COMPARACION ESTRUCTURAL DE INGRES Y ORACLE EN EL AMBIENTE UNIX

Por:
Roy Cahn-Speyer Valenzuela
Alfredo Hernández Cifuentes
Andrés Suárez Rosselli

Ministerio de Hacienda y Crédito Público

Bogotá, Colombia, Agosto de 1987

Presentado a la XIII Conferencia Latinoamericana de Informática.

En 1970 E.F. Codd propuso el modelo relacional para diseño de bases de datos. Luego de una década de trabajo investigativo tendiente a llevar a la práctica este modelo, ORACLE e INGRES fueron anunciados, en 1979 y 1980 respectivamente, como los primeros Sistemas Manejadores de Bases de Datos Relacionales (SMBDR) comerciales. Desde entonces se han lanzado al mercado múltiples implementaciones del modelo relacional. A pesar de ello, INGRES y ORACLE siguen siendo líderes de la tecnología relacional.

También a comienzos de la década del 70 se desarrolló y tomó altura comercial el sistema operacional UNIX. Entre las características de UNIX se destacan las siguientes: (1) Está escrito en un 95% en un lenguaje de alto nivel ("C"), lo que le da flexibilidad para el desarrollo de nuevo "software" y lo hace independiente en alto grado de los dispositivos de "hardware". (2) Provee un conjunto extenso de herramientas orientadas hacia el desarrollo incremental de "software". UNIX tiende a ser un estándar en sistemas operacionales. Recientemente UNIX ha salido de las universidades y los laboratorios científicos hacia ambientes gubernamentales y comerciales.

El presente artículo compara la implementación de INGRES y ORACLE en el ambiente UNIX desde el punto de vista de su arquitectura de archivos y procesos.

COMPARACION ESTRUCTURAL DE INGRES Y ORACLE EN EL AMBIENTE UNIX

1. INTRODUCCION

1.1 Presentación.

El presente artículo examina la implementación sobre UNIX de dos sistemas manejadores de bases de datos relacionales (SMBDR): INGRES, de Relational Technologies, y ORACLE, de Oracle Corporation.

Para cada SMBDR se estudió su arquitectura particular sobre UNIX, ubicando dos aspectos:

a. Estructura de procesos del SMBD como parte de la estructura de procesos de UNIX.
b. Estructura de archivos del SMBD como parte del "file system" de UNIX.

Tanto INGRES como ORACLE están clasificados como SMBD "totalmente relacionales" [CODD79][CODD83], en el sentido de que satisfacen las siguientes condiciones:

1. Toda la información de la base de datos está representada como valores en tablas.

2. No hay enlaces de navegación visibles por el usuario entre las tablas.

3. Los sistemas soportan todos los operadores del álgebra relacional, sin recurrir a comandos especiales para iteración y recursión, y sin restricciones impuestas por los métodos de acceso predefinidos.

4. Soportan las dos reglas generales de integridad del modelo relacional básico.

En la actualidad INGRES y ORACLE son los dos SMBDR con mayor éxito comercial. Su estrategia de mercadeo es la de ofrecer total transportabilidad a través de distintas marcas y tamaños de computadores. INGRES es el resultado de un proyecto de investigación de la Universidad de California, Berkeley, e inició su vida comercial en 1980. ORACLE se comenzó a desarrollar en 1977. La primera copia comercial de ORACLE se produjo en 1979 [SCHM83]. Su desarrollo teórico y práctico ha sido paralelo, aunque ha resultado en arquitecturas diferentes, como se muestra en este artículo.

A continuación se presentan los aspectos generales del manejo de archivos y procesos de UNIX. Las dos secciones siguientes hacen una descripción de la implementación sobre UNIX de INGRES y de ORACLE, respectivamente.

1.2 Generalidades del ambiente UNIX.

a. Sistema de archivos de UNIX ("file system").

Tiene estructura de àrbol. Cada archivo es o un directorio, con referencias a archivos del siguiente nivel, o un archivo de datos.

Cada archivo està dividido fisicamente en bloques (generalmente de 512 o 1024 bytes), llamados pàginas. UNIX utiliza un àrea global de memoria llamada "buffer pool" para guardar temporalmente las pàginas a medida que son escritas o leìdas. El tamaño de este "buffer pool" se define al generar el sistema operacional.

Para responder una solicitud de lectura, UNIX mueve una o màs pàginas de memoria secundaria al "buffer pool" (si es necesario, haciendo que otras pàginas sean escritas al disco) y luego le retorna al usuario la cadena de bytes deseada. Si la misma pàgina es referenciada otra vez (por el mismo u otro usuario) cuando todavìa està en el "buffer pool", entonces no hay entrada/salida real al disco. Una solicitud de escritura simplemente mueve los datos escritos al "buffer pool"; en algùn momento posterior el manejador de "buffers" graba las pàginas en el disco.

El manejador de "buffers" de UNIX utiliza la estrategia de reemplazo de la "pàgina menos recientemente usada" (LRU). Cuando UNIX detecta acceso secuencial a un archivo, hace una prebùsqueda de las pàginas antes de que èstas sean solicitadas.

Las pàginas presentes en el "buffer pool" son escritas en el disco por varios mètodos:

- Cuando se emite el llamado del sistema "fsync".
- Cuando se ejecuta el comando de UNIX "sync".
- Cuando el "buffer pool" se aproxima a su lìmite de capacidad.
- Aproximadamente cada 30 segundos por el mismo UNIX.

b. Estructura de procesos de UNIX.

Un proceso en UNIX es un espacio de direcciones (de mìnimo 64K) que se asocia a un "user-id" y es la unidad de trabajo que maneja el "scheduler" de UNIX. El tamaño de un proceso està limitado por la capacidad de manejo de memoria virtual de la CPU.

Los procesos se pueden bifurcar en subprocesos; asì, un proceso dado puede ser la raìz de un àrbol de subprocesos. Varios procesos se pueden comunicar por medio de "pipes", señales, semàforos y memoria compartida.

UNIX permite que los procesos ejecuten còdigo reentrante, para que varios procesos compartan segmentos ejecutables. También se pueden definir segmentos de datos compartidos por varios procesos.

2. ARQUITECTURA DE INGRES SOBRE CENTIX*

2.1. Generalidades.

INGRES (Interactive Graphics and Retrieval System) es un sistema manejador de bases de datos relacional implementado originalmente sobre UNIX. La implementación está desarrollada primariamente en C, lenguaje de alto nivel en el que también está escrito UNIX [STON76]. INGRES soporta el lenguaje de interrogación QUEL (QUEry Language), y a partir de la versión 4 también soporta SQL, el cual es un estándar (ANSI e ISO).

La arquitectura de INGRES se mantuvo sin variaciones importantes desde su concepción inicial, durante el lanzamiento comercial de INGRES en 1980, hasta la versión 2 de 1984, similar a la que actualmente se conoce como INGRES/CS (para sistemas compactos). La versión 3 y posteriores tienen cambios significativos, principalmente para operar en equipos más poderosos y mejorar el "performance" de INGRES.

Las versiones de INGRES disponibles comercialmente no proveen documentación detallada de su arquitectura interna, particularmente en lo que a la estructura de procesos se refiere. Por esta razón, la principal fuente fué la observación directa sobre dos implementaciones diferentes, utilizando las herramientas de monitoreo que provee el sistema UNIX. Las implementaciones observadas fueron las siguientes:

- INGRES versión 3, bajo CENTIX versión 5.01, en un equipo Burroughs XE-550-3.
- INGRES/CS (versión 2), bajo UNIX Sistema V, en un equipo Tower/XP de NCR.

Esta sección describe la estructura de archivos y procesos de INGRES versión 3 desde el punto de vista de la implementación, es decir, INGRES operando bajo un ambiente específico de UNIX.

2.2. Estructura de archivos de INGRES.

Es igual en las dos versiones observadas, y se acoge a la arquitectura descrita en [STON76], aunque algunos nombres de directorios han cambiado.

(*) CENTIX es la forma comercial de UNIX Sistema V que ofrece UNISYS en la línea del desaparecido Burroughs.

2.2.1. Estructura del directorio "ingres":

INGRES manipula un subàrbol del "file system" de UNIX, cuya raíz es el directorio del usuario "ingres". El usuario "ingres" es el superusuario de INGRES. En dicho subàrbol residen por defecto las bases de datos además de los programas, librerías y utilidades de INGRES.

Los subdirectorios más importantes del directorio "ingres" son:

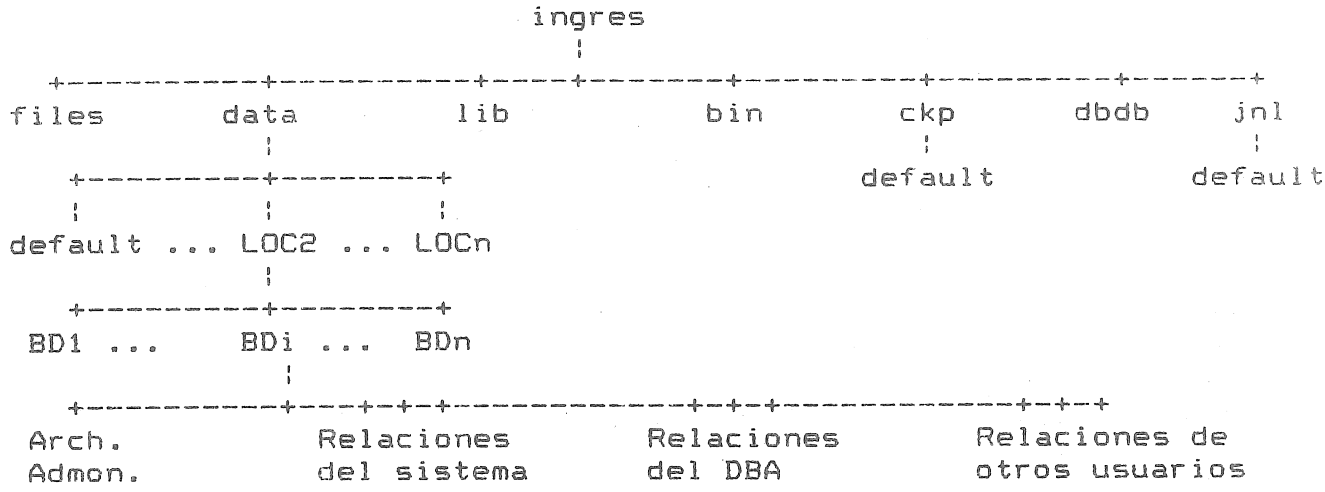

Figura 2.1

- files: Contiene los archivos con los textos de ayuda y los mensajes de error de todas las herramientas de INGRES.

- bin: Incluye todos los màdulos ejecutables de INGRES.

- lib: Contiene las librerías para compilación con EQUEL (el precompilador de QUEL).

- data: Tiene un subdirectorio "default", en el cual se mantienen subdirectorios con todos los archivos correspondientes a cada base de datos, a menos que el usuario "ingres" especifique otras "locaciones" (ver sección 2.2.2.) para algunas base de datos o para algunas tablas (estas locaciones alternas se muestran en la figura 2.1 como LOC2 ... LOCn). Otros subdirectorios de "data" corresponden a locaciones diferentes de la locación "default".

- data/default: Tiene un subdirectorio con el diccionario de datos y las relaciones de cada base de datos.

- ckp y jnl: Contienen un subdirectorio por cada locación (además del subdirectorio "default"), y en cada uno de ellos un subdirectorio por cada base de datos que utilice esa locación. En cada subdirectorio residen los archivos de "checkpoint" y de "journal" (ver sección 2.2.2.), respectivamente, para la base de datos correspondiente.

- dbdb: Contiene el diccionario de datos de las bases de datos, es decir, es la base de datos administrativa. Es una base de datos especial mantenida por INGRES. Se utiliza para almacenar información sobre bases de datos y usuarios autorizados de INGRES, y para especificar las bases de datos que pueden ser utilizadas por cada usuario. También mantiene la información de los directorios y dispositivos en los cuales residen las bases de datos particulares (cuando no se desea usar el "default").

2.2.2. Tipos de archivos de INGRES:

El sistema INGRES se compone de las imágenes ejecutables de INGRES, los archivos de mensajes y las librerías, además de las bases de datos. Una base de datos INGRES consiste de tres componentes: tablas, "checkpoints" y "journals".

En disco, una base de datos está repartida en un conjunto de directorios, y es constituida por una colección de archivos. Dichos directorios son creados como subdirectorios de locaciones INGRES predefinidas. Una locación es un directorio de UNIX (CENTIX) que el sistema INGRES identifica como un directorio en el cual pueden ser colocados componentes de las bases de datos de INGRES. Existe una utilidad especial ("accessdb") para definir locaciones.

i. Tablas:

Una tabla de una base de datos INGRES corresponde a un archivo UNIX. Las tablas de cada base de datos son de cuatro tipos:

a. Archivo de administración. Contiene el "user-id" del DBA ("Data Base Administrator" o administrador de la base de datos, es decir, el usuario que la creó) e información administrativa.

b. Relaciones del sistema. Tienen nombres predefinidos y existen para cada base de datos. Pertenecen al DBA. Junto con el archivo de administración constituyen el diccionario de datos de la base de datos. Pueden ser interrogadas por cualquier usuario autorizado por medio de QUEL. Sólo los puede modificar INGRES de manera automática, o excepcionalmente, el usuario "ingres", quien es el superusuario de todas las bases de datos (i.e., el DBA de la base de datos "dbdb").

c. Relaciones del DBA. Estas son relaciones de la base de datos creadas por el DBA, pero son compartidas en el sentido que cualquier usuario autorizado puede acceder a ellas. El DBA puede modificar los niveles de protección de estas relaciones para cada usuario y para cada relación.

d. Otras relaciones. Son relaciones de la base de datos creadas por usuarios diferentes del DBA. No son compartidas.

El DBA tiene poderes no delegables adicionales a los de los demàs usuarios: crear relaciones compartidas y especificar controles de acceso a ellas; destruir cualquier relaciòn de la base de datos (excepto las del sistema).

A excepciòn del archivo de administraciòn, cada archivo es tratado como una relaciòn. Mantener los archivos del sistema como relaciones trae las siguientes ventajas:

- Se economiza còdigo de acceso.
- Se pueden usar las mismas estructuras de informaciòn que se usan para las demàs relaciones.
- Se pueden examinar vìa QUEL.

Cada relaciòn està separada en un archivo fìsico diferente, i.e., no se intenta colocar en el mismo archivo tuplas de diferentes relaciones.

INGRES maneja sus propios mètodos de acceso para las relaciones, y administra el espacio libre dentro de cada archivo UNIX.

Tanto las tablas del diccionario de datos como las tablas definidas por los usuarios son colocadas en las locaciones como archivos individuales de UNIX. Las relaciones del sistema siempre son creadas en la locaciòn "home" asociada a cada base de datos. La locaciòn "home" por defecto para una base de datos es la locaciòn "default" (que corresponde al directorio "ingres/data/default" segùn se viò en la secciòn 2.2.1.), y se define cuando se crea la base de datos (mediante el comando "createdb").

Las tablas de los usuarios son tambièn creadas en la locaciòn "home" para la base de datos respectiva, a menos que se especifique alguna locaciòn diferente en el comando QUEL o SQL que crea la tabla. Ademàs pueden ser reubicadas en una nueva locaciòn utilizando el comando "relocate".

ii. Checkpoints:

Un checkpoint es una imagen UNIX "tar" de una base de datos, con un significado especial (ver TAR(C) en la documentaciòn de UNIX). El DBA crea un "checkpoint" en un momento en el cual se sabe que la base de datos es consistente. Asì, el "checkpoint" se convierte en un "backup" seguro del cual se puede regenerar la base de datos.

Cada "checkpoint" consiste de mìnimo dos archivos "tar", que residen en la respectiva locaciòn "default" (que corresponde al directorio "ingres/ckp/default"). Para cada "checkpoint", INGRES crea un archivo "tar" que contiene informaciòn sobre las locaciones en las que la base de datos reside, màs un archivo "tar" asociado a cada locaciòn con imàgenes de las tablas presentes en ella.

iii. Journals:

Un journal es un registro de los cambios realizados a una base de datos desde el último "checkpoint". El "journal" se mantiene en un formato que permite al DBA rehacer o deshacer algunas de las operaciones si es necesario.

INGRES mantiene "journals" para todas las tablas sobre las que se ha habilitado el "journaling". Hay un archivo de "journal" por cada base de datos que tenga tablas con "journaling". Los archivos de "journal" son colocados en su locación "default" (que corresponde en este caso al directorio "ingres/jnl/default").

2.2.3. Sistema de recuperación de INGRES:

INGRES tiene un sistema de "journaling" que, valiéndose de los archivos de "checkpoint" y "journal" permite realizar las siguientes operaciones:

- Tomar copias "instantáneas" de la base de datos ("checkpoints") en momentos en que se sabe que está consistente.

- Rehacer la base de datos, en caso de desastre lógico o físico, desde el último "checkpoint" hasta cualquier momento antes de la falla.

- Deshacer transacciones sobre la base de datos hasta cualquier momento posterior al último "checkpoint".

- Producir pistas de auditoría, las cuales son relaciones tipo INGRES, sobre cualquier tabla que tenga "journal".

INGRES recomienda, por lo tanto, tener en dispositivos físicos diferentes los directorios de "checkpoint", "journaling" y de datos de cada base de datos.

2.2.4. "Buffer pool":

INGRES provee manejo de "buffers" privado para cada usuario, sin utilizar el "buffer pool" de UNIX. Esto se debe a que los mecanismos de reemplazo y prebúsqueda de bloques que UNIX tiene degradan en general el desempeño de INGRES, a pesar de que operan correctamente para el uso normal del sistema operacional [STON81]. Uno de los procesos que INGRES lanza para cada usuario ("backend") reserva un "buffer pool" privado y lo administra.

2.3. Estructura de procesos de INGRES.

Los diseñadores de INGRES quisieron que sus componentes fueran procesos típicos de UNIX; si así no fuera, el sistema operacional

tendría que ser alterado, perdiendo portabilidad. No existe un proceso común para todos los usuarios, sino que cada usuario tiene su propio conjunto de procesos.

2.3.1. Manejo de memoria central:

INGRES no tiene procesos comunes que actúen de "kernel" para todos los usuarios. Cuando un usuario invoca INGRES, se crean dos procesos por cada utilidad o aplicación de INGRES que éste invoque. Uno de ellos es el programa de aplicación o el paquete de INGRES, y el otro es el proceso "backend", el cual se describe en la sección 2.3.2.

2.3.2. Procesos lanzados por INGRES:

Al invocar INGRES, ya sea desde la interfaz QUEL, desde alguno de los paquetes de INGRES desde un programa en EQUEL (QUEL anidado en un lenguaje de programación convencional), se crea la siguiente estructura de procesos:

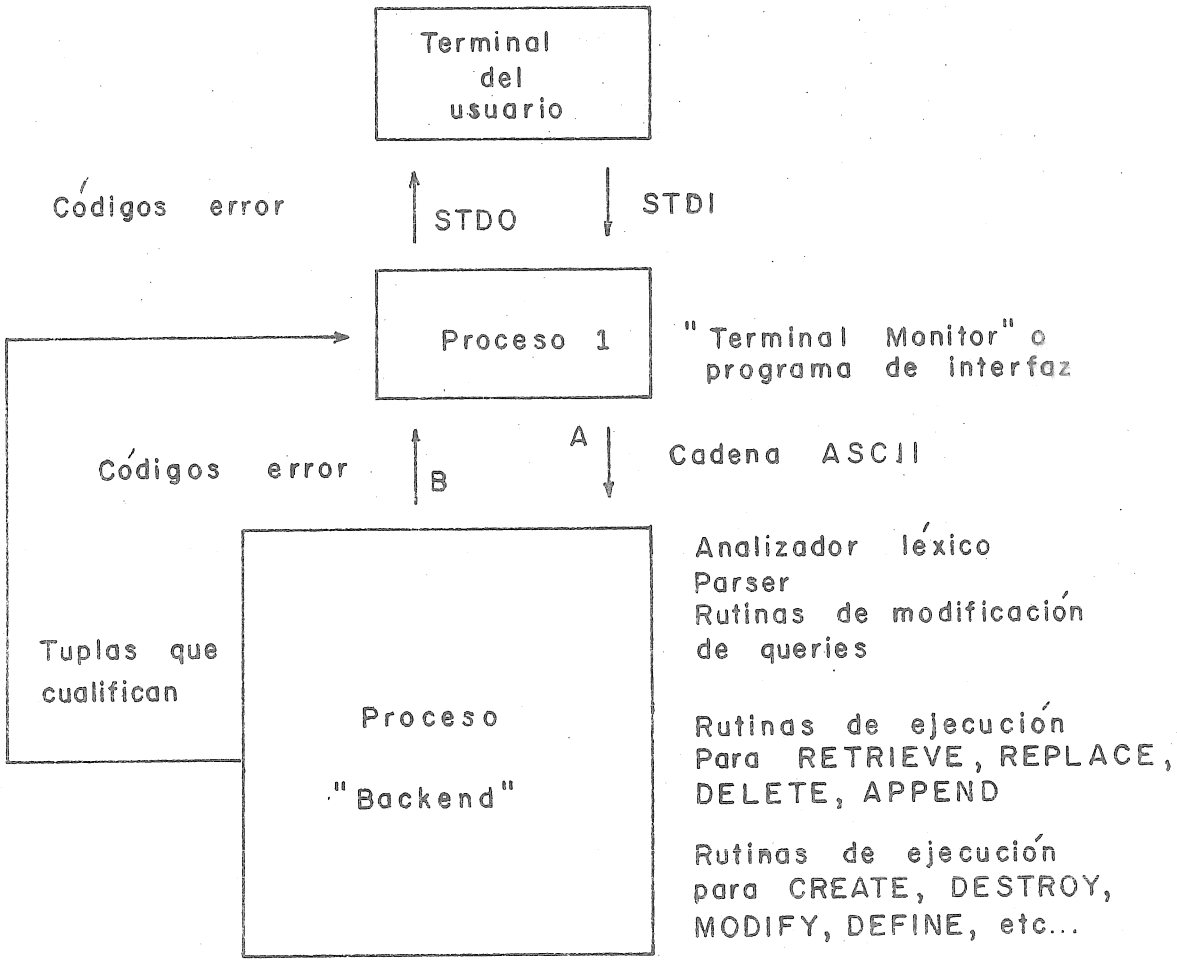

Figura 2.2

Proceso 1: Es el proceso correspondiente al programa que sirve de interfaz con la base de datos. Este puede ser:

a) El "terminal monitor", que permite al usuario editar, listar y ejecutar conjuntos de comandos de QUEL;
b) Alguno de los paquetes de interfaz proveídos por INGRES (INGRES/FORMS, INGRES/MENU, INGRES/QUERY, etc.);
c) Un programa de aplicación escrito en EQUEL (EQUEL/C, EQUEL/Pascal, EQUEL/COBOL, etc.).

Los comandos QUEL que genera el programa de interfaz son transmitidos al proceso "backend" por medio del "pipe" A, como cadenas de caracteres ASCII, en el momento de la ejecución.

Proceso "backend": Contiene un analizador lèxico, un "parser", rutinas de modificación de "queries" para control de integridad y soporte de vistas, rutinas de ejecución para los comandos RETRIEVE, REPLACE, DELETE, APPEND y para los demàs comandos de QUEL. Cualquier modificación a una relación se convierte en un comando RETRIEVE que aisla las tuplas a modificar. Copias de las tuplas revisadas se almacenan en un archivo especial (colocado temporalmente en el directorio "home" de la base de datos), que luego es procesado por un "procesador diferido de actualizaciones", también presente en este proceso.

Los còdigos de error que se generen en el proceso "backend" y las tuplas que resultan de "queries" sin relación resultado son devueltos al proceso 1 por medio del "pipe" B.

El proceso "backend" maneja además el "buffer pool" de cada usuario. INGRES coloca la tabla de "locking" para el control de concurrencia en el interior del sistema operacional.

La principal razón para esta estructura de procesos, comparada con la de versiones anteriores (que era de cuatro o màs procesos y un conjunto de "overlays"), es la del tamaño del espacio de direcciones para un proceso. En el caso de CENTIX, la memoria es virtual, alcanzando un màximo de 3,5 MB por proceso. En este espacio hay capacidad suficiente para unir todos los procesos que en versiones anteriores (INGRES/CS, por ejemplo) se repartían las tareas del proceso "backend".

2.3.3. Concurrencia y sicronización:

El control de concurrencia (en el caso de CENTIX) es llevado a cabo mediante un "device driver" especial instalado en el "kernel" del sistema operacional en el momento de la primera instalación de INGRES. Este "driver" provee niveles màs altos de "locking" que los de CENTIX, permitiendo a INGRES bloquear pàginas y tablas.

La sincronización y la comunicación entre el proceso 1 y el "backend" se hace utilizando las facilidades de UNIX, a través de "pipes".

3. ARQUITECTURA DE ORACLE SOBRE UNIX

3.1. Generalidades.

ORACLE fue implementado originalmente en un computador VAX corriendo el sistema operacional VMS. Al igual que INGRES se desarrollò en C. El SMBDR consta de dos mil mòdulos de los cuales ciento veinte tuvieron que ser modificados para la implementaciòn sobre UNIX. La primera versiòn apareciò en 1979 y con èsta la primera implementaciòn comercial de SQL. La primera versiòn sobre UNIX se instalò en un PDP-11 en 1981.

Esta seccièn describe los procesos y archivos usados por ORACLE bajo UNIX. Puesto que las implementaciones de ORACLE disponibles al escribir este artìculo, o bien no corrìan sobre UNIX o bien no estaban lo suficientemente documentadas, esta seccièn se basa primordialmente en la documentaciòn de Oracle Corporation para la versiòn 5 de ORACLE sobre un UNIX sistema V genèrico.

3.2. Estructura de archivos de ORACLE .

3.2.1. Estructura del directorio "oracle":

La figura 3.1 muestra la estructura de los directorios de ORACLE sobre UNIX. No existe una estructura rìgida de los directorios. Por medio de variables ambientales y el archivo de inicializaciòn de una base de datos, ORACLE detecta la localizaciòn de los distintos archivos.

Ya que ORACLE maneja varias bases de datos concurrentes la labor del administrador de la base de datos es importante. Se sugiere que cada tipo de archivo ("after image", "before image" y base de datos) resida en un disco fìsico diferente.

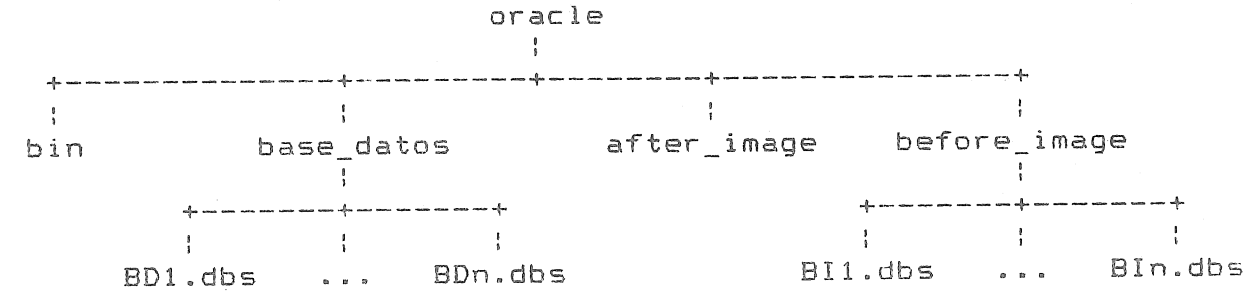

Figura 3.1

3.2.2. Tipos de archivos:

Una base de datos ORACLE consta de dos tipos de archivos: archivos de base de datos y archivos de recuperación.

i. Archivos de base de datos:

Los archivos de base de datos contienen el diccionario de datos y las tablas de los usuarios al igual que todos los índices. Desde el punto de vista de UNIX los archivos de base de datos son de tamaño estático. Se crean de un tamaño predeterminado por el administrador de la base de datos con el utilitario "Create Contiguous File" (CCF) de ORACLE. Sobre UNIX el tamaño máximo de un archivo de base de datos es una unidad física de disco. ORACLE se encarga de asignar espacio a las tablas dentro del archivo. Para direccionar e incrementar el tamaño de una base de datos, ORACLE utiliza el concepto de particiones lógicas. Una partición lógica consta de un máximo de 32 archivos de base de datos. Los archivos de una partición lógica pueden residir en cualquier "file system" (y por consiguiente en cualquier disco) y deben conformarse de bloques contiguos del disco.

Un archivo de base de datos no se puede ampliar dinámicamente. El mecanismo usado para adicionar espacio a la base de datos es crear un archivo nuevo (usando CCF) del tamaño deseado y añadírselo a una partición de la base de datos.

En la figura 3.2a se muestra la configuración de una base de datos con dos particiones y cuatro archivos distribuidos a traves de tres unidades de disco. La figura 3.2b muestra los comandos ORACLE necesarios para llegar a esta configuración. Primero se crean las particiones. Luego se crean los archivos iniciales de cada partición con el utilitario CCF dando como parámetro el número de bloques de UNIX que se quieren separar. Finalmente se hace la conexión entre cada partición y el archivo mediante el comando "alter partition". Por medio de vistas del diccionario de datos, el administrador de la base de datos puede detectar el momento en que se haga necesario crear un nuevo archivo. En tal caso se repite la anterior secuencia de comandos para los archivos /disco3/arch2pers y /disco2/arch2sist. El papel del administrador de la base de datos es muy importante ya que este estima el tamaño inicial y crecimiento de la base de datos.

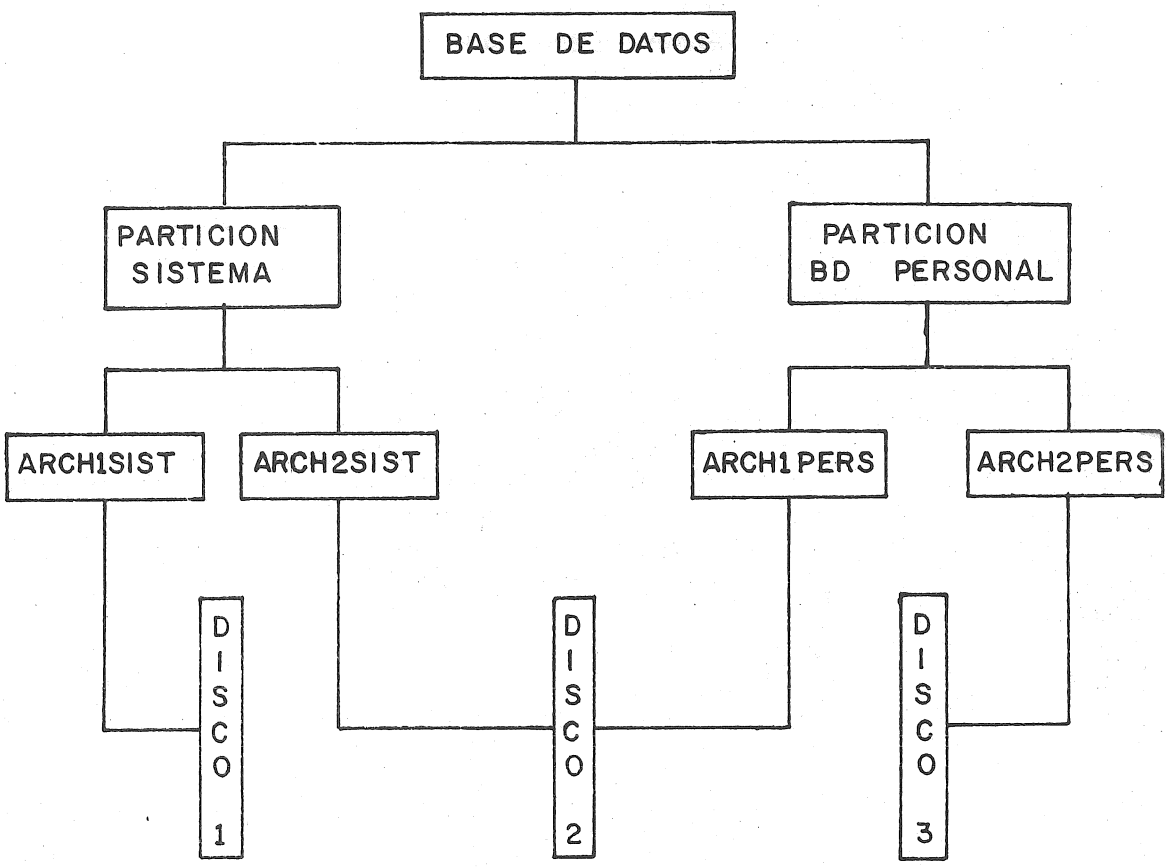

Figura 3.2a

```
CREATE PARTITION SISTEMA
CREATE PARTITION BD_PERSONAL
CCF /disco1/db/arch1sist   10000
CCF /disco2/db/arch1pers   20000
ALTER PARTITION SISTEMA ADD FILE '/disco1/db/arch1sist'
ALTER PARTITION DB_PERSONAL ADD FILE '/disco2/db/arch1pers'
CCF /disco2/db/arch2sist   15000
CCF /disco3/db/arch2pers   25000
ALTER PARTITION SISTEMA ADD FILE '/disco2/db/arch2sist'
ALTER PARTITION DB_PERSONAL ADD FILE '/disco3/db/arch2pers'
```

Figura 3.2b

ORACLE utiliza el sistema de archivos de UNIX únicamente para adquirir grandes archivos compuestos de bloques físicamente contiguos. Una vez hecha esta operación, ORACLE se encarga de asignar el espacio dentro del archivo a las tablas de la base de datos sin involucrar a UNIX en el proceso.

La figura 3.3 muestra la estructura lógica de una partición. Una partición se compone lógicamente de tablas. Una tabla se compone lógicamente de segmentos de datos y de índices. Un segmento se compone físicamente de "extents" donde un "extent" es la mínima cantidad de espacio asignada. Cada "extent" es un conjunto físico de bloques de disco. El tamaño del "extent", dado en bloques de ORACLE, lo asigna el administrador de la base de datos cuando se crea una tabla. Para la asignación de espacio a las tablas de una partición, ORACLE maneja dos listas de espacio libre. Una a nivel de partición y otra a nivel de segmento lógico.

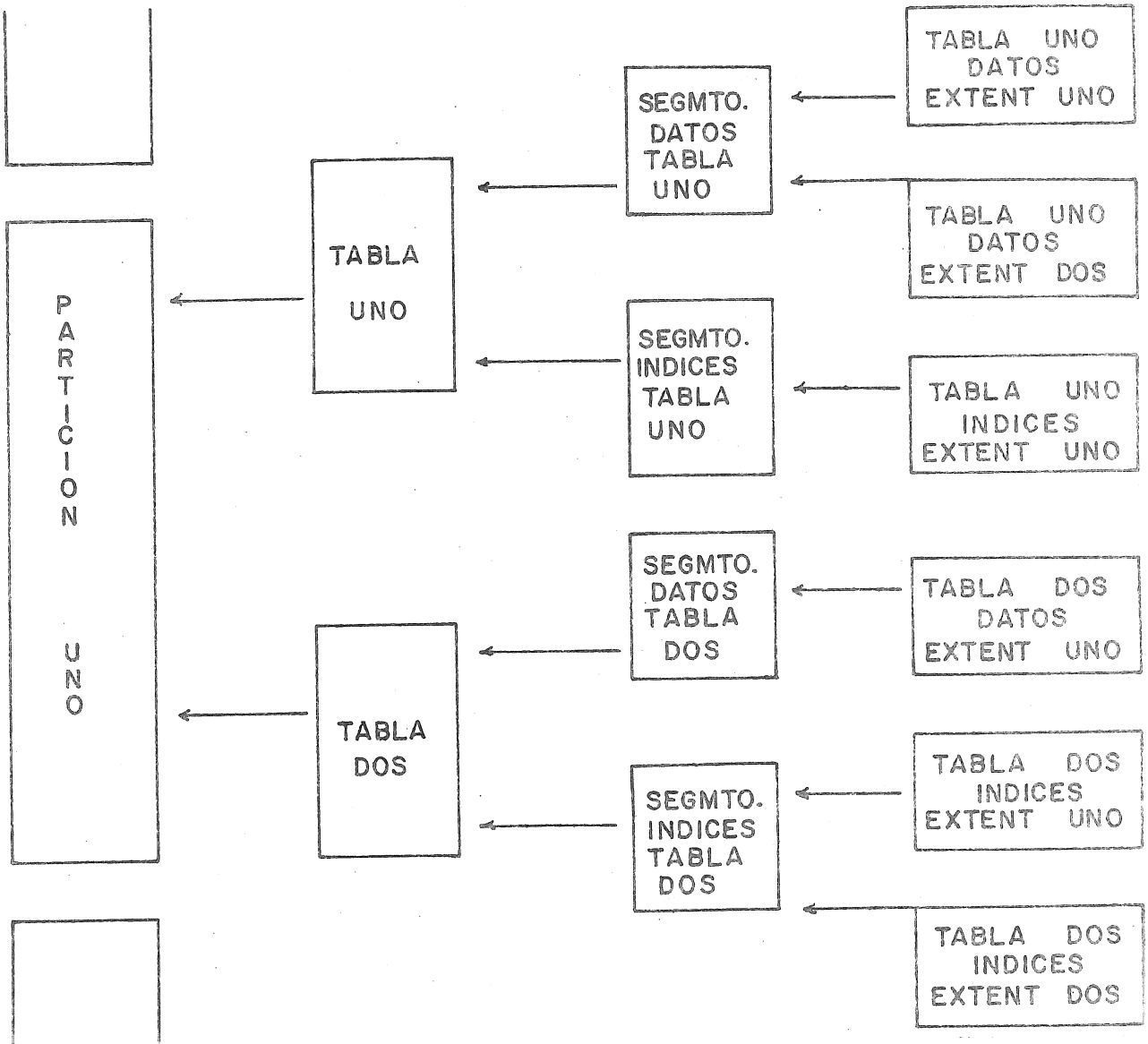

Figura 3.3

ii. Archivos de recuperación:

ORACLE maneja dos tipos de archivos de recuperación el "Before Image" (BI) y el "After Image" (AI). El BI se utiliza para recuperación de transacciones y el AI en caso de falla de una unidad física de disco.

En ORACLE una transacción (unidad lógica de trabajo) tiene tres estados: activa, finalizada o abortada. Cuando se finaliza una transacción las modificaciones que esta haya hecho a la base de datos se vuelven permanentes. Cuando una transacción aborta los cambios parciales que haya hecho se pueden deshacer por medio del comando "rollback". Mientras que una transacción no haya finalizado o abortado su estado se define como activo.

a. Archivo Before Image:

Al igual que los archivos de la base de datos, el archivo BI es un archivo de tamaño estático creado con CCF. Su función principal es permitir el "rollback". El "before image" tiene copias de los bloques de la base de datos antes de estos ser modificados. En el caso de que se ejecute un "rollback" los datos de las tablas afectadas en el "before image" se reescriben al archivo de la base de datos.

El "before image" también cumple la función de controlar lecturas concurrentes. Si un usuario está actualizando mientras otros están leyendo, los lectores leen la copia del "before image". El BI está implementado como un archivo circular de modo que la transacción más vieja es substituida por la transacción corriente cuando el archivo se llena. Se recomienda que el "before image" esté en un disco diferente a la base de datos para minimizar la contención de accesos.

b. Archivo After Image:

El archivo "after image" tiene como única función permitir la recuperación de una base de datos en caso de falla del disco en que se almacena dicha base de datos. Por consiguiente el "after image" debe residir en una unidad de disco diferente a la base de datos. En el "after image" se lleva un registro de todas las escrituras a los archivos de la base de datos y puede ser habilitado o deshabilitado a nivel de base de datos por el administrador. Para la recuperación el archivo de "after image" se aplica a una copia de respaldo de la base de datos bajo control del utilitario AIJ. El resultado es una copia exacta de la base de datos perdida. Este procedimiento se llama "roll-forward". Al habilitar "after image journaling" cada transacción genera una escritura adicional al disco lo cual degrada el desempeño del sistema aproximadamente en un 10% [ORAC86].

3.2.3 Manejo del "buffer pool" de UNIX

Una base de datos ORACLE puede almacenarse en disco bajo dos modalidades: en el "file system" de UNIX o en un "raw device". Un raw device es un disco o sección de un disco que ha sido configurado para no ser controlado por UNIX.

El uso de raw devices conlleva a que:

- La entrada/salida al disco no utiliza el "buffer pool" de UNIX. Las lecturas y escrituras se hacen directamente de los "buffers" de ORACLE (descritos en la sección 3.3) al disco.

- Se evita el "overhead" de UNIX el cual lee varios bloques del disco por cada pedido de lectura.

- Baja la carga del "file system" de UNIX.

- Los "buffer pools" de UNIX y de ORACLE se graduan independientemente.

- El desempeño del sistema mejora de 10 a 20% [ORAC86].

Cuando los archivos de la base de datos se almacenan en el "file system" de UNIX, ORACLE maneja el "buffer pool" de UNIX con una facilidad no documentada de UNIX sistema V [ORAC86a]: el "cache write-thru". Este permite que ORACLE pueda forzar la escritura de un bloque al disco por medio del "buffer pool" de UNIX y asi garantizar la integridad de la base de datos.

3.3. Estructura de procesos de ORACLE .

3.3.1 Manejo de memoria principal:

ORACLE utiliza el "System Global Area" (SGA) para implementar el manejo de memoria compartida. Se utilizan las facilidades de memoria compartida de UNIX sistema V para que varios procesos puedan accesar los mismos datos. Ya que UNIX 4.X de Berkeley no implementa memoria compartida, ORACLE no corre bajo estas versiones de UNIX. EL SGA contiene los "buffers" de lectura y escritura al igual que las tablas de "locks". UNIX tiene tres parámetros que interactúan con el manejo de memoria de ORACLE los cuales se asignan cuando se genera el sistema.

- SHMMAX - El máximo número en bytes de un segmento de memoria compartida.

- SHMSEG - El número máximo de segmentos de memoria compartida que puede accesar un proceso.

- SHMALL - El número máximo de páginas de memoria compartida que soporta el sistema.

Para optimizar el manejo de memoria compartida se recomienda que SHMMAX sea mínimo 131K Bytes y que SHMALL no sea más de 25% de la memoria real de la máquina para evitar "swapping" excesivo de los procesos. ORACLE impone un límite al tamaño del SGA de 10 Mbytes (10 segmentos de 1Mbyte cada uno).

3.3.2. Procesos lanzados por ORACLE:

ORACLE maneja cuatro procesos compartidos para lectura y escritura y dos adicionales por cada usuario bajo UNIX. La figura 3.4 muestra la estructura de los procesos y su relación con el SGA.

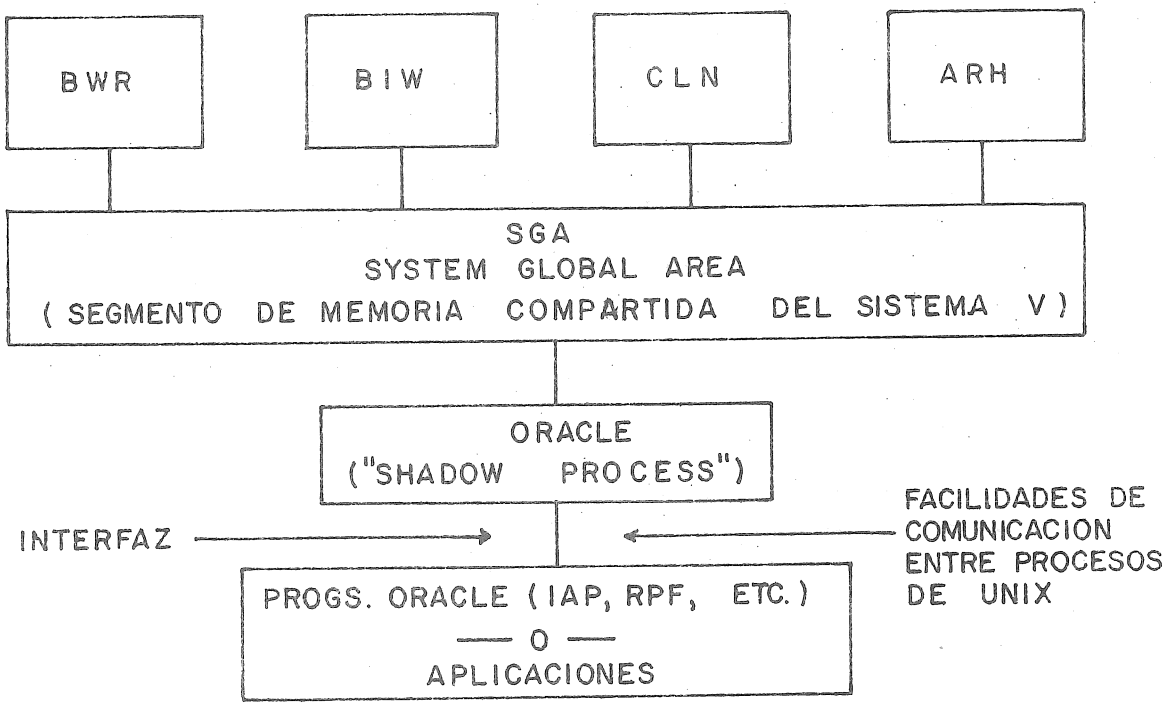

Figura 3.4

Por cada usuario que se conecta al "kernel" de ORACLE se crea un proceso principal y uno que corre en "background". Los utilitarios de ORACLE o aplicaciones hechas por el usuario conforman el proceso principal. El segundo proceso ("shadow process") es una interfaz que se encarga de filtrar los pedidos del proceso principal para evitar la corrupción del area de memoria compartida.

ORACLE usa código reentrante. Cada usuario utiliza el mismo segmento de código cuando hace llamados a las rutinas internas de ORACLE. El apuntador al "stack" de cada proceso mantiene la posición de éste en el código.

Los siguientes son los procesos que hacen interfaz con los archivos de la base de datos.

IOR - Initialize ORACLE

Lanza los cuatro procesos descritos a continuación e inicializa el SGA de acuerdo con los parámetros dados por el administrador de la base de datos. Utilizado también para crear nuevas bases de datos y bajar ("shutdown") una base de datos.

ARH - Asynchronous Read Ahead

Copia bloques del archivo de la base de datos al (SGA). Para mejorar el desempeño de la base de datos lee el bloque que pidió el programa al igual que los bloques vecinos. La lectura se hace en paralelo con la ejecución del programa cliente.

BIW - Before Image Writer

Copia bloques del SGA al archivo BI ("Before Image"). Es el único proceso que tiene acceso de escritura sobre el archivo BI. Los bloques copiados se usan para hacer recuperación en caso de transacciones abortadas ("rollback") y para asegurar la consistencia de los datos leidos por un proceso mientras que otro actualiza los mismos datos.

BWR - Buffer Writer

Escribe los bloques que han sido modificados en el SGA a la base de datos y al archivo AI ("After Image"). Es el único proceso que puede escribir a estos archivos. Los archivos se escriben cuando una transacción termina exitosamente.

CLN - Cleanup

Identifica periódicamente los procesos que han terminado anormalmente y hace la recolección de basura. Tomando la identidad de los procesos abortados, borra las tablas temporales, deshace las transacciones incompletas, libera los "buffers" y finalmente mata el proceso.

3.3.3. Concurrencia y sincronización:

Para manejo de concurrencia y sincronización ORACLE utiliza las facilidadaes de comunicación entre procesos de UNIX. Los procesos se comunican por medio de memoria compartida, señales y semáforos. Las señales se utilizan para informar a los procesos en què dirección de la memoria compartida se encuentran sus datos. Los semáforos se utilizan para controlar la concurrencia en los accesos a la memoria compartida. ORACLE versión 5 no utiliza "pipes". Aunque los "pipes" implementan las funciones de memoria compartida y señales, ORACLE considera que èstos son lentos y poco confiables [ORAC86].

4. CONCLUSIONES

INGRES y ORACLE tienen la misma arquitectura relacional (ver figura 4.1), en la cual los comandos de definición y manipulación de datos pertenecen al mismo lenguaje y pueden ser mezclados libremente [SCHM83].

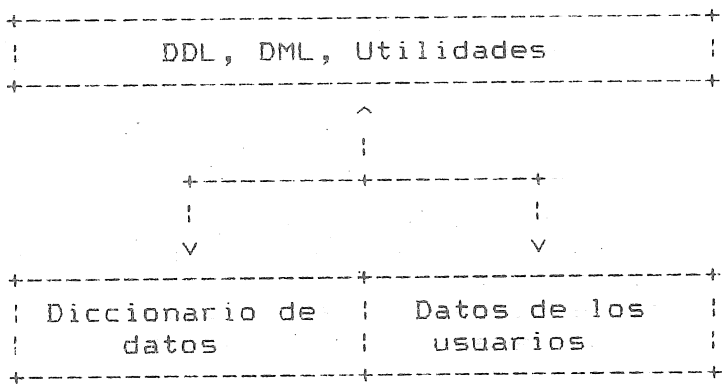

Figura 4.1

Sin embargo, sus arquitecturas físicas, tanto en archivos como en procesos son notablemente diferentes:

1. Archivos: La disparidad entre INGRES y ORACLE en la representación física de las bases de datos puede deberse a los diferentes conceptos de portabilidad establecidos por los creadores. El equipo que diseñó INGRES quería que su SMBDR fuera transportable a cualquier máquina UNIX, por lo cual podía apoyarse en su "file system". Los diseñadores de ORACLE, por su parte, buscaban portabilidad independiente del sistema operacional. De allí que hayan decidido implementar su propio sistema de administración de espacio en disco.

   Cada relación de INGRES corresponde a un archivo UNIX. El tamaño de este archivo varía de acuerdo al tamaño y la estructura interna de la tabla correspondiente, bajo el control de UNIX. ORACLE requiere que el DBA reserve explícitamente archivos UNIX de tamaño fijo en los cuales se almacenan conjuntos de tablas. El tamaño de los archivos UNIX permanece invariable. ORACLE se encarga de la administración del espacio dentro de estos archivos. UNIX no puede hacer uso del espacio libre de ORACLE. Por lo tanto es importante que el DBA de ORACLE planee cuidadosamente el uso del disco, para no reservar espacio que no utilizará la base de datos y que tampoco podrá utilizar UNIX.

2. Procesos: INGRES lanza procesos para cada usuario (incluyendo el "buffer pool"). No hay procesos que actúen como un núcleo común para todos los usuarios. ORACLE tiene cuatro procesos que se encargan de la entrada/salida del disco de todos los usuarios.

Tanto INGRES como ORACLE han escogido manejar su propio "buffer pool" para mejorar el desempeño del SMBDR, en lugar de utilizar el "buffer pool" de UNIX. Sin embargo, sobre el Sistema V de UNIX, ORACLE deja la opción de utilizar los "buffers" de UNIX con la facilidad "cache write-thru" para mantener la integridad de la base de datos, o evitarlos del todo por medio de "raw devices".

Uno de los principales objetivos de los diseñadores de INGRES y ORACLE fuè garantizar la portabilidad de sus SMBDRs. En las implementaciones estudiadas se distinguieron tres niveles de portabilidad:

1. Portabilidad de UNIX: Por estar implementados sobre UNIX, INGRES y ORACLE disfrutan del amplio rango de equipos y marcas que soportan este sistema operacional.

2. Portabilidad del SMBDR: Existen versiones de ambos SMBDRs sobre una gran variedad de sistemas operacionales. INGRES presenta mayor dependencia de UNIX que ORACLE, porque los diseñadores de INGRES consideraron suficiente desde un comienzo el nivel de portabilidad de UNIX y querían aprovechar a fondo su ambiente de desarrollo [STON80]. En particular, la arquitectura de archivos de INGRES se basa en el "file system" de UNIX.

3. Portabilidad del modelo relacional: De los tres principales modelos de bases de datos (red, jerárquico y relacional), el modelo relacional es el que màs alto nivel de independencia de datos provee, a la vez que permite una forma natural de visualizarlos [CHEN76]. Estas dos características garantizan en todo momento transportabilidad de diseño y de datos. De diseño a través de distintas implementaciones del modelo relacional, y de datos a través de arquitecturas diferentes de "hardware" y "software".

Por último, la tendencia de la industria del "software" es ceñirse a estándares. En cuanto a sistemas operacionales, UNIX, y en cuanto a modelos de bases de datos, el modelo relacional. INGRES y ORACLE, además de tener fuertes nexos con UNIX, han constituido la fuerza impulsora de la estandarización del modelo relacional.

REFERENCIAS Y BIBLIOGRAFIA

[BACH86] Bach, M. "The Design of the UNIX Operating System". Prentice-Hall, 1986.

[BURR86a] "XE 500 CENTIX System INGRES: Installation and Implementation Guide". Burroughs Corporation, 1986.

[BURR86b] "XE 500 CENTIX System INGRES: Application Development Guide". Burroughs Corporation, 1986.

[BURR86c] "XE550 INGRES Burroughs Multi-AP Release, version 3.0.7". Burroughs Corporation, 1986.

[CHEN76] Chen, P. P. "The Entity-Relationship Model - Toward a Unified View of Data". ACM TODS, Vol. 1, No. 1, 1976.

[CODD79] Codd, E.F. "Extending the Relational Data Base Model to Capture More Meaning". ACM TODS, Vol. 4, No. 4, 1979.

[CODD83] Codd, E.F. Introducción al libro "Relational Database Systems", editado por Joachim W. Schmidt y Michael L. Brodie. Springer-Verlag, 1983.

[ORAC86a] "ORACLE for UNIX General Information Manual Version 5". ORACLE Corporation, 1986.

[ORAC86b] "ORACLE Data Base Administrator's Guide Version 5 for AOS/VS". ORACLE Corporation, 1986.

[ORAC86c] "ORACLE Tricks of the Trade". ORACLE Corporation, 1986.

[ORCE85] "SQL, The quiet revolution". ORCE Systems Software, 1985.

[SCHM83] Schmidt, J.W. et al. "Relational Database Systems - Analysis and Comparison". Springer-Verlag, 1983.

[STON76] Stonebraker, M. et al. "The Design and Implementation of INGRES". ACM TODS, Vol. 1, No. 3, 1976.

[STON80] Stonebraker, M. "Retrospection on a Database System". ACM TODS, Vol. 5, No. 2, 1980.

[STON81] Stonebraker, M. "Operating System Support for Database Management". Communications of the ACM, Vol. 24, No. 7, 1981.