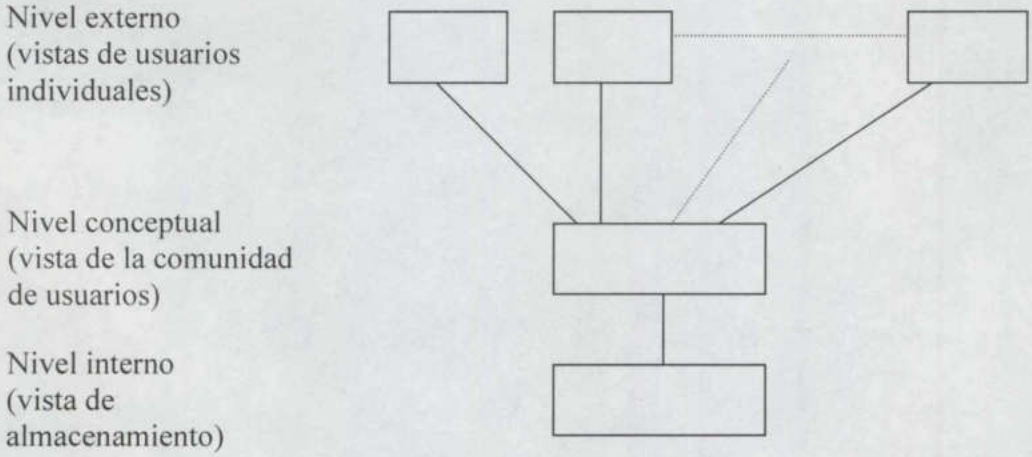


Figura 2.1 Los tres niveles de la arquitectura.

En otras palabras, habrá muchas "vistas externas" distintas, cada una consistente en una representación más o menos abstracta de alguna parte de la base de datos total, y habrá precisamente una "vista conceptual" que del mismo modo consiste en una representación abstracta de la base de datos en su totalidad. [1] (Recordando que la mayoría de los usuarios no se interesarán en toda la base de datos, sino sólo en una parte limitada de la misma.) en forma similar, habrá precisamente una "vista externa" que represente a la base de datos tal como está almacenada físicamente.

Un ejemplo hará más claras estas ideas. La figura 2.2 muestra la vista conceptual, la vista interna correspondiente y dos de las vistas externas (una para un usuario PL/I y otra para un usuario de COBOL), todas ellas para una base de datos personal sencilla. Por supuesto, el ejemplo es completamente hipotético – no pretende reflejar ningún sistema real – y hemos omitido deliberadamente muchos detalles irrelevantes. Explicación:

- En el nivel conceptual, la base de datos contiene información concerniente a un tipo de entidad denominada EMPLEADO. Cada empleado individual tiene un NUMERO_EMPLEADO (de seis caracteres), un NUMERO_DEPARTAMENTO (de cuatro caracteres) y un SALARIO (de cinco dígitos decimales).

- En el nivel interno, los empleados están representados por un tipo de registro denominado EMP_ALMACENADO, de veinte bytes de longitud. EMP_ALMACENADO contiene cuatro campos almacenados: un prefijo de seis bytes (que presumiblemente contiene información de control como los indicadores o apuntadores) y tres campos de datos correspondientes a las tres propiedades de los empleados. Además, los registros de EMP_ALMACENADO están indexados sobre el campo EMP# por medio de un índice de nombre EMPX, cuya definición no se muestra.

[1] Aquí por abstracto simplemente queremos decir que la representación en cuestión comprende construcciones como los registros y campos que están más orientadas a los usuarios, a diferencia de las construcciones como los bits o los bytes que están más orientadas a la máquina.

Externo (PL/I)		Externo (COBOL)
DCL	1 EMPP,	01 EMPC.
	2 EMP# CHAR(6),	02 EMPNO PIC X(6).
	2 SAL FIXED BIN (31),	02 EMPNO PIC X(4).
Conceptual		
EMPLEADO		
	NUMERO_EMPLEADO	CHARACTER(6)
	NUMERO_DEPARTAMENTO	CHARACTER(4)
	SALARIO	NUMERIC(5)
Interno		
	EMP_ALMACENADO	BYTES=20
	PREFIJO	TYPE=BYTES(6), OFFSET=0
	EMP#	TYPE=BYTES(6), OFFSET=6, INDEX=EMPX
	DEPT#	TYPE=BYTES(4), OFFSET=12
	SUELDO	TYPE=FULLWORD, OFFSET=16

Figura 2.2 Un ejemplo de los tres niveles.

- El usuario de PL/I tiene una vista externa de la base de datos en la que cada empleado esta representado por un registro PL/I que contiene dos campos (los números de departamento que son de interés para este usuario y por lo tanto fueron omitidos). El tipo del registro esta definido por una declaración de estructura PL/I ordinaria según las reglas normales de PL/I.

- En forma similar, el usuario de COBOL tiene una vista externa en la que cada empleado esta representado por un registro de COBOL, que una vez más contiene dos campos (esta vez fueron omitidos los salarios). El tipo de registro está definido por una descripción ordinaria de registro COBOL, de acuerdo con las reglas normales de COBOL.

Observe que los elementos correspondientes de los datos pueden tener diferentes nombres en distintos puntos. Por ejemplo, el numero de empleado se denomina EMP# en la vista externa de PL/I, EMPNO en la vista externa de COBOL, NUMERO_EMPLEADO en la vista conceptual y nuevamente EMP# en la vista interna. Desde luego, el sistema debe estar al tanto de las correspondencias; por ejemplo, debemos indicar que el campo EMPNO de COBOL se deriva del campo conceptual NUMERO_EMPLEADO, el aual se deriva a su ve del campo almacenado EMP# al nivel interno. Dichas correspondencias, o transformaciones, no se muestran de manera explícita en la figura 2.2.

Sin embargo, sería útil indicar cómo son concebidos típicamente los tres niveles de la arquitectura en un sistema relacional en particular.

- Primero, el nivel conceptual en dicho sistema será definitivamente relacional, en el sentido de que los objetos visibles en ese nivel serán tablas relacionales y los operadores serán de tipo relacional (incluyendo los operadores restringir y proyectaren particular).
- Segundo, una vista externa dada también será casi siempre relaiconal, o algo muy parecido a ello; por ejemplo, las declaraciones de registro de PL/I y COBOL de la figura 2.2 podrían ser consideradas a grandes rasgos como análogas de las declaraciones de PL/I y COBOL, respectivamente, de una tabla relacional en un sistema relacional.

Nota: debemos mencionar de paso que el término "Vista externa" (a menudo abreviado solamente como "vista") tiene por desgracia un significado más bien específico en contextos relaciónales y que éste no es idéntico al significado que se le asigna en este capítulo.

- Tercero, el nivel interno no será relacional, ya que los objetos en ese nivel no serán solo tablas relacionales(almacenadas); en vez de ello, serán los mismos tipos de objetos que se encuentran en el nivel interno de cualquier otro tipo de sistema (registros almacenados, apuntadores, índices, tablas de dispersión, etc.). de hecho, el

modelo relacional como tal no tiene nada en absoluto que decir acerca del nivel interno; para repetir lo dicho en el tema anterior, tiene que ver conla forma en que la base de datos se presenta ante el usuario.

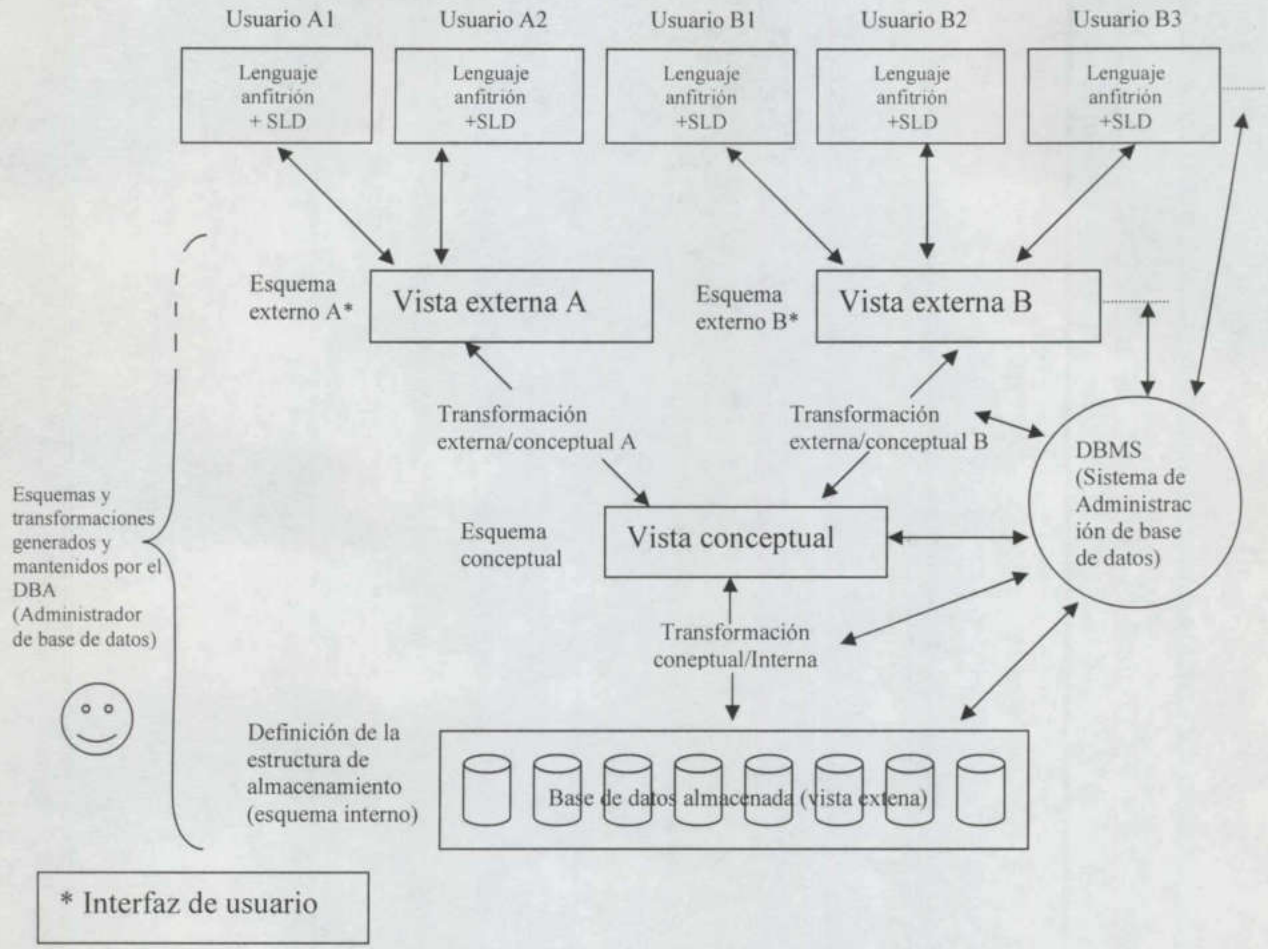


Figura 2.3 Arquitectura detallada del sistema.

Ahora procederemos a explicar con más detalle los tres niveles de la arquitectura, comenzando con el nivel externo. A lo largo de nuestra explicación haremos constantes referencias a la figura 2.3, la cual muestra los principales componentes de la arquitectura y sus interrelaciones.

2.3 EL NIVEL EXTERNO

El nivel externo es el nivel del usuario individual. Como se explico en el tema anterior, un usuario dado puede ser un programador de aplicaciones o bien un usuario final con cualquier grado de sofisticación. (El DBA es un importante caso especial, pero a diferencia de otros usuarios, el DBA también necesitará interesarse en los niveles conceptual e interno. Vea las dos secciones siguientes.).

Cada usuario tiene a su disposición un lenguaje:

- Para el programador de aplicaciones, éste será ya sea un lenguaje de programación convencional (por ejemplo, PL/I, C++, Java) o bien un lenguaje de tipo propietario que sea específico al sistema en cuestión. A menudo, a estos lenguajes de tipo propietario se les denomina lenguajes de "cuarta generación" (4GLs); siguiendo las – confusas – bases de que:
    - El código de máquina, el lenguaje ensamblador y los lenguajes como PL/I pueden ser vistos como tres generaciones de lenguajes anteriores y,
    - Los lenguajes de tipo propietario representan la misma clase de mejora sobre los lenguajes de tercera generación (3GLs) que la que tuvieron estos lenguajes sobre el lenguaje ensamblador y la que tuvo el lenguaje sobre el código de máquina.

- Para el usuario final, el lenguaje será ya sea un lenguaje de consulta o bien algún lenguaje de finalidad específica, tal vez controlado por formularios o por menús, confeccionado para los requerimientos de ese usuario y manejado por algún programa de aplicación en línea.

En nuestro caso, lo importante acerca de dichos lenguajes es que incluirán un sublenguaje de datos; es decir, un subconjunto del lenguaje total que se ocupe específicamente de los objetos de los objetos y operaciones de la base de datos. Se dice que el sublenguaje de datos (abreviado como SLD en la figura 2.3) esta incrustado

dentro de su lenguaje anfitrión correspondiente. El lenguaje anfitrión es el responsable de proporcionar diversas propiedades que no son específicas de la base de datos, como las variables locales, las operaciones de cálculo, la lógica de bifurcación, etc. Un sistema determinado podría manejar cualquier cantidad de lenguajes anfitrión y cualquier número de sublenguajes de datos; sin embargo, un sublenguaje de datos específico soportado por casi todos los sistemas actuales es el lenguaje SQL. La mayoría de dichos sistemas permiten que SQL sea utilizado de manera interactiva, como un lenguaje de consulta independiente, e incrustado en otros lenguajes como PL/I o Java.

Ahora bien, aunque para fines de la arquitectura es conveniente distinguir entre el sublenguaje y el lenguaje anfitrión que lo contiene, ambos podrían no ser distintos en lo que al usuario concierne; y de hecho, desde el punto de vista del usuario, es probablemente mejor que no lo sean. Si no son distintos, o si difícilmente pueden distinguirse, decimos que están fuertemente acoplados. Si son clara y fácilmente separables, decimos que están débilmente acoplados. Aunque algunos sistemas comerciales (en especial los orientados a objetos) si manejan el acoplamiento fuerte, la mayoría no lo hace; en particular, los sistemas SQL generalmente solo manejan el acoplamiento débil. (El acoplamiento fuerte ofrece un conjunto más uniforme de propiedades para el usuario, aunque obviamente implica más esfuerzo por parte de los desarrolladores del sistema, un hecho que presumiblemente cuenta para el statu quo).

En principio, cualquier sublenguaje de datos determinado es en realidad una combinación de por lo menos dos lenguajes subordinados: un DDL (Lenguaje de definición de datos), que permite la definición o declaración de objetos de base de datos, y un DML (lenguaje de manipulación de datos), que permite la manipulación o procesamiento de dichos objetos. Por ejemplo, considere al usuario de PL/I de la figura 2.2. el sublenguaje para ese usuario consiste en aquellas características de PL/I que se usan para comunicarse con el DBMS.

- La parte del DDL consiste en aquellas construcciones declarativas de PL/I necesarias para declarar objetos de base de datos; la propia instrucción DECLARE(DCL), ciertos tipos de datos de PL/I, tal vez extensiones especiales de PL/I para manejar nuevos objetos que no maneja el PL/I existente.
- La parte del DML consiste en aquellas instrucciones ejecutables de PL/I que transfieren información hacia y desde la base de datos; de nuevo, que incluyen tal vez nuevas instrucciones especiales.

Volviendo a la arquitectura, ya indicamos que un usuario individual se interesa generalmente sólo por alguna parte de toda la base de datos; es más, la vista que el usuario tiene de esa parte normalmente será un poco abstracta al compararla con la forma en que los datos están almacenados físicamente. El término ANSI/SPARC utilizado para la vista de un usuario individual es el de vista externa. Por lo tanto, una vista externa es el contenido de una base de datos como lo ve algún usuario en particular (es decir, para ese usuario, la vista externa es la base de datos). Por ejemplo, un usuario del Departamento de Personal podría ver a la base de datos como una colección de ocurrencias de los registros de empleado y departamento, y podría desconocer las ocurrencias de los registros de parte y proveedor que ven los usuarios del departamento de compras.

Entonces, en general, una vista externa consiste en muchas ocurrencias de cada uno de los tipos de registro externo (que no es necesariamente lo mismo que un registro almacenado).[2] El sublenguaje de datos del usuario es´ta definido en términos de registros externos; por ejemplo, una operación de recuperación del DML recuperará ocurrencias de registros externos, no ocurrencias de registros almacenados. Nota: ahora podemos ver que el término "registro lógico".

Cada vista externa está definida por medio de un esquema externo, el cual consiste básicamente en definiciones de cada uno de los diversos tipos de registros externos de esa

[2] Aquí, damos por hecho que toda la información está representada a un nivel externo en forma de registros. Sin embargo, algunos sistemas permiten que la información sea representada en otras formas en vez de, o también en forma de, registros. Para que un sistema emplee dichos métodos alternativos, será necesario adecuar las definiciones y explicaciones dadas en esta sección. Se aplican también observaciones similares a los niveles conceptual e interno.

vista (observe una vez más la figura 2.2, un par de ejemplos sencillos). El esquema externo fue escrito utilizando la parte DDL del sublenguaje de datos del usuario. (De ahí que en ocasiones se haga referencia a ese DDL como DDL externo). Por ejemplo, el tipo de registro externo de empleado podría definirse como un campo de número de empleado con seis caracteres, más un campo de salario con cinco dígitos (decimales) y así sucesivamente. Además, debe haber una definición de la transformación entre el esquema externo y el esquema conceptual subyacente (vea la siguiente sección).

2.4 EL NIVEL CONCEPTUAL

La vista conceptual es una representación de todo el contenido de la información de la base de datos, de nuevo (a igual que con la vista externa) en una forma un poco abstracta comparada con la firma en la que por lo regular se almacenan los datos físicamente. También será muy diferente de la forma en que cualquier usuario específico ve los datos. En términos generales, la vista conceptual pretender ser una vista de los datos "tal como son", en vez de tal como los usuarios están obligados a verlos debido a las limitaciones (por ejemplo) del lenguaje o el hardware en particular que pudiera utilizar.

La vista conceptual consiste en muchas ocurrencias de varios tipos de registro conceptual. Por ejemplo, podría consistir en un conjunto de ocurrencias de los registros de departamento, más un conjunto de ocurrencias de los registros de empleado, más un conjunto de ocurrencias de los registros de proveedor, mas un conjunto de ocurrencias de los registros de parte (y así sucesivamente). Por otra parte, un registro conceptual no es necesariamente lo mismo que un registro externo, ni que un registro almacenado.

La vista conceptual está definida por medio del esquema conceptual, el cual comprende definiciones de cada uno de los diversos tipos de registros conceptuales (de nuevo, consulte la figura 2.2 para ver un ejemplo sencillo). El esquema conceptual está escrito con otro lenguaje de definición de datos, el DDL conceptual. Si se va a lograr la independencia física de los datos, entonces las definiciones conceptuales de DDL no deben comprender en lo absoluto ninguna consideración de la representación física ni de la técnica de acceso;

deben ser únicamente definiciones del contenido de la información. Por lo tanto en el esquema conceptual no debe haber ninguna referencia para la representación de campos almacenados, la secuencia de registros almacenados, los índices, los esquemas de dispersión, los apuntadores o cualquier otro detalle de almacenamiento y acceso. Si el esquema conceptual se hace verdaderamente independiente de los datos, entonces los esquemas externos, que están definidos en términos del esquema conceptual, también serán forzosamente independientes de los datos.

2.5 EL NIVEL INTERNO

El tercer nivel de la arquitectura es el nivel interno. La vista interna es una representación de bajo nivel de toda la base de datos y consiste en muchas ocurrencias de cada uno de los diversos tipos de registros internos. "Registro interno" es el término de ANSI/SPARC para la construcción que hemos venido llamando registro almacenado . por lo tanto, la vista interna está todavía distante del nivel físico, ya que no tiene que ver con términos como registros físicos – también denominados bloques o páginas- ni con ninguna consideración específica de los dispositivos, como el tamaño de los cilindros o de las pistas. En otras palabras, la vista interna en efecto da por hecho un espacio de direcciones lineal infinito; los detalles de cómo el espacio de direcciones se asocia con el almacenamiento físico, son en gran medida específicos del sistema y se omiten deliberadamente de la arquitectura general. Nota: El bloque o página, es la unidad de E/S; es decir, es la cantidad de datos transferidos entre el almacenamiento secundario y la memoria principal en una sola operación de E/S. Los tamaños típicos de página son 1 KB, 2 KB, o 4 KB(1 KB = 1024 Bytes).

La vista interna se describe por medio del esquema interno, el cual no solo define los diversos tipos de registros almacenados sino que especifica también qué índices existen, como están representados los campos almacenados, en qué secuencia están dichos registros, etcétera. (Una vez más observe un ejemplo sencillo en la figura 2.2). el esquema interno está escrito utilizando otro lenguaje más de definición de datos: el DDL interno. Nota: aquí usaremos normalmente los términos más intuitivos "Estructura de almacenamiento" o

"Base de datos almacenada" en lugar de "Vista interna", así como el término "definición de la estructura de almacenamiento" en lugar de "esquema interno".

Para termina, señalamos que, en ciertas situaciones excepcionales, a los programas de aplicación – en particular, las aplicaciones de utilería – se les podría permitir operar directamente en nivel interno en vez del nivel externo. Sobra decir que no es recomendable esta práctica, pues representa un riesgo para la seguridad (ya que se ignoran las restricciones de seguridad) y un riesgo para la integridad (debido a que, de igual manera, se ignoran las restricciones de integridad). Además, para iniciar, el programa será dependiente de los datos aunque en ocasiones, ésta podría ser la única forma de obtener la funcionalidad o el rendimiento requeridos (tal como le sucede al usuario de un lenguaje de programación de alto nivel que ocasionalmente tendría que descender al lenguaje ensamblador para satisfacer ciertos objetivos de funcionalidad o rendimiento).

2.6 TRANSFORMACIONES

Además de los tres niveles como tales, la arquitectura de la figura 2.3 comprende ciertas transformaciones; en general, una transformación conceptal/interna y varias transformaciones externas/conceptual:

- La transformación conceptual/interna define la correspondencia entre la vista conceptual y la base de datos almacenada, y especifica como están representados los registros y campos conceptuales en el nivel interno. Si se modifica la estructura de la base de datos – es decir, si se hace un cambio a la definición de la estructura de almacenamiento -, entonces por consiguiente será necesario modificar la transformación conceptual/interna, de manera que el esquema conceptual pueda permanecer invariable. (Por supuesto es responsabilidad del DBA administrar dichos cambios). En otras palabras, los efectos de dichos cambios deben aislarse por debajo del nivel conceptual, a fin de preservar la independencia física de los datos.
- La transformación externa/conceptual define la correspondencia entre una vista eterna en particular y la vista conceptual. En general, las diferencias que puedan

existir entre estos dos niveles son análogas a aquellas que puedan existir entre la vista conceptual y la base de datos almacenada. Por ejemplo, los campos pueden tener diferentes tipos de datos, los nombres de los registros y campos pueden ser cambiados, varios campos conceptuales pueden combinarse en un solo registro externo (virtual); etcétera. Puede existir cualquier cantidad de vistas externas al mismo tiempo; cualquier número de usuarios puede compartir una vista externa dada; es posible traslapar diferentes vistas externas.

Nota: Debe quedar claro que así como la transformación conceptual interna es la clave para la independencia física de los datos, también las transformaciones externas/conceptual son la clave pára la independencia lógica de los datos. Como vimos en el capítulo anterior, un sistema proporciona la independencia física de los datos si los usuarios y los programas de usuario son inmunes a los cambios en la estructura física de la base de datos almacenada. De igual manera, un sistema proporciona la independencia lógica de los datos si los usuarios y los programas de usuario también son inmunes a los cambios en la estructura lógica de la base de datos (lo que significa cambios a nivel conceptual o "lógico de la comunidad".

Por cierto, la mayoría de los sistemas permiten que la definición de ciertas vistas externas se exprese en términos de otras (en efecto, mediante transformaciones externas/externas), en vez de requerir siempre una definición explícita de la transformación al nivel conceptual; una característica útil si varias vistas externas son parecidas entre sí. En particular, los sistemas relacionales brindan dicha posibilidad.

2.7 EL ADMINISTRADOR DE BASE DE DATOS

El DA (el administrador de datos) es la persona que toma las decisiones de estrategia y política con respecto a los datos de la empresa y el DBA (Administrador de base de datos) es la persona que proporciona el apoyo técnico necesario para implementar dichas decisiones. Por lo tanto el DBA es el responsable del control general del sistema al nivel

técnico. Ahora podemos describir con un poco más de detalle algunas de las tareas del DBA. Es general, estas tareas comprenden al menos todas las siguientes:

- Definir el esquema conceptual. Es trabajo del administrador de datos decidir exactamente qué información contendrá la base de datos, en otras palabras, identificar las entidades de interés para la empresa e identificar la información que hay que registrar acerca de dichas entidades. Por lo regular a este proceso se le conoce como diseño lógico – en ocasiones conceptual – de la base de datos. Una vez que el administrador decidió el contenido de la base de datos a un nivel abstracto, entonces el DBA creará el esquema conceptual correspondiente, utilizando el DLL, conceptual. El DBMS usará la forma objeto (compilada) de ese esquema para responder a las peticiones de acceso. La forma fuente (sin compilar) actuará como documento de referencia para los usuarios del sistema. Nota: En la práctica las cosas pueden no ser tan claras como sugieren los señalamientos anteriores. En algunos casos, el administrador de datos podría crear directamente el esquema conceptual. En otras, el DBA podría hacer el diseño lógico.

- Definir el esquema interno. El DBA también debe decidir la forma en que van a ser re´resentados los datos en la base de datos almacenada. A este proceso se le conoce comúnmente como diseño físico de la base de datos. [3] Una vez realizado el diseño físico, el DBA deberá crear la definición de la estructura de almacenamiento correspondiente (es decir, el esquema interno), utilizando el DDL interno. Además, también deberá definir la transformación conceptual/interna asociada. En la práctica, es factible que uno de los dos DDLs (el conceptual o el interno; pero más probablemente el primero) incluya los medios para definir esa transformación, aunque las dos funciones (crear el esquema y definir la transformación) deben ser claramente separables. Al igual que el esquema conceptual, tanto el esquema interno como la transformación correspondiente existirán en las formas fuente y objeto.

[3] Observe la secuencia: Primero decida qué datos desea, luego decida cómo representarlos en el almacenamiento. El diseño físico sólo deberá hacerse después de realizar el diseño lógico.

- Establecer un enlace con los usuarios. Es asunto del DBA enlazarse con los usuarios para asegurar que los datos necesarios estén disponibles y para escribir (o ayudar a escribir) los esquemas externos necesarios, utilizando el DDL externo aplicable. (Como ya mencionamos, un sistema dado podría manejar varios DDLs externos distintos.) Además, también es necesario definir las transformaciones externas/conceptual correspondientes. En la práctica, es probable que el DDL externo incluya los medios para especificar dichas transformaciones, pero, una vez más los esquemas y las transformaciones deben ser claramente separables. Cada esquema externo, con la transformación correspondiente, existirá en las formas tanto fuente como objeto.

Otros aspectos de la función de enlace con los usuarios incluyen la asesoría sobre el diseño de aplicaciones; una capacitación técnica; ayuda en la determinación y resolución de problemas; así como otros servicios profesionales similares.

- Definir las restricciones de seguridad y de integridad. Éstas pueden ser bistas como parte del esquema conceptual. El DDL conceptual debe incluir facilidades para especificar dichas restricciones.

- Definir las políticas de vaciado y recarga. Una vez que una empresa se compromete con un sistema de base de datos, se vuelve drásticamente dependiente del funcionamiento exitoso de dicho sistema. En el caso de que se produzca un daño en cualquier parte de la base de datos – ocasionado, por ejemplo, por un error humano o por una falla en el hardware o en el sistema operativo – resulta esencial poder reparar los datos afectados con el mínimo de demora y con tan poco efecto como sea posible sobre el resto del sistema. Por ejemplo, de manera ideal no debería afectarse la disponibilidad de los datos que no fueron afectados. El DBA debe definir e implementar un esquema apropiado de control de daños que comprenda :

- La descarga o "vaciado" periódico de la base de datos en un dispositivo de almacenamiento de respaldo y
- La recarga de la base de datos cuando sea necesario, a partir del vaciado mas reciente.

Por cierto, la necesidad de una rápida reparación de los datos es una de las razones por las que podría ser buena idea repartir todos los datos en varias bases de datos, en vez de mantenerlos todos en un mismo lugar; una base de datos individual podría muy bien definir una unidad para fines de vaciado y de recarga. En este sentido, observe que ya existen los sistemas de terabytes[4] - es decir, los sistemas comerciales que almacenan más de un billón de bytes, aproximadamente – y se predice que los sistemas futuros serán mucho más grandes. Sobra decir que tales sistemas BLVD (Base de datos muy grande) requieren de una administración muy cuidadosa y sofisticada, en especial si hay necesidad de una disponibilidad continua (lo que sucede normalmente). Sin embargo, para efectos de simplicidad seguiremos hablando como si de hecho solo existen una base de datos individual.

- Supervisar el rendimiento y responder a los requerimientos cambiantes. El DBA es el responsable de organizar el sistema de tal manera que se obtenga el rendimiento "ideal para la empresa" y de hacer los ajustes apropiados – es decir, afinar – conforme las necesidades cambien. Por ejemplo, podría ser necesario reorganizar de vez en cuando la base de datos almacenada para asegurar que los niveles de rendimiento se mantengan aceptables. Todo cambia al nivel interno de almacenamiento físico debe estar acompañado por el cambio correspondiente en la definición de la transformación conceptual/interna, de manera que el esquema conceptual permanezca constante.

[4] 1024 bytes = 1 KB (kilobyte); 1024 KB = 1 MB (Megabyte); 1024 MB = 1 GB (gigabyte); 1024 GB =1 B (terabyte); 1024 TB= 1 PB (petabyte); 1024 PB = 1 EB(exabyte). Tome nota que un gigabyte equivale aproximadamente al mil millones de bytes y que algunos textos en ingles emplean en ocasiones la abreviatura BB en lugar de GB.

Desde luego, la anterior no es una lista detallada; simplemente pretende dar una idea del alcance y naturaleza de las responsabilidades del DBA.

2.8 EL SISTEMA DE ADMINISTRACIÓN DE BASE DE DATOS

El DBMS es el software que maneja todo acceso a la base de datos. De manera conceptual, lo que sucede es lo siguiente:

1.- un usuario emite una petición de acceso, utilizando algún sublenguaje de datos específico (por lo regular SQL).

2.- El DBMS intercepta esa petición y la analiza.

3.- El DBMS inspecciona, en su momento, (las versiones objeto de) el esquema externo para este usuario, la transformación externa/conceptual correspondiente, el esquema conceptual, la transformación conceptual/interna y la definición de la estructura de almacenamiento.

4.- El DBMS ejecuta las operaciones necesarias sobre la base de datos almacenada.

A manera de ejemplo, considere lo que implica la recuperación de una ocurrencia de un registro externo en particular. En general, los campos serán solicitados desde barias ocurrencias de registros conceptuales, y cada ocurrencia de un registro conceptual solicitará a su vez campos de barias ocurrencias de registros almacenados. Entonces, de manera conceptual, el DBMS debe primero recuperar todas las ocurrencias solicitadas de los registros almacenados, luego construir citadas de los registros externos. En cada etapa, podrían ser necesarias conversiones de tipos de datos u otras.

Desde luego, la descripción anterior está muy simplificada; en particular, implica que todo el proceso es interpretativo, ya que asume que todo el proceso de analizar la petición, inspeccionar los diversos esquemas, etcétera, se realiza en tiempo de ejecución. La

interpretación, por su parte, a menudo implica un rendimiento deficiente debido a la sobrecarga del tiempo de ejecución. Sin embargo, en la práctica podría ser posible compilar las peticiones de acceso antes del tiempo de ejecución.

Analicemos ahora las funciones del DBMS con un poco mas de detalle. Dichas funciones comprenderán por lo menos el manejo de todas las siguientes (consulte la figura 2.4):

- Definición de datos.

El DBMS debe ser capaz de aceptar definiciones de datos (esquemas externos, el esquema conceptual, el esquema interno y todas las transformaciones respectivas) en la forma fuente y convertirlas a la forma objeto correspondiente. En otras palabras, el DBMS debe incluir entre sus componentes un procesador DDL, o compilador DDL, para cada uno de los versos DDLs (lenguajes de definición de datos). El DBMS también debe entender las definiciones DDL, en el sentido que, por ejemplo, entienda que los registros externos EMPLEADO incluyen un campo SALARIO; entonces, debe poder utilizar este conocimiento para analizar y responder a las peticiones de manipulación de datos (por ejemplo, "Obtener todos los empleados con salario <$50,000").

- Manipulación de datos.

El DBMS debe ser capaz de manejar peticiones para recuperar, actualizar o eliminar datos existentes en la base de datos o agregar nuevos datos a ésta. En otras palabras, el DBMS debe incluir un componente procesador DML o compilador DML para tratar con el DML (lenguaje de manipulación de datos).

- En general, las peticiones DML pueden ser "planeadas" y "no planeadas".

a) Una petición planeada es aquella cuya necesidad fue prevista antes del momento de ejecutar la petición. Probablemente el DBA habrá afinado el diseño físico de la base de datos de tal forma que garantice un buen desempeño para las peticiones planeadas.

b) En contraste, una petición no planeada es una consulta ad hoc; es decir, una petición para la que no se previó por adelantado su necesidad, sino que en vez de ello, surgió sin pensarlo. El diseño físico de la base de datos podría o no ser el adecuado para la petición específica en consideración.

Para utilizar la terminología presentada en el capítulo anterior, las peticiones planeadas son características de las aplicaciones "operacionales" o de "producción", mientras que las peticiones no planeadas son características de las aplicaciones de "apoyo a la toma de decisiones". Además, peticiones planeadas serán emitidas generalmente desde programas de aplicación preescritos, mientras que las no planeadas, por definición, serán emitidas en forma interactiva mediante algún procesador del lenguaje de consulta.

- Optimización y ejecución.

Las peticiones DML, planeadas o no planeadas, deben ser procesadas por el componente optimizador, cuya finalidas es determinar una forma eficiente de implementar la petición. Las peticiones optimizadas se ejecutan entonces bajo el control del administrador en tiempo de ejecución. Nota: En la práctica, el administrador en tiempo de ejecución recurrirá probablemente a algún tipo de administrador de archivos para acceder a los datos almacenados.

- Seguridad e integridad de los datos.

El DBMS debe vigilar las peticiones del usuario y rechazar todo intento de violar las restricciones de seguridad y de integridad definidas por el DBA. Estas tareas pueden realizarse durante el tiempo de compilación, de ejecución o entre ambos.

- Recuperación de los datos y recurrencia.

El DBMS – o más probablemente, algún otro componente de software relacionado, denominado comúnmente administrador de transacciones o monitor de procesamiento de transacciones (monitor PT) – debe imponer ciertos controles de recuperación y

concurrencia. Los detalles de estos aspectos del sistema están fuera del alcance de este capítulo. Nota: El administrador de transacciones no se muestra en la figura 2.4 debido a que, por lo regular, no forma parte del DBMS como tal.

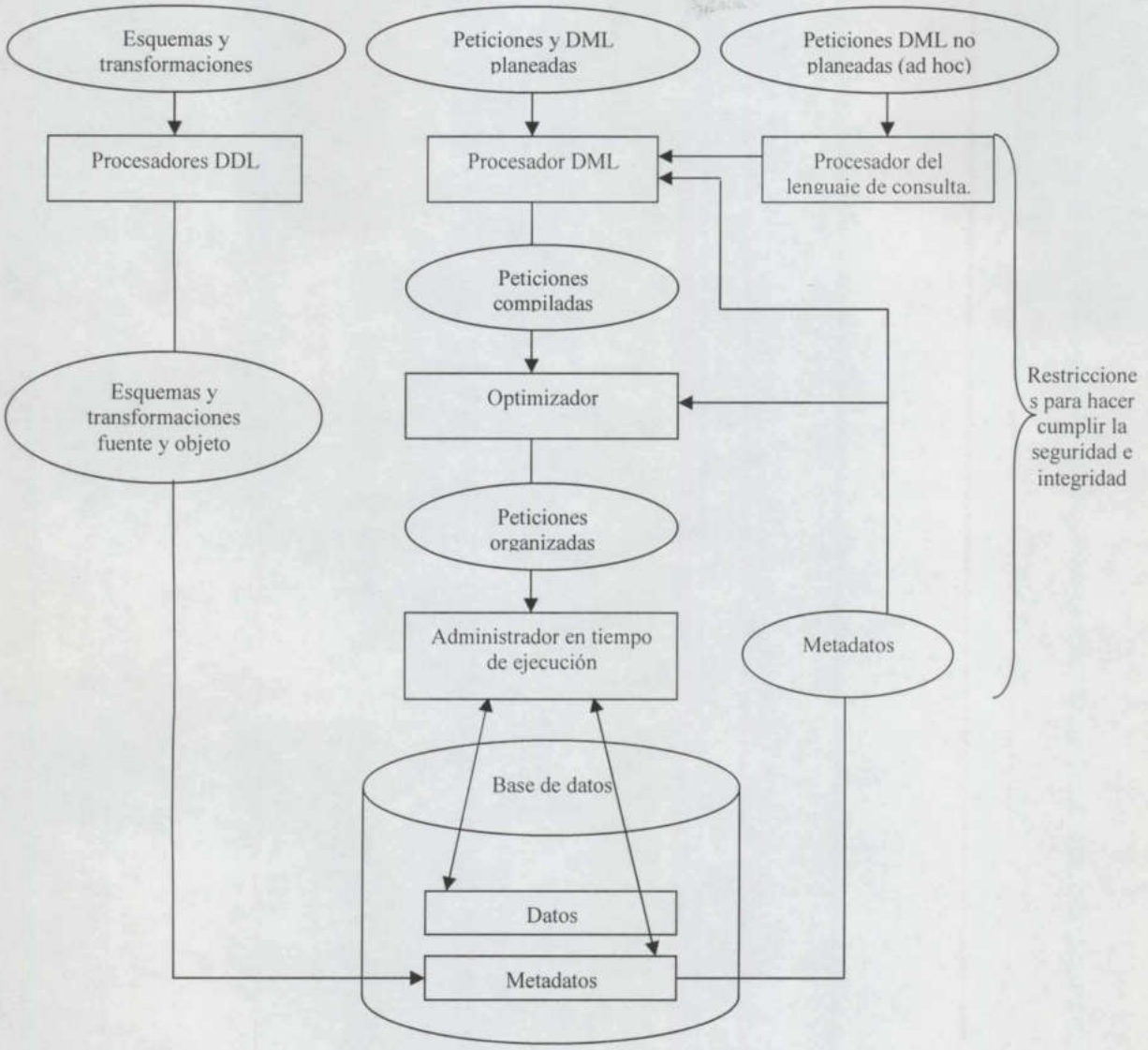


Figura 2.4 Funciones y componentes principales del DBMS

- Diccionario de datos.

El DBMS debe proporcionar una función de diccionario de datos. Este diccionario puede ser visto como una base de datos por derecho propio (aunque una base de datos del sistema mas que como una base de datos del usuario). El diccionario contiene "datos acerca de los datos" (en ocasiones llamados metadatos o descriptores); es decir, definiciones de otros objetos del sistema, en lugar de simples "datos en bruto". En particular, todos los diversos esquemas y transformaciones (externos, conceptuales, etcétera) y todas las diversas restricciones de seguridad y de integridad, serán almacenadas en el diccionario, tanto en forma fuente como objeto. Un diccionario extenso incluirá además mucha información adicional; mostrará por ejemplo que programas utilizan que partes de la base de datos, qué usuarios necesitan qué informes, etcétera. El diccionario podría incluso – y de hecho, debería – estar integrado dentro de la base de datos que define, e incluir por lo tanto su propia definición. En realidad, debe ser posible consultar el diccionario del mismo modo que cualquier otra base de datos, de manera que por ejemplo, sea posible saber qué programas o usuarios se podrían ver afectados por un cambio propuesto al sistema.

Nota: Aquí estamos tocando un área en la que hay mucha confusión de terminología. Algunas personas podrían referirse a lo que nosotros llamamos diccionario como directorio o catálogo – con la implicación de que los directorios y catálogos son, en cierta forma, inferiores a un verdadero diccionario – y podrían reservar el término diccionario para hacer referencia a una clase específica (importante) de herramienta de desarrollo de aplicaciones. Otros términos que también son utilizados, a veces, para hacer referencia a este último tipo de objeto son depósito de datos y enciclopedia de datos.

- Rendimiento

Sobra decir que el DBMS debe realizar todas las tareas antes identificadas de la manera mas eficiente posible.

Podemos resumir todo lo anterior diciendo que la finalidad general del DBMS consiste en proporcionar una interfaz de usuario para el sistema de base de datos. Podemos definir la interfaz de usuario como un límite en el sistema debajo del cual todo es invisible para el

usuario. Por lo tanto, por definición la interfaz de usuario se encuentra en el nivel externo. Sin embargo, existen algunas situaciones en las que es poco probable que la vista externa difiera de manera significativa de la parte relevante de la vista conceptual subyacente, por lo menos en los productos comerciales SQL actuales.

Concluimos esta sección con una breve comparación entre los sistemas de administración de base de datos y los sistemas de administración de archivos (administradores de archivos o servidores de archivos, para abreviar). En esencia, el administrador de archivos es el componente del sistema operativo subyacente que administra los archivos almacenados; por lo tanto, hablando en términos generales, está "más cerca del disco" de lo que lo está el DBMS. (De hecho, el DBMS es generalmente construido sobre algún tipo de administrador de archivos.) Por lo tanto, el usuario de un sistema de administración de archivos podrá crear y destruir archivos almacenados y realizar operaciones sencillas de recuperación y actualización sobre registros almacenados en dichos archivos. Sin embargo, en contraste con el DBMS:

- Los administradores de archivos no están al tanto de la estructura interna de los registros almacenados, de ahí que no puedan manejar peticiones que se basen en el conocimiento de esa estructura.
- Por lo regular ofrecen poco o ningún soporte para las restricciones de seguridad y de integridad.
- Por lo regular ofrecen poco o ningún soporte para los controles de recuperación y concurrencia.
- No hay un concepto real de diccionario de datos en el nivel del administrador de archivos.
- Proporcionan mucho menos independencia de datos que el DBMS.
- Por lo regular los archivos no están "integrados" o "compartidos" en el mismo sentido que en una base de datos (normalmente son exclusivos de cirto usuario o aplicación en particular).

2.9 EL ADMINISTRADOR DE COMUNICACIONES DE DATOS.

En esta sección consideraremos brevemente el tema de las comunicaciones de datos. Las peticiones emitidas a la base de datos por parte de un usuario final, en realidad son transmitidas desde la estación de trabajo del usuario – la cual podría ser físicamente remota con respecto al propio sistema de base de datos – hacia cierta aplicación en línea (integrada u otra) y de ahí hacia el DBMS en la forma de mensajes de comunicación. De forma similar, las respuestas que regresan del DBMS y la aplicación en línea hacia la estación de trabajo del usuario, también son transmitidas en forma de dichos mensajes. Todas estas transmisiones de mensajes se llevan a cabo bajo el control de otro componente de software, el administrador de comunicaciones de datos (administrados CD).

Este administrador no forma parte del DBMS, sino que es un sistema autónomo por derecho propio. Sin embargo, puesto que es claramente necesario que trabaje en armonía con el DBMS, en ocasiones se les considera como socios igualitarios en una empresa de más alto nivel denominada sistema de base de datos/comunicaciones de datos (sistema BD/CD); en el cual el DBMS se ocupa de la base de datos y el administrador CD maneja todos los mensajes hacia y desde el DBMS o más precisamente hacia y desde las aplicaciones que el DBMS utiliza. No obstante, en este libro tendremos relativamente poco que decir con respecto al manejo de mensajes como tal (por sí solo, es un tema muy extenso).

2.10 ARQUITECTURA CLIENTE – SERVIDOR

Hasta ahora, en este capítulo hemos venido explicando los sistemas de base de datos desde el punto de vista de la así llamada arquitectura ANSI/SPARC. En la figura 2.3 en particular sentamos una imagen simplificada de esta arquitectura. En esta sección estudiaremos los sistemas de base de datos desde una perspectiva ligeramente diferente. Desde luego, la finalidad principal de dichos sistemas es apoyar el desarrollo y la ejecución de las aplicaciones de bases de datos. Por lo tanto, desde un punto de vista más elevado, un sistema de base de datos puede ser visto como un sistema que tiene una estructura muy

sencilla de dos partes, las cuales consisten en un servidor (también denominado parte dorsal o servicios de fondo) y un conjunto de clientes (también llamados partes frontales, aplicaciones para el usuario o interfaces). Consulte la figura 2.5. Explicación:

- El servidor es precisamente el propio DBMS. Soporta todas las funciones básicas del]DBMS expuestas en la sección 2.8: definición de datos, manipulación de datos, seguridad e integridad de los datos, etcétera. En particular, proporciona todo el soporte de los niveles externo, conceptual e interno explicados en las secciones 2.3 a 2.6. Por lo tanto, en este contexto, "servidor" es solo el nombre del DBMS.
- Los clientes son las diversas aplicaciones que se ejecutan sobre el DBMS, tanto aplicaciones escritas por el usuario como aplicaciones integradas (es decir, aplicaciones proporcionadas por el fabricante del DBMS o por alguna otra compañía). Por supuesto, en lo que concierne al servidor, no hay diferencia entre las aplicaciones escritas por el usuario y las integradas al servidor, no hay diferencia entre las aplicaciones escritas por el usuario y las integradas; todas usan la misma interfaz con el servidor, como la interfaz de nivel externo expuesta en la sección 2.3.

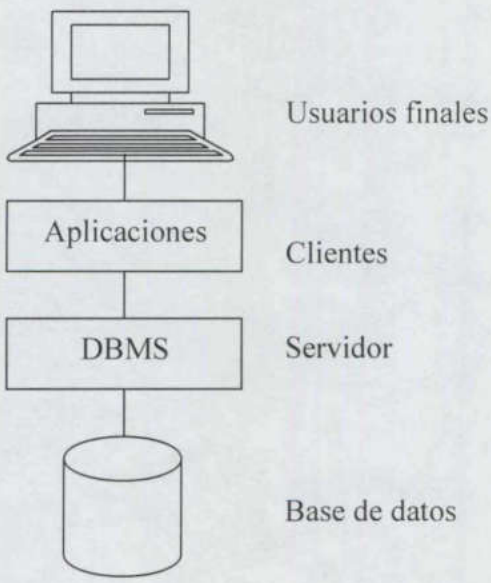


Figura 2.5 Arquitectura cliente-servidor.

Nota: Ciertas aplicaciones especiales de "utilería" podrían constituir una excepción a lo anterior, ya que en ocasiones podrían necesitar operar directamente en el nivel interno del sistema (como mencione en la sección 2.5). estas utilerías son consideradas mas bien como componentes integrales del DBMS, en vez de aplicaciones en el sentido usual. Se explican con mas detalle en la siguiente sección.

Profundizaremos un poco sobre la cuestion de aplicaciones escritas por el usuario en comparación con las aplicaciones proporcionadas por el fabricante:

- Las aplicaciones escritas por el usuario son en esencia programas de aplicación comunes, escritos por lo regular ya sea en un lenguaje convencional de tercera generación (como C o COBOL) o en algún lenguaje de tipo propietario de cuarta generación; aunque en ambos casos es necesario acoplar de alguna manera el lenguaje con un sublenguaje de datos apropiado.

- Las aplicaciones proporcionadas por el fabricante (a menudo llamadas herramientas) son aplicaciones cuya finalidad básica es auxiliar en la creación y ejecución de otras aplicaciones. Las aplicaciones creadas son aplicaciones confeccionadas para alguna tarea específica (podrían no parecer aplicaciones en el sentido convencional; de hecho, la idea general de las herramientas es permitir a los usuarios, en especial a los usuarios finales, la creación de aplicaciones sin tener que escribir programas en un lenguaje de programación convencional). Por ejemplo, una herramienta proporcionada por el fabricante sería un generador de informes, cuyo propósito es permitir a los usuarios obtener del sistema informes con cierto formato. Toda petición de informe puede verse como un pequeño programa de aplicación, escrito en lenguaje generador de informes de muy alto nivel (y de finalidad especial).

Las herramientas proporcionadas por el fabricante pueden dividirse en varias clases relativamente distintas:

a) Procesadores de lenguaje de consulta;

b) Generadores de informes;
c) Subsistemas de gráficos comerciales;
d) Hojas de cálculo;
e) Procesadores de lenguaje natural;
f) Paquetes estadísticos;
g) Herramientas de administración de copias o "extracción de datos";
h) Generadores de aplicaciones (incluyen procesadores 4GL);
i) Otras herramientas de desarrollo de aplicaciones, incluyendo productos de ingeniería de software asistida por computadora (CASE);

Y muchos otros. Los detalles de dichas herramientas exceden el alcance de este libro; sin embargo, señalaremos que (como dijimos antes) debido a que la idea general de un sistema de base de datos es apoyar la creación y ejecución de aplicaciones, la calidad de las herramientas disponibles es, o debería ser, un factor primordial en "la decisión de la base de datos" (es decir, en el proceso de seleccionar un nuevo producto de base de datos). En otras palabras, el DBMS como tal no es el único factor a tomar en consideración, ni tampoco es necesariamente el factor más importante.

Cerramos esta sección con una referencia anticipada. Puesto que el sistema en su conjunto puede ser dividido claramente en dos partes (clientes y servidores), surge la posibilidad de operar los dos en máquinas diferentes. En otras palabras, existe el potencial para el procesamiento distribuido. Procesamiento distribuido significa que es posible conectar distintas máquinas en cierto tipo de red de comunicaciones, de tal manera que una sola tarea de procesamiento de datos pueda dividirse entre varias máquinas de la red. De hecho, esta posibilidad es tan atractiva – por diversas razones, principalmente económicas – que el término cliente-servidor ha llegado a aplicarse casi exclusivamente al caso en el que el cliente y el servidor están, en efecto, en máquinas distintas.

2.11 UTILERIAS

Las utilerías son programas diseñados para ayudar al DBA en sus numerosas tareas administrativas. Como mencione en la sección anterior, algunos programas de utilería operan en el nivel externo del sistema y en realidad no son más que aplicaciones de propósito especial; algunas podrían incluso no ser proporcionadas por el fabricante del DBMS, sino por alguna otra compañía. Sin embargo, otras utilerías operan directamente en el nivel interno (en otras palabras, son realmente parte del servidor), de ahí que deban ser proporcionadas al fabricante del DBMS.

Aquí hay algunos ejemplos del tipo de utilerías que comúnmente son requeridas en la práctica:

- Rutinas de carga, para crear la versión inicial de la base de datos a partir de uno o más archivos del sistema operativo.
- Rutinas de descarga/recarga, para descargar la base de datos (o partes de ella), para respaldar los datos almacenado y para recargar datos desde dichas copias de respaldo (por supuesto, la "rutina de recarga" es básicamente idéntica a la rutina de carga recién mencionada).
- Rutinas de reorganización, para reordenar los datos en la base de datos almacenada, por distintas razones que normalmente tienen que ver con el desempeño; por ejemplo, agrupar datos en el disco de alguna forma en particular o recuperar espacio ocupado por datos que se volvieron obsoletos;
- Rutinas estadísticas, para calcular diversas estadísticas de desempeño, como el tamaño de los archivos, las distribuciones de valores, los contadores de operaciones de E/S, etcétera;
- Rutinas de análisis, para analizar las estadísticas arriba mencionadas;

Y así sucesivamente. Por supuesto, esta lista representa solo una pequeña muestra del rango de funciones que ofrecen generalmente las utilerías; existe una gran cantidad de otras posibilidades.

2.12 EL PROCESAMIENTO DISTRIBUIDO

Para repetir lo dicho en la sección 2.10, el término "procesamiento distribuido" significa que distintas máquinas pueden conectarse en una red de comunicaciones como internet, de tal manera que una sola tarea de procesamiento de datos pueda extenderse a varias máquinas de la red. (en ocasiones, también se usa el término "procesamiento paralelo" básicamente con el mismo significado, con excepción de que las máquinas distintas tienden a estar físicamente muy cerca en un sistema "paralelo", lo que no es necesario en un sistema distribuido; en último caso, por ejemplo, podrían estar geográficamente dispersas). La comunicación entre las diversas máquinas es manejada mediante algún tipo de software de administración de redes; tal vez una extensión del administrador CD que explicamos en la sección 2.9 y más probablemente un componente de software independiente.

El procesamiento distribuido presenta muchos niveles o variedades posibles. Para repetir lo dicho en la sección 2.10, un caso sencillo comprendería la operación de los servicios dorsales del DBMS (el servidor) en una máquina y las aplicaciones de usuario 8los clientes) en otra. Consulte la figura 2.6.

Aunque estrictamente hablando el término "cliente-servidor" es puramente arquitectónico. Se ha convertido casi en sinónimo de la organización ilustrada en la figura 2.6, en la que un cliente y un servidor se ejecutan en máquinas diferentes. De hecho, hay muchos argumentos a favor de dicho esquema:

- El primero es básicamente el simple argumento de procesamiento paralelo normal: es decir, ahora se están aplicando muchas unidades de procesamiento para las tareas en conjunto, mientras que el procesamiento del servidor (base de datos) y del cliente (aplicación) se están haciendo en paralelo. De ahí que el tiempo de respuesta y la velocidad real de transporte tengan mejorías.

- Además, la máquina servidor podría ser una máquina construida a la medida y adaptada a la función del DBMS (una máquina de base de datos) y podría, por lo tanto, proporcionar un mejor desempeño del DBMS.

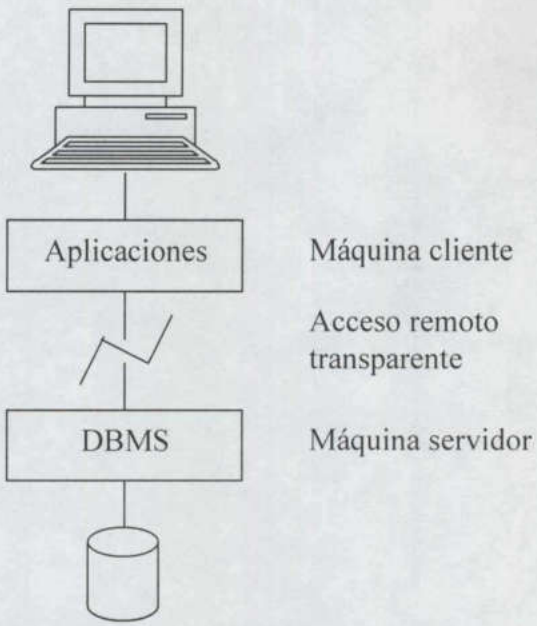


Figura 2.6 Cliente y servidor operando en máquinas diferentes.

- En forma similar, la máquina cliente podría ser una estación de trabajo personal adaptada a las necesidades del usuario final y por lo tanto, capaz de proporcionar mejores interfaces, una alta disponibilidad, respuestas más rápidas y en general una mejor facilidad de uso para el usuario.

- Varias máquinas cliente distintas podrían ser (de hecho serían) capaces de acceder a la misma máquina servidor. Por lo tanto, una sola base de datos podría ser compartida entre varios sistemas cliente distintos (vea figura 2.7).

Además de los argumentos anteriores, esta el punto de que ejecutar los clientes y el servidor en máquinas separadas coincide con la forma en que operan en realidad las empresas.

Es muy común que una sola empresa – por ejemplo, un banco – opere muchas computadoras, de tal modo que los datos de una parte de la empresa estén almacenados en otra computadora.

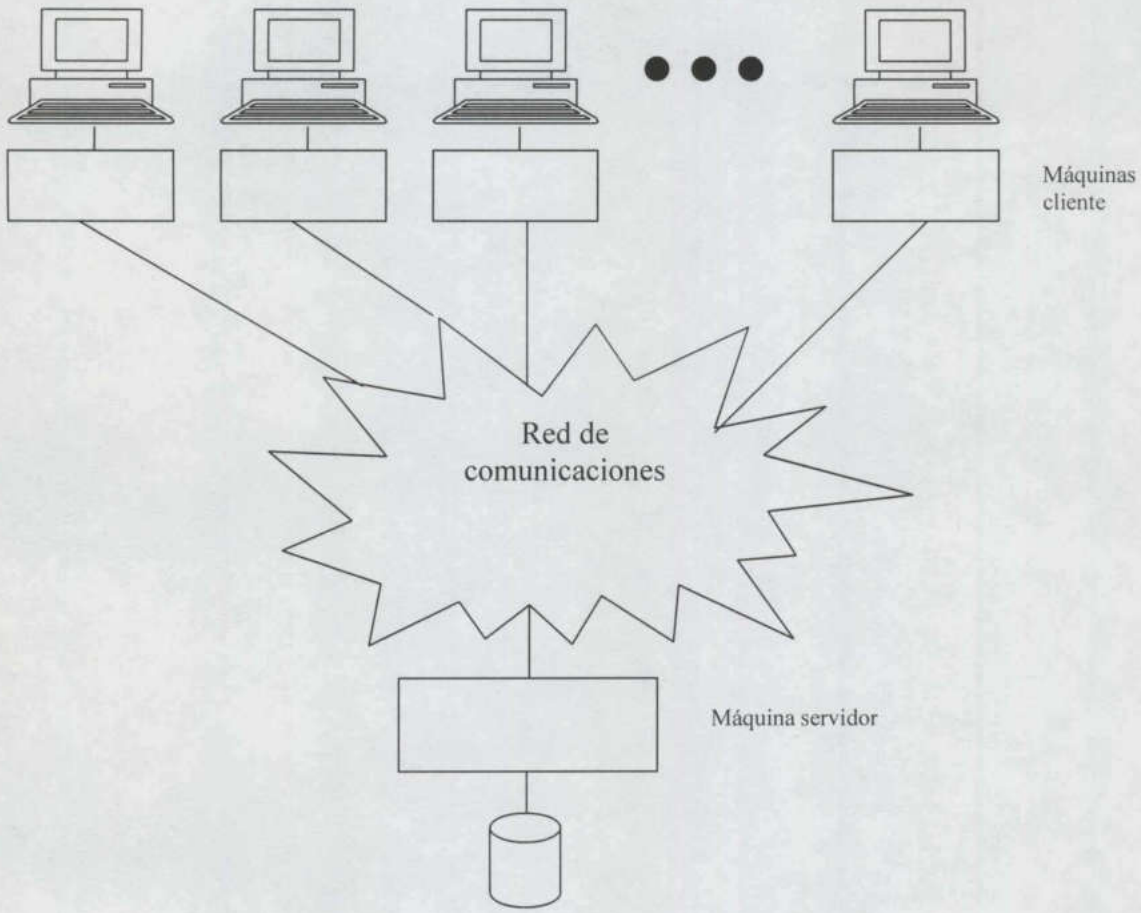


Figura 2.7 Una máquina servidor, varias máquinas cliente.

También es bastante común que los usuarios de una computadora necesiten acceso por lo menos ocasional a los datos almacenados en otra computadora. Siguiendo con el ejemplo del banco, es muy probable que los usuarios de una sucursal necesiten acceder ocasionalmente a los datos almacenados en otra. Observe, por lo tanto, que las máquinas cliente podrían tener almacenados datos propios y que la máquina servidor podría tener sus propias aplicaciones. Por lo tanto, es común que cada máquina actúe como servidor para ciertos usuarios y como cliente para otros (vea la figura 2.8); en otras palabras, cada

máquina soportará un sistema de base de datos completo, en el sentido al que se refieren las secciones anteriores de este capítulo.

La idea final es que una sola máquina cliente podría ser capaz de acceder a varias máquinas servidor diferentes (lo contrario al caso ilustrado en la figura 2.7).

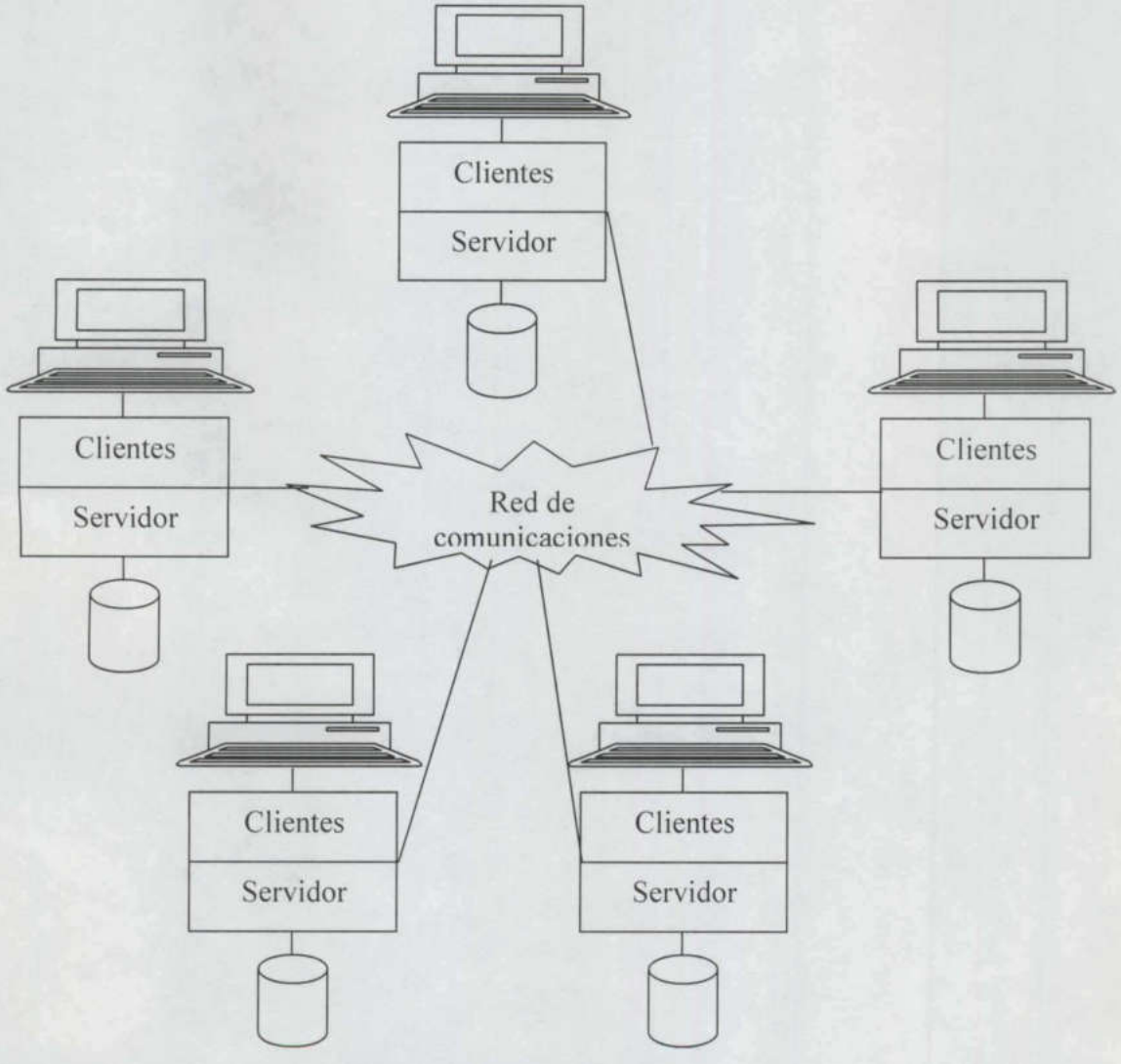


Figura 2.8 Cada máquina opera como clientes y como servidor

Esta posibilidad es conveniente ya que, como mencioné antes, las empresas operan por lo regular de tal manera que la totalidad de sus datos no están almacenados en una sola máquina, sino más bien están esparcidos a través de muchas máquinas distintas, y las

aplicaciones necesitarán a veces tener la posibilidad de acceder a los datos de más de una máquina. Básicamente, este acceso puede ser proporcionado en dos formas:

- Una máquina cliente dada podría ser capaz de acceder a cualquier cantidad de servidores, pero solo uno a la vez (es decir, cada petición individual de base de datos debe ser dirigida a un solo servidor). En un sistema así, no es posible combinar los datos de dos o más servidores con una sola petición. Además, el usuario de dicho sistema debe saber que máquina en particular contiene qué piezas de datos.

- El cliente podría ser capaz de acceder a varios servidores en forma simultánea (es decir, una sola petición de base de datos podría combinar datos de varios servidores). En este caso, los servidores ven al cliente – desde el punto de vista lógico – como si en realidad fuera un solo servidor y el usuario no tiene que saber qué máquinas contienen qué piezas de datos.

Este último caso constituye lo que por lo regular se denomina un sistema de base de datos distribuida. La base de datos distribuida es en sí un tema extenso. Llevada a su conclusión lógica, el soporte total a la base de datos distribuida implica que una sola aplicación debe ser capaz de operar de manera transparente sobre los datos que están dispersos a través de una variedad de base de datos diferentes, las cuales son administradas por una variedad de DBMSs distintos, operan en varias máquinas distintas, con manejadas por varios sistemas operativos diferentes y están conectadas por una variedad de redes de comunicación distintas, aquí "de manera transparente" significa que la aplicación opera desde un punto de vista lógico como si los datos fueran manejador por un solo DBMs y en una sola máquina. ¡una posibilidad como esta podría parecer muy difícil de lograr!; pero es bastante conveniente desde una perspectiva práctica, y los fabricantes están haciendo un gran esfuerzo para hacer realidad dichos sistemas.

3. ADMINISTRACIÓN DE UNA BASE DE DATOS LOCAL

3.1.-CONEXIÓN DE UNA RED LOCAL.

Este capítulo trata fundamentalmente sobre los aspectos *"lógicos"* de la arquitectura de red. Lo que puede o no puede hacer una red está generalmente determinado por los protocolos que dicha red soporta y la calidad de sus implementaciones, más que por la tecnología concreta de red usada, como Ethernet, Token Ring, etc. Además, en la práctica, la elección de la tecnología de red está basada en decisiones puramente pragmáticas: qué tipo de red soporta el tipo de ordenadores que queremos conectar, las distancias entre los equipos, las características del cableado, etc. Por regla general, se suele usar Ethernet para sistemas de media escala, Ethernet o una red basada en el cableado de par trenzado para pequeñas redes, o redes de alta velocidad (típicamente Token Ring) para la red principal de un campus y para redes de super ordenadores, que ejecutan aplicaciones de altas prestaciones.

Por tanto, vamos a asumir que hemos llegado a conectar *"fisicamente"* unas redes individuales, del tipo Ethernet o Token Ring. Ahora nos enfrentamos a los siguientes problemas interrelacionados:

* configurar el software necesario;

* conectar las distintas Redes Ethernet, Token Ring, etc, para formar una única red de forma coherente;

* conectar las redes al mundo exterior, o sea, Internet.

Las anteriores decisiones requieren un pequeño análisis. De hecho, la mayoría de las redes necesitan una *"arquitectura"* , que determina la manera en que se asignan las direcciones, cómo se hace el enrutado y otras elecciones, sobre cómo los ordenadores interaccionan con la red. Estas decisiones deben hacerse para la red en su conjunto, preferiblemente cuando se esta procediendo a su instalación inicial.

3.1.1. TERMINOLOGIA.

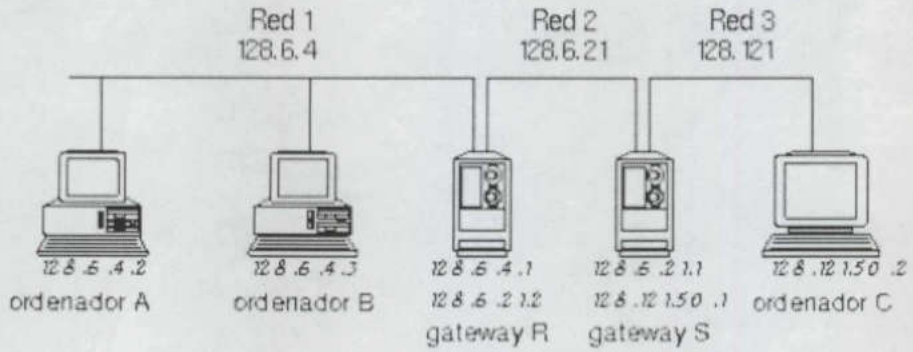
Vamos a usar el término *"IP"* para referirnos a las redes diseñadas para trabajar con TCP/IP. IP es el protocolo a nivel de red de la familia de protocolos TCP/IP, usados en Internet. Es una práctica común usar el término *"IP"* cuando nos referimos a direcciones, enrutamiento y otros elementos a nivel de red. La distinción muchas veces no es lo suficientemente clara. Así que, en la práctica, los términos Internet TCP/IP e IP pueden parecer incluso intercambiables.

Los términos *"paquete"* y *"datagrama"* también suelen parecer intercambiables. Conceptualmente, un *"paquete"* es la unidad física de más bajo nivel, mientras que *"datagrama"* se refiere a la unidad de datos a nivel IP. Sin embargo, en la mayoría de las redes no se pueden distinguir porque coinciden, así que la gente suele usar los dos términos indistintamente.

Otro término "conflictivo" es el de pasarela (*"gateway"*) y enrutador (*"router"*). Pasarela es el término original usado en Internet. Sin embargo, la comunidad OSI empezó a usar esta palabra con un significado distinto, así que la gente empezó a usar enrutador para evitar dicha ambigüedad. Nosotros, no obstante, seguiremos usando el término gateway.

3.2.- ASIGNACION DE DIRECCIONES Y ENRUTAMIENTO.

Muchas de las decisiones que se necesitan para la configuración de una red IP depende del enrutamiento. En general, un datagrama IP pasa a través de numerosas redes mientras se desplaza entre el origen y el destino. Veamos un ejemplo típico:



Este gráfico muestra tres ordenadores, 2 gateways y tres redes. Las redes pueden ser Ethernet, Token Ring o de cualquier otro tipo. La red 2 podría ser una línea punto a punto que conecta los gateways R y S.

El ordenador A puede enviar datagramas al B directamente, usando la red 1. Sin embargo, no puede llegar al ordenador C directamente, puesto que no están en la misma red. Hay varias maneras de conectar redes. En el gráfico asumimos el uso de gateways (más adelante veremos otras alternativas). En este caso, los datagramas que van desde A a C deben ser enviados a través del gateway R, red 2 y gateway S. Todos los ordenadores que usan TCP/IP necesitan que se les suministre la información y algoritmos apropiados para que puedan saber cuándo un datagrama debe ser enviado a través de un gateway, y elegir el gateway apropiado.

El enrutado está íntimamente relacionado con la asignación de direcciones. Podemos apreciar que la dirección de cada ordenador comienza con el número de la red a la que pertenece. Por tanto, 128.6.4.2 y 128.6.4.3 se encuentran en la red 128.6.4. Luego los gateways, cuyo trabajo es conectar dos redes, tienen una dirección de ambas redes. Por ejemplo, el gateway R conecta la red 128.6.4 y 128.6.21. Su conexión a la red 128.6.4 tiene la dirección 128.6.4.1. Su conexión a la red 128.6.21 tiene la dirección 128.6.21.2.

Debido a esta relación entre direcciones y redes, las decisiones de enrutado deben basarse estrictamente en el número de red de destino. La información de enrutamiento del ordenador A tendrá el siguiente aspecto:

red gateway métrica

128.6.4 - 0

128.6.21 128.6.4.1 1

128.121 128.6.4.1 2

En esta tabla, el ordenador A puede enviar datagramas a los ordenadores de la red 128.6.4 directamente, y para los datagramas a los ordenadores de las redes 128.6.21 y 128.121 es necesario usar el gateway R. La *"métrica"* será usada por algún tipo de algoritmo de enrutamiento, como medida de la lejanía del destinatario. En nuestro caso, la métrica simplemente indica cuantos diagramas tiene que atravesar para llegar a su destino (conocida como *"cuenta de saltos"*).

Cuando el ordenador A está listo para enviar un datagrama se examina la dirección del destinatario. Comparamos el inicio de dicha dirección de red con las direcciones de la tabla de enrutamiento. Las distintas entradas de la tabla indican si el datagrama debe ser enviado directamente, o a un gateway.

Un gateway consiste simplemente en un ordenador conectado a dos redes diferentes, y está habilitado para enviar datagramas entre ellos. En muchos casos es más eficiente usar un equipo especialmente diseñado para desempeñar el papel de gateway. Sin embargo, es perfectamente posible usar un ordenador, siempre y cuando tenga más de un interfaz de red y un software capaz de enviar datagramas.

Un gateway tiene varias direcciones, una para cada red a la que esté conectado. Aquí encontramos una diferencia entre IP y otros protocolos de red: cada interface de un ordenador tiene una dirección. Con otros protocolos, cada ordenador tiene una única dirección, aplicable a todos sus interfaces. Un gateway entre las redes 128.6.4 y 128.6.21

tendrá una dirección que comience por 128.6.4 (por ejemplo, 128.6.4.1). Esta dirección se refiere a su conexión a la red 128.6.4. También tendrá una dirección que comience con 128.6.21 (por ejemplo, 128.6.21.2). Esta se refiere a su conexión a la red 128.6.21.

El término *"red"* generalmente se suele identificar a dispositivos del tipo Ethernet, en la cual varias máquinas están conectadas. Sin embargo, también se aplica a líneas punto a punto. En el gráfico anterior, las redes 1 y 3 podrían estar en ciudades distintas; la red 2 podría ser una línea serie, un enlace satélite, u otro tipo de conexión punto a punto. Una línea punto a punto es tratada como una red que consta sólo de dos ordenadores. Como cualquier otra red, una línea punto a punto tiene una dirección de red (en este caso, 128.6.21). Los sistemas conectados por la línea (*gateways R and S*) tienen direcciones en dicha red (en este caso, 128.6.21.1 y 128.6.21.2).

Es posible diseñar software que no necesite distintos números de red para cada línea punto a punto. En este caso, el interface entre el gateway y la línea punto a punto no tiene una dirección. Esta solución es apropiada cuando la red es tan grande que peligra el hecho de que nos quedemos sin direcciones. Sin embargo, tales *"interfaces anónimas"* pueden dificultar bastante el manejo de la red. Puesto que no tienen dirección, el software de red no tiene manera de referirse a dicho interface, y, por tanto, no es posible obtener información sobre el flujo y los errores de la interface.

3.3.- ELIGIENDO UNA ESTRUCTURA DE DIRECCIONES.

Antes de comenzar a montar una estructura de IP, necesitamos uno o más números de red oficiales. Una dirección IP tiene un aspecto como el siguiente: 128.6.4.3. Esta dirección sólo podrá ser usada por un ordenador de la Universidad de Marx. La primera parte de dicha dirección, 128.6, es un número de red asignado a dicha Universidad por una autoridad central. Por tanto, antes de asignarse direcciones a nuestros ordenadores, deberemos obtener una dirección oficial de red. Sin embargo, alguna gente configura sus redes usando, o bien una dirección aleatoria o usando una dirección genérica suministrada por defecto en el equipo. Esta forma de trabajar podría funcionar en pequeñas redes, pero seguramente no lo hará en una mayor. Además, es posible que quisiéramos conectar nuestra red con la red de otra organización. Incluso si nuestra organización tuviese un gran control

de seguridad, es posible que tuviéramos un ordenador dedicado a la investigación que estuviese conectado a una universidad u otra organización investigadora. Esta universidad o entidad estaría seguramente conectada a una red de nivel nacional. Tan pronto como uno de nuestros datagramas salga de nuestra red local va a provocar un estado de confusión en la organización con la que nos comuniquemos, porque la dirección que aparece en nuestros datagramas está probablemente asignada oficialmente a alguien distinto.

La solución es simple: obtener una dirección propia desde el principio. Además, no cuesta nada.

La decisión más importante que tenemos que hacer para configurar una red es, sin lugar a dudas, cómo asignar las direcciones IP a los ordenadores. Esta elección debe de hacerse desde el punto de vista de cómo nuestra red puede crecer. Si no se hiciese así, es casi seguro que tendremos que cambiar las direcciones en un futuro. Y cuando tengamos varios cientos de ordenadores, cambiar tantas direcciones es casi imposible.

Las direcciones son muy importantes porque los datagramas IP son enrutados en base a dicha dirección. Por ejemplo, las direcciones de la Universidad Groucho Marx tienen una estructura de dos niveles. Una dirección típica puede ser 128.6.4.3. La dirección 128.6 es la asignada a dicha Universidad. Visto desde el exterior, 128.6 es una simple red. Cualquier datagrama enviado desde el exterior, que comience por 128.6, se dirigirá al gateway más cercano de la Universidad Groucho Marx. Sin embargo, dentro de Groucho Marx dividimos el espacio de direcciones en *"subredes"* . Usamos los siguientes 8 bits de dirección para indicar a qué subred pertenece el ordenador. Así, 128.6.4.3 pertenece a la subred 128.6.4. Generalmente, las subredes se corresponden con redes *"físicas"* o reales, por ejemplo una red Ethernet; sin embargo, veremos algunas excepciones más adelante. Los sistemas dentro de Groucho Marx, a diferencia de los de fuera, contienen información sobre la estructura de subredes de Groucho Marx. Así, una vez que un datagrama para 128.6.4.3 llega a Groucho Marx, la red de Groucho Marx lo enrutará hacia la Ethernet, Token Ring o cualquier otro tipo de red del departamento que tiene asignado la subred 128.6.4.

Cuando queremos configurar una red, hay varias decisiones de direccionamiento que debemos afrontar:

* ¿Dividimos nuestro espacio de direcciones?

* Si lo hacemos, ¿usamos subredes o direcciones de clase C?

* ¿Cómo debe ser de grande el espacio de direcciones que necesitamos?

3.3.1. SUBREDES Y MULTIPLES NUMEROS DE RED.

Supongamos que estamos convencidos de que es una buena idea imponer alguna estructura en nuestras direcciones. La siguiente cuestión es cuál es la más adecuada. Hay dos enfoques básicos: subredes y múltiples números de red.

Los estándares de Internet especifican el formato de las direcciones. Para las direcciones que comienzan entre 128 y 191 (las más usadas actualmente), los dos primeros octetos forman el número de red; por ejemplo, en 140.3.50.1, 140.3 es el número de red. Los números de red están asignados a una organización particular. ¿Qué hacemos con los dos siguientes octetos que le siguen?. Podríamos optar por hacer al siguiente octeto un número de subred, u otro esquema completamente distinto. Los gateways dentro de nuestra organización deben configurarse para conocer qué esquema de división de redes estamos usando. Sin embargo, fuera de la organización nadie sabrá si 140.3.50 es una subred y 140.3.51 es otra; simplemente, fuera se sabe que 140.3 es una organización. Desafortunadamente, esta habilidad de añadir una estructura adicional a las direcciones, mediante el uso de subredes, no estaba presente en las especificaciones originales y, por tanto, un software antiguo sería incapaz de trabajar con subredes. Si una parte importante del software que hemos de usar tiene este problema, entonces no podremos dividir nuestra red en subredes.

Algunas organizaciones usan un enfoque distinto. Es posible que una organización use varios números de red. En lugar de dividir un simple número de red, por ejemplo 140.3, en varias subredes, como de 140.3.1 a 140.3.10, podríamos usar 10 números distintos de red. De esta manera haríamos una asignación desde 140.3 hasta 140.12. Todo el software IP sabrá que estas direcciones se corresponden con redes distintas.

A pesar de que usando números de red distintos todo funciona correctamente dentro de la organización, hay dos serias deventajas. La primera, y menos importante, es que se malgasta un gran espacio de direcciones. Hay solamente sobre unas 16.000 posibles direcciones de clase B. No queremos malgastar diez de ellas en nuestra organización, a no ser que sea bastante grande. Esta objección es 'menos seria', porque podríamos pedir una dirección C para este propósito y hay sobre 2 millones de direcciones C.

El problema más serio para usar varias direcciones de red, en lugar de subredes, es que sobrecarga las tablas de enrutamiento en el resto de Internet. Como comentamos anteriormente, cuando dividimos nuestro número de red en subredes, esta división sólo es conocida dentro de la organización, pero no fuera. Los sistemas externos a la organización sólo necesitan una entrada en sus tablas para ser capaces de llegar. Por tanto, otras Universidades tienen entradas en sus tablas de enrutamiento para 128.6, similar al número de la red de Groucho Marx. Si usa un rango de redes en lugar de subredes, dicha división será visible en todo Internet. Si usamos los números 128.6 a 128.16, en lugar de 128.6, las otras universidades necesitarían tener una entrada para cada uno de estos números de red en sus tablas de enrutamiento. La mayoría de los expertos de TCP/IP recomiendan el uso de subredes, en lugar de múltiples redes. La única razón para considerar múltiples redes es el uso de un software que no puede manejar subredes. Esto era un problema hace algunos años, pero actualmente es poco frecuente.

Una última indicación sobre subredes: Las subredes deben ser *"adyacentes"*. Esto significa que no podemos conectar la subred 128.6.4 con la subred 128.6.5 mediante otra red, como la 128.121.

3.3.2. COMO ASIGNAR LAS SUBREDES O LOS NUMEROS DE RED.

Ahora, una vez decidido si vamos a usar subredes o múltiples números de red, tenemos que asignarlos. Normalmente es bastante fácil. Cada red física, ya sea Ethernet o Token Ring, se le asigna un número distinto de subred. Sin embargo, existen otras opciones.

En algunos casos, puede que tenga sentido asignar varios números de subred a una única red física.

También hemos de elegir una *"máscara de subred"*, que será usada por el software del sistema para separar la parte de subred del resto de la dirección. Hasta ahora hemos asumido que los dos primeros octetos son el número de red y el siguiente es el número de subred. Para las direcciones de clase B, el estándar especifica que los dos primeros octetos pertenecen al número de red. Y, por otro lado, tenemos libertad para establecer el límite del número de subred y el resto de la dirección. Es bastante usual utilizar un octeto de número de subred, pero no es la única posibilidad. Veamos de nuevo esta dirección de clase B, 128.6.4.3. Es fácil deducir que, si el tercer octeto es usado como número de subred, entonces habrá 256 posibles subredes y, en cada subred, habrá 256 posibles direcciones. (En realidad es más acertado decir que disponemos de 254, ya que no es buena idea usar 0 ó 255 como números de subred o dirección). Supongamos que sabemos que nunca vamos a tener más de 128 ordenadores por subred, pero es probable que necesitemos más de 256 subredes (por ejemplo, un campus con una gran cantidad de pequeños edificios). En ese caso, podríamos establecer 9 bits como número de red, dejando 7 bits para el direccionamiento de cada subred. Esta decisión queda plasmada en una máscara de bits, usando unos para los bits usados por los números de red y de subred y ceros para los bits usados para el direccionamiento individual. La máscara de red más común es 255.255.255.0. Si elegimos 9 bits para el número de subredes y 7 para las direcciones, la máscara de subred sería 255.255.255.128.

Generalmente, es posible especificar la máscara de subred como parte de la configuración del software IP. Los protocolos IP también permiten a los ordenadores que envíen un mensaje preguntando cuál es su máscara de subred. Si nuestra red soporta el envío de estos mensajes, y hay, al menos, un ordenador o gateway de la red que conoce dicha máscara de subred, posiblemente será innecesario especificarlo en cada uno de los restantes ordenadores. Pero esta posibilidad puede traer muchos problemas. En caso de que nuestra implementación TCP/IP diera una máscara de subred errónea, se causaría una mala configuración en toda la red. Por lo tanto, es más seguro poner cada máscara de subred explícitamente en cada sistema.

3.3.3. TRABAJAR CON MULTIPLES SUBREDES *"VIRTUALES"* EN UNA RED.

La mayoría del software está desarrollado bajo el supuesto de que cada red local tiene el mismo número de subred. Cuando existe un flujo hacia una máquina con un distinto número de subred, el software espera encontrar un gateway que pueda dirigirlo hacia esa subred. Veamos detalladamente qué ocurre en este caso. Supongamos que tenemos las subredes 128.6.19 y 128.6.20 en la misma Ethernet. Consideremos las cosas que ocurren desde el punto de vista de un ordenador con direcciòn 128.6.19.3. Dicho ordenador no tendrá problemas para comunicarse con las máquinas de dirección 128.6.19.x. Estas máquinas están en la misma subred, y nuestro ordenador simplemente deberá enviar los datagramas al 128.6.20.2. Puesto que esta dirección indica que está en una subred distinta, la mayoría del software esperará encontrar un gateway que haga de puente entre ambas subredes. Por supuesto, no existe un gateway entre las *"subredes"* 128.6.19 y 128.6.20, puesto que están en la misma Ethernet. De aquí se deduce que tenemos que encontrar una manera de indicarle al software que el 128.6.20 se encuentra realmente en la misma Ethernet.

La mayoría de las implementaciones TCP/IP pueden manejar más de una subred en la misma red. Por ejemplo, el Berkeley Unix nos permite hacerlo usando una ligera modificación del comando usado para definir gateways. Si, por ejemplo, queremos que para pasar de la subred 128.6.19 a la subred 128.6.4 se use el gateway con dirección 128.6.19.1, podemos usar el comando

route add 128.6.4.0 128.6.19.1 1

Esto indica que para llegar a la subred 128.6.4 el flujo debe ser enviado a través del gateway 128.6.19.1. El "1" se refiere a la *"métrica de enrutamiento"*. Si usamos la métrica "0", estamos diciendo que la subred de destino está en la misma red y, por consiguiente, no se necesita ningún gateway. En nuestro ejemplo, deberemos usar en el sistema 128.6.19.3

route add 128.6.20.0 128.6.19.1 0

La dirección usada en el lugar de 128.6.19.1 es irrelevante. La métrica "0" nos informa de que no va a usarse ningún gateway, luego no se usará dicha dirección. Sin embargo, deberá ampliarse una direción legal de la red local.

3.3.4 MÚLTIPLES SUBREDES: CONSECUENCIAS EN EL BROADCASTING.

Cuando tenemos más de una subred en una misma red física, hay que tener cuidado respecto a las direcciones de broadcasting. De acuerdo con los últimos estándares, hay dos formas distintas para que un host de la subred 128.6.20 pueda enviar un broadcast en la red local. Una es usar la dirección 128.6.20.255. La otra es usar la dirección 255.255.255.255. La dirección 128.6.20.255 dice, explícitamente, *"todos los hosts de la subred 128.6.20"*; la 255.255.255.255 expresa *"todos los hosts de mi red local"*. Normalmente, ambas tienen el mismo efecto. Pero no lo tienen cuando hay varias subredes en una red física. Si la red 128.6.19 está en la misma red, también recibirá el mensaje enviado a 255.255.255.255. Sin embargo, los hosts con números 128.6.19.x no escucharán los mensajes enviados a 128.6.20.255. El resultado es que ahí tenemos dos tipos distintos de direcciones de broadcast con dos significados distintos. Esto conlleva que debemos tener cuidado configurando el software de red, para asegurarnos de que nuestros broadcasting llegan a donde queremos que lo hagan.

3.3.5. ELIGIENDO UNA CLASE DE DIRECCION.

Cuando solicitamos un número oficial de red se nos preguntará qué clase de número de red necesitamos. Las posibles respuestas son A, B y C. La decisión elegida limitará nuestro espacio de direcciones a usar. Las direcciones de clase A ocupan un octeto; las de clase B, dos octetos, y la clase C, tres octetos. Luego, hay más direcciones de clase C que direcciones de clase A, pero las de clase C no pueden tener muchos hosts. La idea que podemos sacar de lo anterior es que debería haber pocas grandes redes, un número moderado de redes de tamaño Mediano y bastantes pequeñas redes. En la siguiente tabla observamos dicha distinción:

Clase Rango 1er. octeto red resto direcciones posibles

A 1 - 126 p q.r.s 16777214

B 128 - 191 p.q r.s 65534

C 192 - 223 p.q.r s 254

Por ejemplo, la red 10 es de la clase A y por tanto tiene direcciones entre 10.0.0.1 y 10.255.255.254. Esto signfica 254^3, que son sobre unos 16 millones de posibles direcciones (realmente, la red 10 tiene algunas direcciones con octetos a cero, así que habrá algunas direcciones posibles más). La red 192.12.88, una dirección de clase C, tendrá sus hosts entre el 192.12.88.1 y el 192.12.88.254 y, por lo tanto, habrá 254 posibles hosts.

En general, deberemos elegir la clase menor que nos proporcione suficientes direcciones capaces de direccionar nuestra red, con sus posibles futuras ampliaciones. Aquellas organizaciones que usan ordenadores en varios edificios, probablemente necesitarán una dirección de clase B, suponiendo que vamos a usar subredes. (Y si vamos a tratar con distintos números de red, deberíamos solicitar varias direcciones de clase C). Las direcciones de clase A, normalmente, sólo se usan en grandes redes públicas y algunas pocas redes de grandes corporaciones.

En la asignación de Direcciones IP, la autoridad máxima es la IANA (Internet Assigned Number Authority). A escala continental, la IANA delega grandes bloques de direcciones IP a los denominados registros regionales, de los que, de momento, existen tres en el mundo:

* El RIPE NCC (RIPE Network Coordination Center) es el registro delegado de Internet a nivel europeo y se encarga, entre otras tareas, de la asignación de bloques de direcciones IP a los proveedores de servicios Internet en Europa y su área de influencia.

* El AP-NIC lleva a cabo la tarea de asigacion de bloques de direcciones IP a los proveedores de la región del Asia-Pacífico.

* El InterNIC se encarga de la asignación de bloques de direcciones IP a los proveedores de Internet en América del Norte y, de momento, en el resto del mundo.

Las organizaciones y usuarios finales han de obtener las direcciones IP necesarias para conectarse a Internet a través de su proveedor de acceso a Internet, quien a su vez las habrá obtenido bien de su proveedor de tránsito, bien del registro regional correspondiente.

3.3.6. LINEAS IP Y MICRO GATEWAYS: DIRECCIONES ASIGNADAS DINAMICAMENTE.

En la mayoría de los casos, cada uno de los ordenadores tendrá su propia dirección IP permanente. No obstante, hay algunas situaciones donde tiene más sentido asignar direcciones dinámicamente. La mayoría de los casos que manejan líneas IP constan de gateways destinados principalmente a microcomputadoras

3.3.6.1. LÍNEAS IP.

Es posible usar IP sobre líneas telefónicas. Uno de los protocolos para hacer esto es el SLIP (*"Serial line IP"*). SLIP se usa frecuentemente en, al menos, dos circunstancias distintas:

* Como una alternativa barata a líneas punto a punto permanentes, para aquellos casos en los que no está suficientemente justificado una línea dedicada.

* Como una manera de conectar individualmente un PC a una red, cuando se encuentran localizados en edificios que no tienen Ethernets o cualquier otro tipo LAN.

Vamos a usar el término *"servidor SLIP"* para referirnos a un sistema de ordenador(es) que incluye una serie de modems, con los que otros sistemas pueden conectarse usando SLIP. Se trata de un sistema que proporciona un gateway de nuestra red para usuarios de PC, o para otras redes que se conectan usando SLIP.

Si tenemos varios PC's conectados mediante SLIP, muchas veces no es práctico usar una dirección IP propia para cada PC. Una de las razones puede ser que no haya suficientes direcciones. Para que el enrutamiento funcione correctamente, estos sistemas conectados

deben tener sus direcciones en la misma subred que el servidor SLIP. Por lo general, hay solamente del orden de 256 direcciones disponibles en cada subred. Si el número de PC's que pueden conectarse es mayor que esa cifra, no podremos asignarles su propia dirección. Si, además, tenemos servidores SLIP en más de una subred, la asignación permanente de direcciones se hace aún más complicada. Si un usuario es capaz de llamar a dos servidores, su PC necesitaría dos direcciones, una para cada subred.

Para solucionar estos problemas, la mayoría de las implementaciones SLIP asignan las direcciones dinámicamente. Cuando un PC se conecta con el servidor SLIP, el servidor busca una dirección IP que no se esté usando y se la asigna al PC. La forma más simple de manejar esto es dando a cada servidor SLIP un rango de direcciones IP que controle y pueda asignar.

Cuando usamos este esquema, el software SLIP debe comunicar al PC, de alguna manera, qué dirección debe usar. Si cada PC tiene una dirección permanente, tendríamos el problema contrario: cuando un PC se conecta con un servidor debe de haber algún método para que el PC comunique al servidor su dirección. Este problema debe ser estudiado cuidadosamente, porque en otro caso alguien podría usar la dirección de otro y tener acceso a sus ficheros.

Desafortunadamente, no hay un estándar para manejar estos problemas de direccionamiento con SLIP. Hay varias implementaciones SLIP que lo hacen, pero todavía no hay un estándar. Hasta que no se elabore éste, deberemos tener cuidado con el software SLIP. Tenemos que asegurarnos de que dicha asignación de dirección se lleva a cabo de la manera que queremos y que nuestro servidor SLIP y los PC's tienen claro la forma en que se asignan las direcciones.

Recomendamos dar direcciones permanentes a los PC's en aquellos casos en que los demás ordenadores tienen que ser capaces de conocer con qué PC están hablando. Este podría ser el caso de un PC para recibir correo privado, o cualquier otro servicio con transacciones delicadas. Y recomienda el direccionamiento dinámico cuando tenemos un gran número de PC's y las aplicaciones que utilizan para acceder a la red tienen sus propios mecanismos de seguridad.

Cuando usemos SLIP para conectar dos redes, hay que considerar tres elecciones para el manejo de direcciones (teniendo en cuenta que no todo el software SLIP puede controlar los tres apartados):

* Tratar a las conexiones SLIP como si se tratasen de líneas punto a punto que no están disponibles permanentemente. Si podemos conectar con más de un ordenador, cada par de ordenadores que se comunican tienen un número de red distinto del que ellos usarían cuando se comunican con el otro.

* Usar un software de enrutamiento que permita interfaces anónimos. En este caso, no serían necesarias las direcciones.

* Asignar direcciones dinámicamente cuando la conexión está abierta, tan pronto como el PC haya contactado.

Si hacemos sólo una o dos conexiones a otro sistema, es bastante razonable usar un número de red para cada conexión. Este método es fácil de usar y limita los errores estadísticos.

Si tenemos muchas conexiones distintas, probablemente es mejor usar interfaces anónimos. Aunque si los sistemas de enrutamiento no lo soportan, debemos usar asignación dinámica.

Al igual que SLIP, PPP *"Point to Point Protocol"* es un protocolo serie distinto utilizado para enviar datagramas a través de una conexión serie, pero mejora algunas de las carencias del anterior. El PPP permite a las partes comunicantes negociar opciones como las direcciones IP y el tamaño máximo de los datagramas al comenzar la conexión, y proporciona permisos de conexión a los clientes (autorizaciones). Para cada una de estas capacidades, el PPP tiene un protocolo concreto.

A continuación, citaremos los elementos básicos que constituyen el PPP. Esta descripcion esta muy lejos de ser completa; si quiere saber mas sobre el PPP, lea sus especificaciones en el RFC 1548, asi como en la docena de RFCs que le acompañan.

En la parte más baja del PPP está el protocolo de Control de Conexión de Datos de Alto-Nivel, abreviadamente HDLC. (En realidad, el HDLC es un protocolo mucho más general

publicado por la Organización Internacional de Estándares, ISO) que define los límites de las tramas PPP individuales, y proporciona un control de errores de 16 bit. Al contrario de lo que ocurría en las encapsulaciones SLIP más antiguas, una trama PPP es capaz de llevar paquetes de otros protocolos distintos al IP, como los IPX de Novell o Appletalk. El PPP consigue esto añadiendo a la trama básica HDLC un campo de control que identifica el tipo de paquete contenido en la trama.

El LCP, Protocolo de Control de Enlace, es utilizado en la parte más alta del HDLC para negociar las opciones concernientes a la conexión de datos, tales como la Unidad Máxima de Recepción (MRU) que establece el tamaño máximo del datagrama que una de las partes de la conexión acepta recibir.

3.3.6.2. MICRO GATEWAYS.

Es perfectamente posible que un microcomputador forme parte de una red IP. Pero hay una tendencia de que los micros utilicen distintas tecnologías de red que la de los grandes sistemas. Esto es debido a que muchos de los usuarios de micros empiezan a demandar un software de red diseñado específicamente para las necesidades de un micro, incluso para un particular tipo de micro. Muchos usuarios están interesados en usar TCP/IP sin tener que abandonar su red especial de micro, a la que están acostumbrados. Por esta razón, hay un creciente número de productos, especialmente gateways, que dan acceso a los PC's tanto a redes orientadas a micros como a TCP/IP.

En esta sección vamos a hablar del AppleTalk, de Apple, a modo de ejemplo. No obstante, existen productos similares para otros tipos de redes de micros. Hay que aclarar que el término AppleTalk se asocia a los protocolos de red de Apple, mientras que LocalTalk se asocia a una tecnología específica de par trenzado, en la que AppleTalk fue inicialmente implementada. Por tanto, el AppleTalk es análogo a los protocolos TCP/IP, mientras que LocalTalk es análogo a medio Ethernet.

Algunas compañías ofrecen gateways para conectar una red AppleTalk corriendo sobre LocalTalk, con redes IP corriendo sobre Ethernet. A pesar de que hay varios productos de este tipo, la mayoría de ellos incluyen los siguientes servicios:

* Las aplicaciones TCP/IP de un PC pueden conectarnos a sistemas TCP/IP de la Ethernet. Se definen utilidades especiales para permitirnos llevar datagramas IP desde el PC hasta el gateway, a través del LocalTalk. Las aplicaciones TCP/IP de PC han sido escritas usando unas librerías especiales que mezclan AppleTalk y TCP/IP. Las utilidades AppleTalk se necesitan para llevar los datagramas hasta el gateway, donde se transformarán en datagramas 100% TCP/IP, antes de dejarlos en la Ethernet.

* Se pueden escribir aplicaciones AppleTalk para grandes sistemas, de tal manera que un PC podrá usarlos como servidores. Dichas aplicaciones también han sido escritas haciendo uso de una librería especial que mezcla AppleTalk y TCP/IP. Pero, en esta ocasión, son utilidades TCP/IP para dejar datagramas en el gateway, donde se transformarán totalmente en AppleTalk, antes de dejarlos en la AppleTalk y lleguen al PC.

* Una red IP de un campus o una corporación puede ser usada para conectar redes AppleTalk. Los gateways de cada Applet realizarán las conversiones necesarias antes de enviar los datagramas a la red IP.

Además, algunos gateways pueden hacer traducciones a nivel de aplicación. Por ejemplo, algunos gateways pueden hacer traducciones entre el sistema de ficheros de Apple y el sistema de fichero de red de Sun (NFS). Esto permite a un PC acceder al sistema de ficheros Unix, donde el PC usa el sistema de ficheros Apple, y el acceso al sistema Unix se hace mediante el uso del sistema NFS, o sistema de ficheros de red (*Network File System*), de Sun.

Desafortunadamente, la gran flexibilidad de estos productos se traduce en una gran complejidad. El tema de direcciones es especialmente complicado. Por las mismas razones que SLIP, y PPP estos gateways usan frecuentemente asignación dinámica de direcciones IP. Para ello asignaremos un rango de direcciones IP a cada gateway. Cuando un PC intenta abrir una conexión TCP/IP, el gateway se hace con una dirección IP libre y se la asigna al PC. Al igual que SLIP, en muchos casos necesitaremos elegir si queremos que las direcciones se asignen de esta manera, o bien queremos que cada PC tenga su propia dirección. Otra vez, la elección dependerá del número de PC's que tengamos y de si

tenemos aplicaciones capaces de usar la dirección IP para identificar qué PC, en particular, es el que está conectado.

El direccionamiento es mucho más complejo, debido a que AppleTalk tiene su propia estructura de direcciones. Deberemos establecer una correspondencia entre direcciones AppleTalk y números de red IP. También habrá una correspondencia entre direcciones IP y AppleTalk, que se establecerá dinámicamente en los gateways.

3.4. SERVICIOS A NIVEL DE RED, NOMBRES.

Si vamos a tener una red TCP/IP, hay algunas tareas importantes que realizar. Algunas de ellas son simplemente de tipo administrativo. La más importante es crear un registro central de nombres y direcciones IP. Existen organizaciones que realizan esta labor para toda la red Internet. Si estamos conectados a Internet, el administrador de nuestro sistema necesita registrarse a una de estas organizaciones, para que cualquier demanda por parte de otra institución sobre nuestros hosts sean dirigidos a nuestros servidores.

Queremos mantener una base de datos que contenga la información de cada sistema de la red. Como mínimo, necesitaremos el nombre y la dirección IP de cada sistema. Probablemente, el registro central será el encargado de asignar las direcciones IP. Si nuestra red está estructurada en subredes, o si usamos varios números de clase C, el registro posiblemente asignará los números de red a las nuevas redes o subredes. Pero, habitualmente, se permitirá que los propios administradores de los hosts elijan el nombre del host. Sin embargo, el registro debe de, al menos, verificar que no haya nombres duplicados. Si estamos trabajando con una gran red, puede que sea buena idea delegar algunas de estas tareas a subregistros, posiblemente uno para cada departamento.

Se recomienda asignar las direcciones de la forma más simple: empezando por 1. Así, si nuestra red es la 128.6, podríamos asignar como 128.6.1 a la primera subred; 128.6.2, a la segunda, etc. La asignación de direcciones IP para hosts individuales podrían empezar por 2. De esta manera reservamos la dirección 1 de cada subred para que sea usada por el

gateway correspondiente. Por consiguiente, el primer host de la subred 128.6.4 sería el 128.6.4.2; el siguiente sería 128.6.4.3, y así sucesivamente. Hay una razón básica para mantener las direcciones tan cortas como sean posibles. Si tenemos una gran organización, podríamos quedarnos sin números de subred. Si esto ocurriera, y nuestros hosts tienen números de red bajos, podríamos asignar otro bit para el direccionamiento de las subredes. Si, por ejemplo, usamos el tercer octeto como número de subred, en tanto en cuanto nuestros hosts tengan unos números inferiores a 128, podremos ampliar el número de red a 9 bits. Así, por ejemplo, la subred 128.6.4 podría dividirse en dos subredes distintas: 128.6.4.0 y 128.6.4.128. Si hubiésemos asignado a los hosts números por encima de 128, la división habría sido imposible.

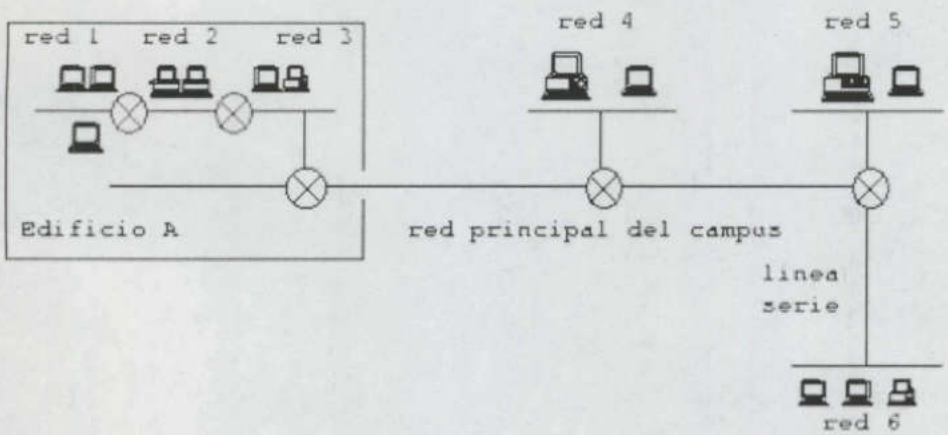
La asignación de nombres de los hosts no es tan sistemática. Pueden ser cualquier expresión compuesta de letras, números y guiones. Es más seguro que el primer carácter sea una letra. Y, desde el punto de vista de los usuarios, es recomendable que los nombres sean lo más cortos posibles (incluso hay software que tiene problemas trabajando con nombres más largos de 16 caracteres). Muchas veces, los departamentos o proyectos eligen un tema o nombre relacionado con ellos. Por ejemplo, las máquinas usadas por los estudiantes de Informática de Groucho Marx tienen nombres de bandas de rock: OASIS, BLUR, IRONMAIDEN, SAVOY, etc. Nuestro departamento de Matemáticas usa el nombre de famosos matemáticos: GAUSS, FERMAT, etc. Si la institución no tiene ninguna relación con el mundo exterior, cualquier nombre es adecuado.

Si estamos conectados a Internet, nuestra organización necesitará un *"nombre de dominio"* (*domain name*). Al igual que en el caso del espacio de direcciones IP, la autoridad máxima del espacio de nombres de Internet (DNS, Domain Name System) es la IANA (Internet Assigned Number Authority). La raíz del DNS es gestionada por el InterNIC por delegación de la IANA. Bajo la raíz se encuentran los distintos dominios de primer nivel (Top Level Domains o TLD's) gestionados por distintos registros delegados de Internet. Algunos de ellos son: Dominios "especiales" como COM, ORG, NET, EDU,... controlados por InterNIC (nodo central del Network Internet Center); y dentro de los dominios nacionales, el dominio ES, correspondiente a España, está delegado a ES-NIC.

A diferencia del número de red, podremos arreglárnosla sin él si la red está aislada. Si posteriormente lo necesitamos, es fácil de añadir un nombre de dominio. (Recomendamos usar un número de red desde el principio, porque cambiar números de red posteriormente puede ser traumático). Los nombres de dominio, normalmente, terminan en .EDU para las instituciones educativas, .COM, para las compañías, etc.

3.5.- PUENTES Y GATEWAYS.

En esta sección vamos a tratar con más detalle la tecnología usada para construir grandes redes. Vamos a centrarnos especialmente en cómo conectar varias Ethernet, token rings, etc. Hoy día la mayoría de las redes son jerárquicas. Los hosts están incluídos en una red de área local, como una Ethernet o un token ring. Estas redes se conectan entre sí mediante alguna combinación de redes principales o enlaces punto a punto. Una universidad puede tener una red como se muestra, en parte, a continuación:



Las redes 1, 2 y 3 están en un edificio. Las redes 4 y 5 están en edificios distintos del campus. La red 6 puede estar en una localización más distante. El diagrama anterior nos muestra que las redes 1, 2 y 3 están conectadas directamente, y los mecanismos que manejan las conexiones se marcan con *"x"*. El edificio A está conectado a otros edificios en el mismo campus por una red principal. El tráfico desde la red 1 a la red 5 tomará el siguiente camino:

- de 1 a 2 a través de la conexión entre estas redes;

- de 2 a 3 a través de su conexión directa;

- de 3 a la red principal;

- a través de la red principal, desde el edificio A al edificio donde la red 5 está emplazada;

- de la red principal a la red 5.

El tráfico hacia la red 6 debería pasar adicionalmente a través de la línea serie. Con la misma configuración, se usaría la misma conexión para conectar la red 5 con la red principal y con la línea serie. Así, el tráfico de la red 5 a la red 6 no necesita pasar a través de la red principal, al existir esa conexión directa entre la red 5 y la línea serie.

En esta sección vamos a ver qué son realmente estas conexiones marcadas con *"x"*.

3.5.1. DISEÑOS ALTERNATIVOS.

Hay que hacer constar que hay distintos diseños alternativos al mostrado anteriormente. Uno de ellos es usar líneas punto a punto entre los hosts, y otro puede ser usar una tecnología de red a un nivel capaz de manejar tanto redes locales como redes de larga distancia.

3.5.1.1. UNA RED DE LÍNEAS PUNTO A PUNTO.

En lugar de conectar los hosts a una red local como una Ethernet, y luego conectar dichas Ethernets, es posible conectar directamente los ordenadores a través de líneas serie de largo alcance. Si nuestra red consiste primordialmente en un conjunto de ordenadores situados en localizaciones distintas, esta opción tiene sentido.

En el primer diseño, la tarea de enrutamiento de los datagramas a través de red era realizada por unos mecanismos de propósito específico que marcábamos con *"x"*. Si hay líneas que

conectan directamente un par de hosts, los propios hosts harán esta labor de enrutamiento, al mismo tiempo que realizan sus actividades normales. A no ser que haya líneas que comuniquen directamente todos los hosts, algunos sistemas tendrán que manejar un tráfico destinado a otros. Por ejemplo, en nuestro diseño, el tráfico de 1 a 3 deberá pasar a través de 4, 5 y 6. Esto es perfectamente posible, ya que la inmensa mayoría de las implementaciones TCP/IP son capaces de reenviar datagramas. En redes de este tipo podemos pensar que los propios hosts actúan como gateways. Y, por tanto, deberíamos configurar el software de enrutamiento de los hosts como si se tratase de un gateway. Este tipo de configuraciones no es tan común como podría pensarse en un principio debido, principalmente, a estas dos razones:

* la mayoría de las grandes redes tienen más de un ordenador por localización. En estos casos es menos caro establecer una red local en cada localización que establecer líneas punto a punto entre todos los ordenadores;

* las unidades de propósito especial para conectar redes son más baratas, lo que hace que sea más lógico descargar las tareas de enrutamiento y comunicaciones a estas unidades.

Por supuesto, es factible tener una red que mezcle los dos tipos de tecnologías. Así, las localizaciones con más equipos podría manejarse usando un esquema jerárquico, con redes de área local conectadas por este tipo de unidades, mientras que las localizaciones lejanas con un sólo ordenador podrían conectarse mediante líneas punto a punto. En este caso, el software de enrutamiento usado en los ordenadores lejanos deberá ser compatible con el usado por las unidades conmutadoras, o bien tendrá que haber un gateway entre las dos partes de la red.

Las decisiones de este tipo generalmente se toman tras estudiar el nivel de tráfico de la red, la complejidad de la red, la calidad del software de enrutamiento de los hosts y la habilidad de los hosts para hacer un trabajo extra con el tráfico de la red.

3.5.1.2. TECNOLOGÍA DE LOS CIRCUITOS DE CONMUTACIÓN.

Otro enfoque alternativo al esquema jerárquico LAN/red principal es usar circuítos conmutadores en cada ordenador. Realmente, estamos hablando de una variante de la técnica de las líneas punto a punto, donde ahora el circuíto conmutador permite tener a cada sistema aparentar que tiene línea directa con los restantes. Esta tecnología no es usada por la mayoría de la comunidad TCP/IP debido a que los protocolos TCP/IP suponen que el nivel más bajo trabaja con datagramas aislados. Cuando se requiere una conexión continuada, el nivel superior de red la implementa usando datagramas. Esta tecnología orientada al datagrama no coincide con este sistema orientado a los circuítos de forma directa. Para poder usar esta tecnología de circuítos conmutadores, el software IP debe modificarse para ser posible construir circuítos virtuales de forma adecuada. Cuando hay un datagrama para un destino concreto se debe abrir un circuíto virtual, que se cerrará cuando no haya tráfico para dicho destino por un tiempo. Un ejemplo de este enfoque es la DDN (Defense Data Network). El protocolo principal de esta red es el X.25. Esta red parece desde fuera una red distribuída X.25. El software TCP/IP trata de manejar la DDN mediante el uso de canales virtuales. Técnicas similares podrían usarse con otras tecnologías de circuítos de conmutación, como, por ejemplo, ATT's DataKit, aunque no hay demasiado software disponible para llevarlo a cabo.

3.5.1.3. REDES DE UN SÓLO NIVEL.

En algunos casos, los adelantos en el campo de las redes de larga distancia pueden sustituir el uso de redes jerárquicas. Muchas de las redes jerárquicas fueron configuradas así para permitir el uso de tecnologías tipo Ethernet y otras LAN, las cuáles no pueden extenderse para cubrir más de un campus. Así que era mecesario el uso de líneas serie para conectar las distintas LANs de varios lugares. Sin embargo, ahora hay tecnologías de características similares a Ethernet, pero que pueden abarcar más de un campus y, por tanto, pensar en una sola red de larga distancia que no hace uso de una estructura jerárquica.

Las principales limitaciones de este tipo de redes son cuestiones de rendimiento y flexibilidad. Si una sola red es usada por todo el campus es muy fácil que se sobrecargue. Las redes jerárquicas pueden manejar un volumen de trabajo mucho mayor que las redes de un solo nivel. Además, el tráfico dentro de los departamentos tiende a ser mayor que el tráfico entre departamentos.

3.5.1.4. DISEÑOS MIXTOS.

En la práctica, pocas redes se permiten el lujo de adoptar un diseño teóricamente puro.

Es poco probable que una red grande sea capaz de evitar el uso de un diseño jerárquico. Supongamos que la configuramos como una red de un solo nivel. Incluso si la mayoría de los edificios tienen sólo uno o dos ordenadores, habrá alguna localización donde haya bastantes ordenadores para justificar el uso de una red local. El resultado es una mezcla entre una red de un solo nivel y una red jerárquica. En la mayoría de los edificios sus ordenadores están conectados directamente a una red de área amplia, como una red de un solo nivel, pero en un edificio hay una red de área local usando su red de área amplia como red principal, a la cuál se conecta a través de unidades conmutadoras.

Por otro lado, incluso los diseñadores de redes que defienden el uso de una enfoque jerárquico, en muchas ocasiones encuentran partes de redes donde simplemente no resulta económico instalar una red de área local, así que algunos hosts se enganchan directamente a la red principal, o bien se usa una línea serie.

Además de las razones económicas de la instalación en sí, hay que tener en cuenta que a la larga hay que valorar aspectos de mantenimiento, de manera que a veces es mejor hacer un desembolso económico en el diseño para ahorrarnos dinero en el mantenimiento futuro. Por tanto, el diseño más consistente será aquél que podamos ser capaces de mantener más fácilmente.

3.5.2. INTRODUCCION A LAS DISTINTAS TECNOLOGIAS DE CONMUTACION.

En esta sección discutiremos las características de varias tecnologías usadas para intercambiar datagramas entre redes. En efecto, trataremos de dar más detalles sobre esas *"cajas negras"* que hemos visto en las anteriores secciones. Hay tres tipos básicos de conmutadores, como *repetidores*, *bridges* (o puertas) y *gateways* (o pasarelas), o, alternativamente, *switches* de nivel 1, 2 y 3 (basándonos en el nivel del modelo OSI en el que operan). También hay que aclarar que hay sistemas que combinan características de más de uno de estos dispositivos, especialmente bridges y gateways.

Las diferencias más importantes entre estos tipos de dispositivos residen en el grado de aislamiento a fallos, prestaciones, enrutamiento y las facilidades que ofrecen para la administración de la red. Más adelante examinaremos esto con más detalle.

La diferencia mayor se encuentra entre los repetidores y los otros dos tipos de *switches*. Hasta hace relativamente poco tiempo, los gateways proporcionaban unos servicios muy distintos a los ofrecidos por los bridges, pero ahora hay una tendencia a unificar estas dos tecnologías. Los gateways están empezando a adoptar un hardware de propósito específico que antes era característico de los bridges. Los bridges están empezando a adoptar un enrutamiento más sofisticado, características de aislamiento y de administración de redes que antes sólo se podían encontrar en los gateways. Incluso hay sistemas que pueden funcionar como bridge y gateway. Esto significa que la decisión crucial no es decidir si tenemos que usar un bridge o un gateway, sino qué características necesitamos en un switch y cómo éste afecta el diseño global de la red.

3.5.2.1. REPETIDORES.

Un repetidor es un equipo que conecta dos redes que usan la misma tecnología. Recibe los paquetes de datos de cada red y los retransmite a la otra red. La red resultante se caracteriza por tener la unión de los paquetes de ambas redes. Para las redes Ethernet, o que cumplen el

protocolo IEEE 802.3, hay dos tipos de repetidores (otras tecnologías de red no hacen estas distinciones).

Un repetidor trabaja a muy bajo nivel. Su objetivo principal es subsanar las limitaciones de la longitud del cable que provocan pérdidas de señal, dispersión temporal, etc. Nos permiten construir redes más grandes y liberarnos de las limitaciones de la longitud del cable. Podríamos pensar que un repetidor se comporta como un amplificador a ambos lados de la red, pasando toda la información contenida en la señal (incluso las colisiones) sin hacer ningún procesamiento a nivel de paquetes. No obstante, hay un número máximo de repetidores que pueden introducirse en una red. Las especificaciones básicas de Ethernet requieren que las señales lleguen a su destino dentro de un límite de tiempo, lo que determina que haya una longitud máxima de la red. Poniendo varios repetidores en el camino se introducen dificultades para estar dentro del límite (de hecho, cada repetidor introduce un retraso, así que de alguna manera se introducen nuevas dificultades).

Un *"repetidor con buffer"* trabaja a nivel de paquetes de datos. En lugar de pasar la información contenida en la señal, almacena paquetes enteros de una red en un buffer interno y, luego, lo retranstime a la otra red, por lo que no deja pasar las colisiones. Debido a que los fenómenos de bajo nivel, como las colisiones, no son repetidos, se puede considerar como si las dos redes continuasen separadas en lo que se refiere a las especificaciones Ethernet. Por tanto, no hay restricciones respecto al número de repetidores con buffer que se pueden usar. De hecho, no es necesario que ambas redes sean del mismo tipo, pero han de ser suficientemente similares, de manera que tengan el mismo formato de paquete. Generalmente, esto significa que se emplean repetidores con buffer entre redes de la familia IEEE 802.x (asumiendo que elegimos la misma longitud para las direcciones y el mismo tamaño máximo para los paquetes), o entre dos redes de otra familia. Además, un par de repetidores con buffer pueden usarse para conectar dos redes mediante una línea serie.

Los repetidores con buffer y los repetidores básicos tienen una característica en común: repiten cada paquete de datos que reciben de una red en la otra. Y así ambas redes, al final, tienen exactamente el mismo conjunto de paquetes de datos.

4 ADMINISTRACIÓN Y CONFIGURACIÓN DE NET8

4.1.-CONFIGURACIÓN DE NET8.

Una vez instalado el servidor y creada la base de datos, configuramos los servicios de red. Contamos con la herramienta *Net Easy Config* mediante el comando netec, mediante el cual creamos un nuevo servicio (*service name*):

1. Seleccionamos la opcion *Create* y asignamos un nombre de servicio (en mi caso oracle).
2. Elegimos el protocolo de red a utilizar: *TCP/IP*
3. Introducimos el nombre de host ó la dirección IP, del servidory como puerto dejamos por defecto el 1521.
4. En el campo *Oracle8i release 8.1 Service Name* introducimos el nombre de la base de datos (en nuestro caso oracle8i.world).
5. Se nos ofrece la opción de probar la configuración del servicio, pero no es recomendable (a mí no me funciona).

De esta manera hemos creado el archivo tnsnames.ora en el directorio $ORACLE_HOME/network/admin. Además se crea un *Listener* de nombre listener.

A continuación reservamos un puerto para el *listener* en el archivo /etc/services, añadiendo la siguiente linea:

listener 1521/tcp # Net8 listener

Para probar si el *listener* se ha configurado correctamente, intentamos arrancarlo manualmente, con el comando:

$ lsnrctl start

Si el *listener* ya estuviera arrancado obtenemos un mensaje como el siguiente: **TNS-01106: El listener que usa el nombre LISTENER ya ha sido arrancado.**

Podemos obtener el estado del *listener* con el comando:

$ lsnrctl status

Y deberemos obtener algo como lo que sigue:

LSNRCTL for Linux: Version 8.1.5.0.0 - Production on 05-OCT-99 17:54:56
(c) Copyright 1998 Oracle Corporation. All rights reserved.

Conectándose a (ADDRESS=(PROTOCOL=ipc)(KEY=PNPKEY))
ESTADO del LISTENER

Alias LISTENER
Versión TNSLSNR for Linux: Version 8.1.5.0.0 - Production
Fecha de Inicio 05-OCT-99 17:49:14
Tiempo de actividad 0 días 0 hr. 5 min. 42 seg.

Nivel de Rastreo off
Seguridad OFF
SNMP OFF
Fichero Log del Listener /opt/oracle/app/oracle/product/8.1.5/network/log/listener.log
Resumen de servicios...
ORCL tiene 2 gestores de servicio.
El comando se ha ejecutado correctamente.

Si al ejecutar el comando lsnrctl obtenemos un mensaje como el siguiente: **lsnrctl: error in loading shared libraries: libclntsh.so.8.0**

Se trata de un problema con las librerias dinámicas, y podemos resolverlo añadiendo la linea siguiente al archivo /etc/ld.so.conf:

/opt/oracle/app/oracle/product/8.1.5/lib

Y ejecutamos a continuación:

 # ldconfig

Podemos automatizar el arranque y la parada del *listener* en el script dbora,añadiendo las lineas:

su - $ORA_OWNER -c "$ORA_HOME/bin/lsnrctl start"

su - $ORA_OWNER -c "$ORA_HOME/bin/lsnrctl stop"

en las secciones 'start' y 'stop' respectivamente.

Para comprobar que el *listener* funciona correctamente, y el archivo tnsnames.ora es correcto, ejecutamos el comando:

$ tnsping oracle

Con el *listener* y la base de datos arrancada, podemos probar desde un cliente SQLPlus:

$ sqlplus

SQL*Plus: Release 8.1.5.0.0 - Production on Mar Oct 5 19:43:11 1999

(c) Copyright 1999 Oracle Corporation. All rights reserved.

Introduzca el nombre de usuario: **system/manager@oracle**

Conectado a:

Oracle8i Enterprise Edition Release 8.1.5.0.1 - Production

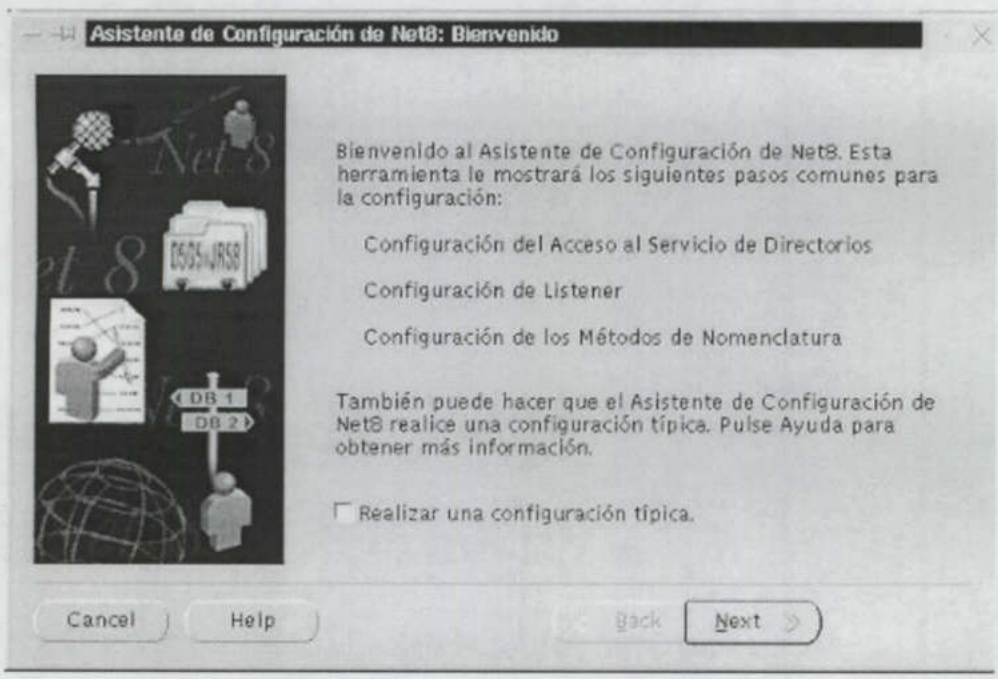
With the Partitioning and Java options

PL/SQL Release 8.1.5.0.0 - Production

SQL>

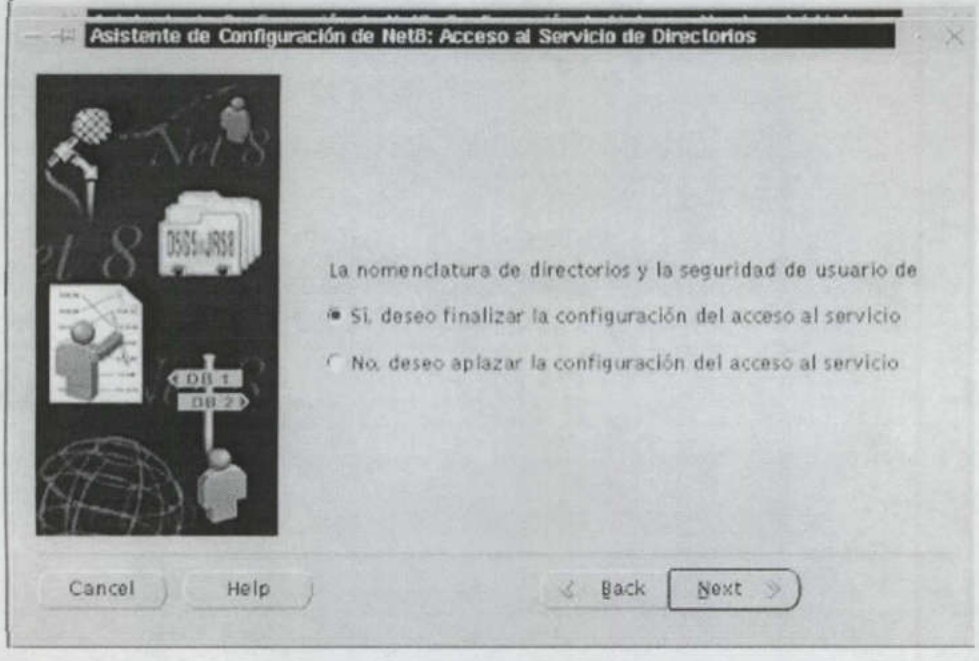
4.1.1 CONFIGURACIÓN DESDE EL ASISTENTE

Explicaré la configuración del Net8 desde el asistente:

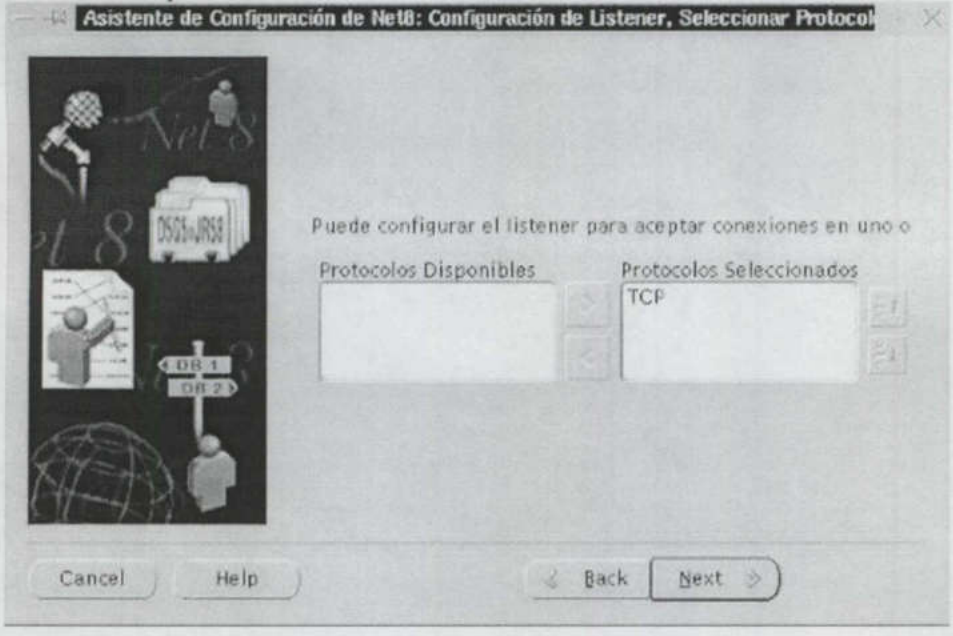


Esta es la pantalla inicial del asistente, aquí explicaré lo básico para crear un fichero listener.ora, que será el que utilice el proceso escucha para aceptar conexiones locales o remotas a la base de datos, y el tnsnames.ora, para definir alias de conexión a la base de datos...

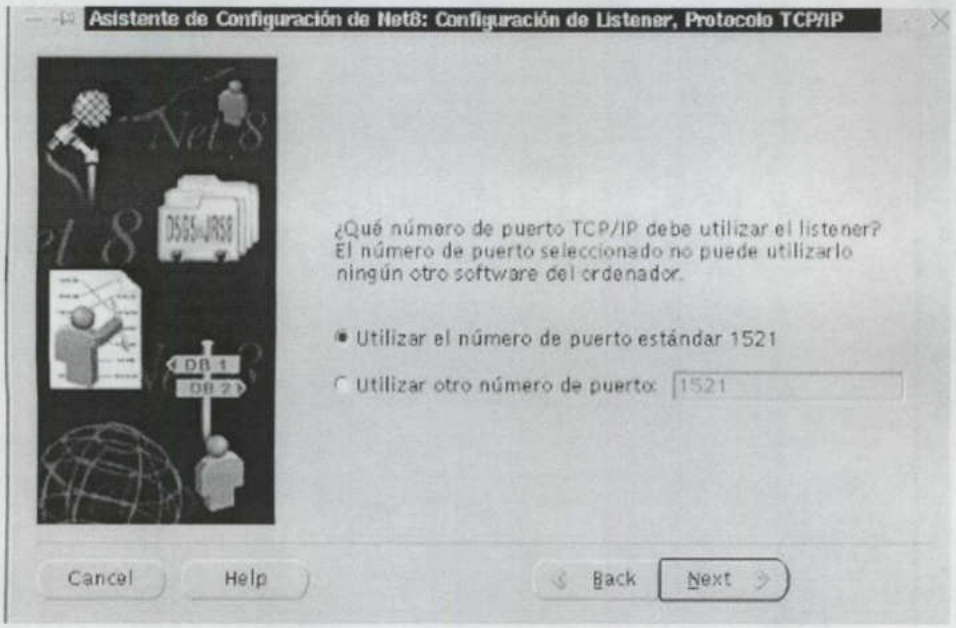
... en esta pantalla aplazamos la configuración del servicio de directorios, y pasamos a la siguiente pantalla...



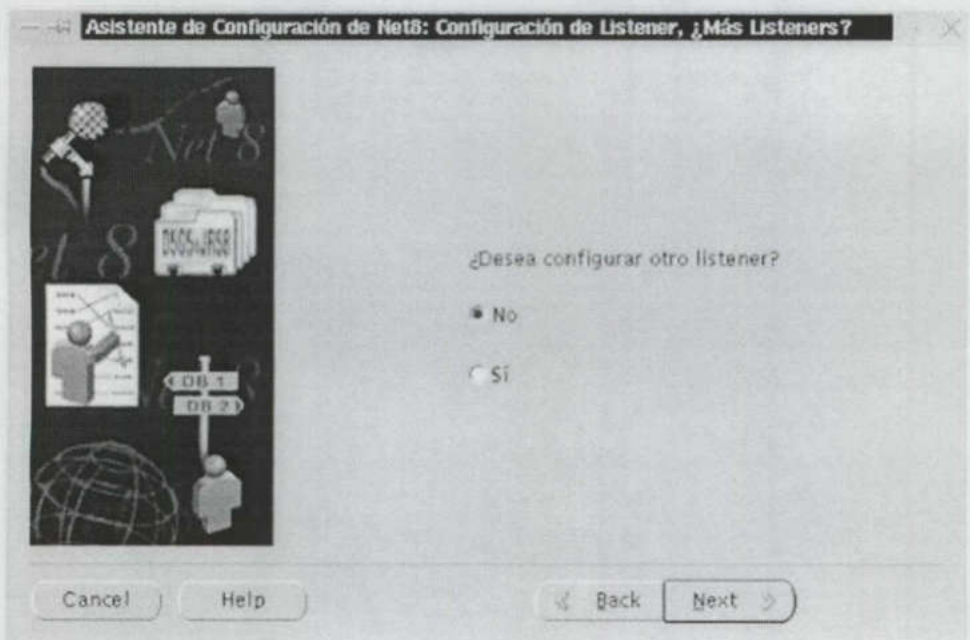
..., comenzamos creando el fichero para el proceso de escucha (listener), aceptamos el nombre por defecto...



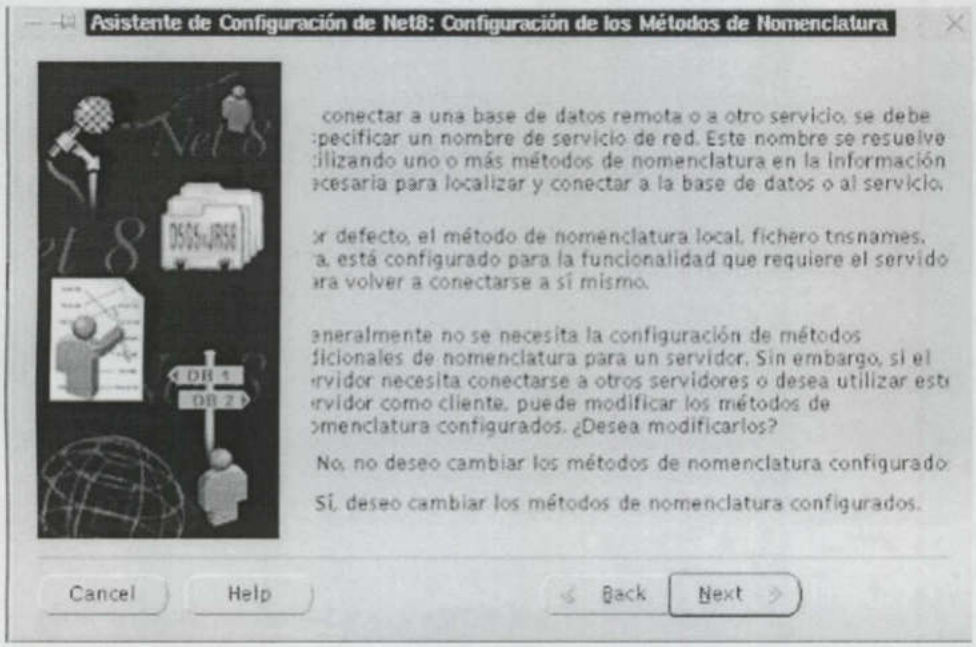
..., en protocolos seleccionamos TCP, para aceptar conexiones a la base de datos por TCP/IP...



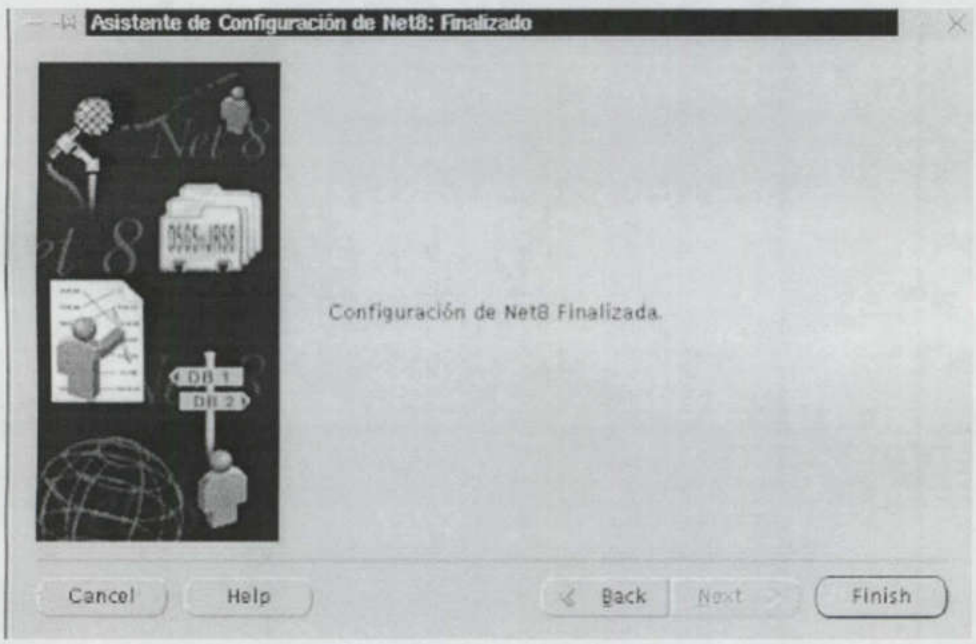
..., ahora debemos seleccionar el puerto por el que la base de datos escucha, por defecto es el 1521, otros que se suelen utilizar son 1526, 1531...



..., podemos configurar otro proceso de escucha, pero en este caso no lo vamos a crear...



...Una vez configurado el listener, pasaremos a la creación del fichero de alias para poder conectarnos a otras bases de datos en otros equipos, seleccionaremos que no, y terminaremos con la configuración del Net8.



4.2.-ADMINISTRACION Y OPTIMIZACIÓN DE BASE DE DATOS ORACLE

Sql * Net

Instalación de Oracle

Servidor		Cliente
SGBDR O8		Aplicación (SQL * Plus, ...)
SQL * Net Server		SQL * Net Client
TCP/IP Protocol Adapter		TCP/IP Protocol Adapter
Red TCP/IP	Red TCP/IP	Red TCP/IP

SQL * Net V1

- Cadenas de conexión

```
prefijo del protocolo:nombre del servidor:nombre de la
instancia
```

Prefijos de protocolo

TCP/IP t

IPX/SPX

SNA

DECNet d

SQL * Net V2

- TNS (Transparent Network Substrate; sustrato de red transparente)
- MPI(Multiprotocol Interchange; Intercambio Multiprotocolo)

Listener

```
c:\orant\bin>lsnrctl80.exe
```

HELP

HELP SET

HELP SET PASSWORD

SET PASSWORD ORACLE

START

STOP

SERVICES

STATUS

tnsnames.ora

```
prueba1.WORLD =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS =
    (PROTOCOL = TCP)
    (Host = 127.0.0.1)
    (Port = 1521)
   )
  )
  (CONNECT_DATA =
   (SID = ORCL)
  )
 )
```

- Los puertos de Oracle por defecto son 1521 y 1526
- Para conocer la IP del equipo

```
c:\>ipconfig /ALL
```

- Cambios en tnsnames.ora (cliente)
  - Automáticos
  - Reiniciar SQL * Plus (caché)
- Cambios en listener.ora (servidor)
  - Reiniciar listener

Reiniciar listener

- set password oracle
- stop
- set password oracle
- start

Cambiar la password del listener

- editar el fichero listener.ora
- set password vieja_password
- stop
- set password nueva_password
- start
- Nota: la password sólo se comprueba en el stop

Herramientas

SQL * Net V2:

- SQL * Net Easy Configuration (tnsnames.ora)
- SQL * Net Configuration Tool (tnsnames.ora)

SQL * Net 8:

- Oracle Net 8 Assistant (tnsnames.ora / Oracle Names Server)
- Oracle Net 8 Easy Config (tnsnames.ora)

Oracle Names Server

SQL * Net V2:

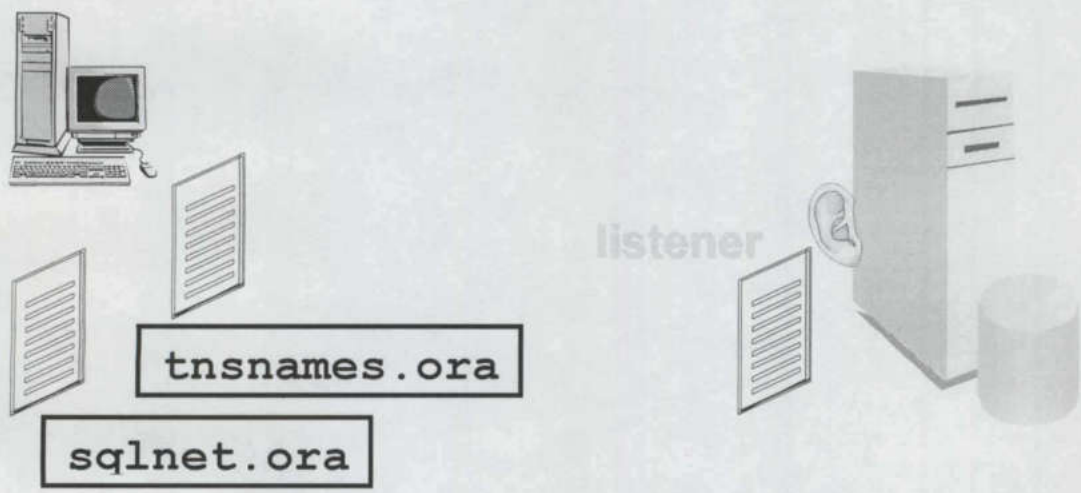
- Oracle Net 8 Assistant
- Crea un servicio en el Panel del Control "OracleNamesService80NombreDelServidorDeNombres"
- Crea los ficheros de configuración
  - c:\orant\net80\admin\names.ora
  - c:\orant\net80\names\ckpcfg.ora
  - c:\orant\net80\admin\ckpreg.ora
  - c:\orant\net80\net8asst\NETPROPERTIES
- Mensajes de log en c:\orant\net80\log\names.log

Versiones de SQL * Net y SGBDR Oracle

Versión Oracle	5 6 7.2	7.3	8
Versión SQL * Net	1	2	Net 8
Configuración cliente	-	c:\orant\network\admin\tnsnames.ora	c:\orant\net80\admin\tnsnames.ora
Configuración servidor	-	c:\orant\network\admin\listener.ora	c:\orant\net80\admin\listener.ora
Controlador del	-	c:\orant\bin\lsnrctl.exe	c:\orant\bin\lsnrctl80.exe

listener en el servidor			
Proceso Listener	-	c:\orant\bin\tnslsnr.exe	c:\orant\bin\tnslsnr80.exe
Servicio Listener	-	OracleTNSListener	OracleTNSListener80
Pruebas	-	-	c:\orant\bin\tnsping80.exe

4.2.1.-EL PROCESO LISTENER



CONCLUSIONES

Primero es posible pensar en un sistema de base de datos como un sistema de registros computarizado. Dicho sistema comprende a los propios datos (almacenados en una base de datos), al hardware, al software (en particular al sistema de administración de base de datos o DBMS) y a los usuarios. A su vez, los usuarios pueden ser divididos en programadores de aplicaciones, usuarios finales y administrador de base de datos o DBA. El DBA es el responsable de administrar la base de datos y el sistema de base de datos, de acuerdo con las políticas establecidas por el administrador de datos.

Las bases de datos están integradas y por lo regular son compartidas; se emplean para almacenar datos persistentes. Dichos datos pueden considerarse, de manera útil aunque informal como una representación de entidades, junto con los vínculos que están entre éstas (aunque de hecho, un vínculo es en realidad solo una clase especial de entidad).

Los sistemas de bases de datos ofrecen diversos beneficios. Uno de los más importantes es el de la independencia (física) de los datos. Podemos definir la independencia de los datos como la inmunidad que tienen los programas de aplicación ante los cambios en la forma almacenar o acceder físicamente a los datos. Entre otras cosas, la independencia de los datos requiere que se haga una clara distinción entre el modelo de datos y su implementación.

Los sistemas de base de datos también soportan por lo regular transacciones o unidades de trabajo lógicas. Una ventaja de las transacciones es que está garantizado que sean atómicas (todo o nada), incluso si el sistema falla a mitad de su ejecución.

Los sistemas de base de datos pueden estar fundamentados en varias teorías diferentes. En particular, los sistemas relacionales se basan en una teoría formal denominada modelo relacional, según la cual los datos están representados como filas de tablas (interpretadas

como proposiciones verdaderas) y cuentan con operadores que manejan directamente el proceso de inferir proposiciones verdaderas adicionales a partir de las ya dadas. Desde una perspectiva tanto económica como teórica, los sistemas relacionales son sin duda los mas importantes.

BIBLIOGRAFIA

- Comunicaciones y redes de computadores, 5ª ed.
  Stallings, W. (1997).
  Prentice Hall Iberia

- Redes de Computadoras. Tercera Edición.
  Tanenbaum, A.S. (1997).
  Prentice-Hall.

- Comunicaciones de datos, rede de computadores y sistemas abiertos.
  Halsall, F. (1998).
  Addison-Wesley.

- Redes de Telecomunicaciones, Protocolos, Modelado y Análisis.
  M. Schwartz
  Addison-Wesley, 1994.