CAPITULO I

CONCEPTOS DE SISTEMAS DE BASES DE DATOS

1.1 Conceptos básicos

Base de Datos (BD): Conjunto de datos relacionados, cuyo acceso es a través de un SABD.

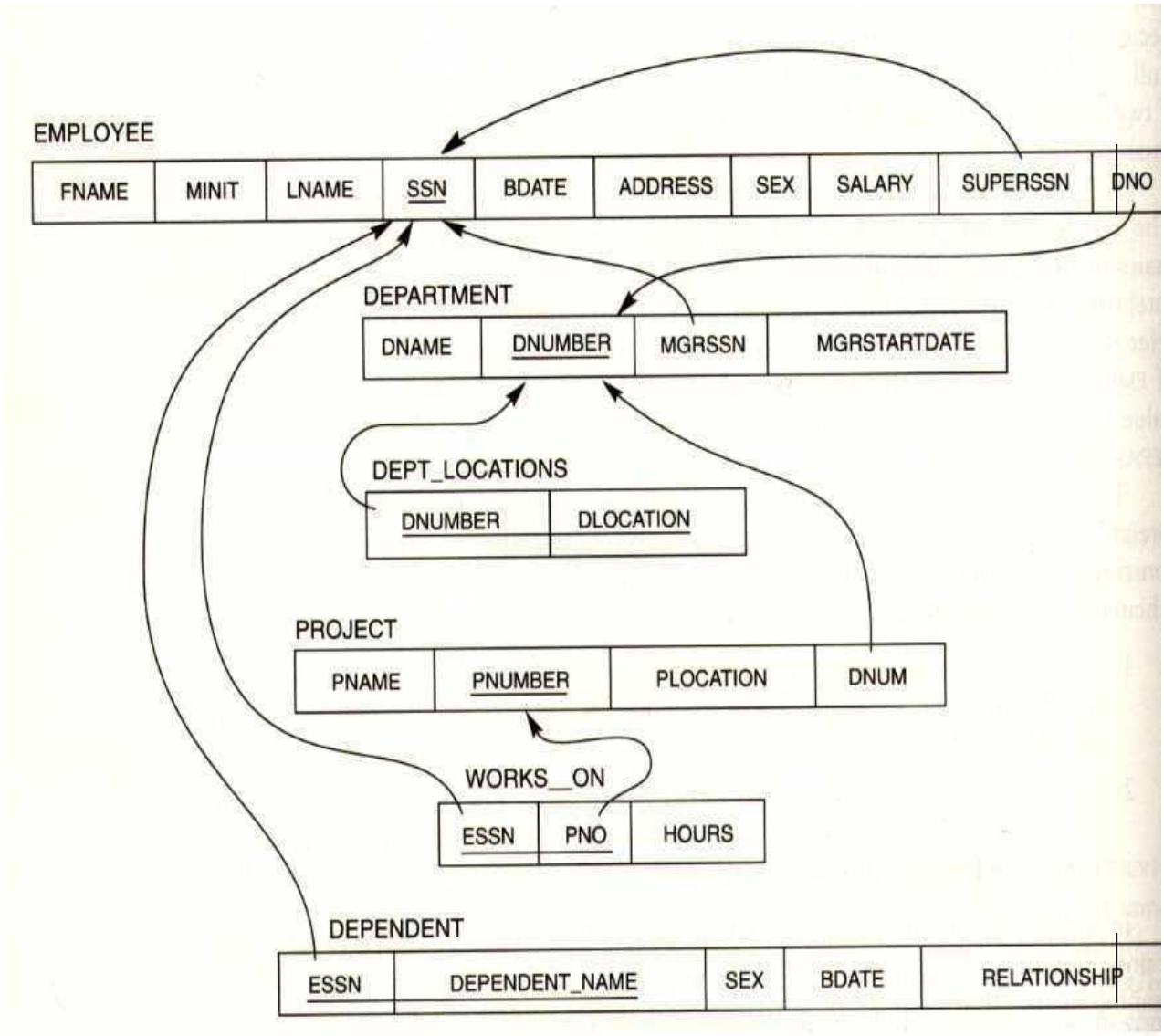
Dato: Es un hecho conocido, que puede ser registrado, y que tiene un significado implícito.

Una base de datos tiene dos **propiedades** importantes:

- **Integrada:** Los archivos de datos que la componen han sido organizados lógicamente para reducir la redundancia y facilitar el acceso a los datos
- **Compartida:** Todos los usuarios calificados tienen acceso a los mismos datos

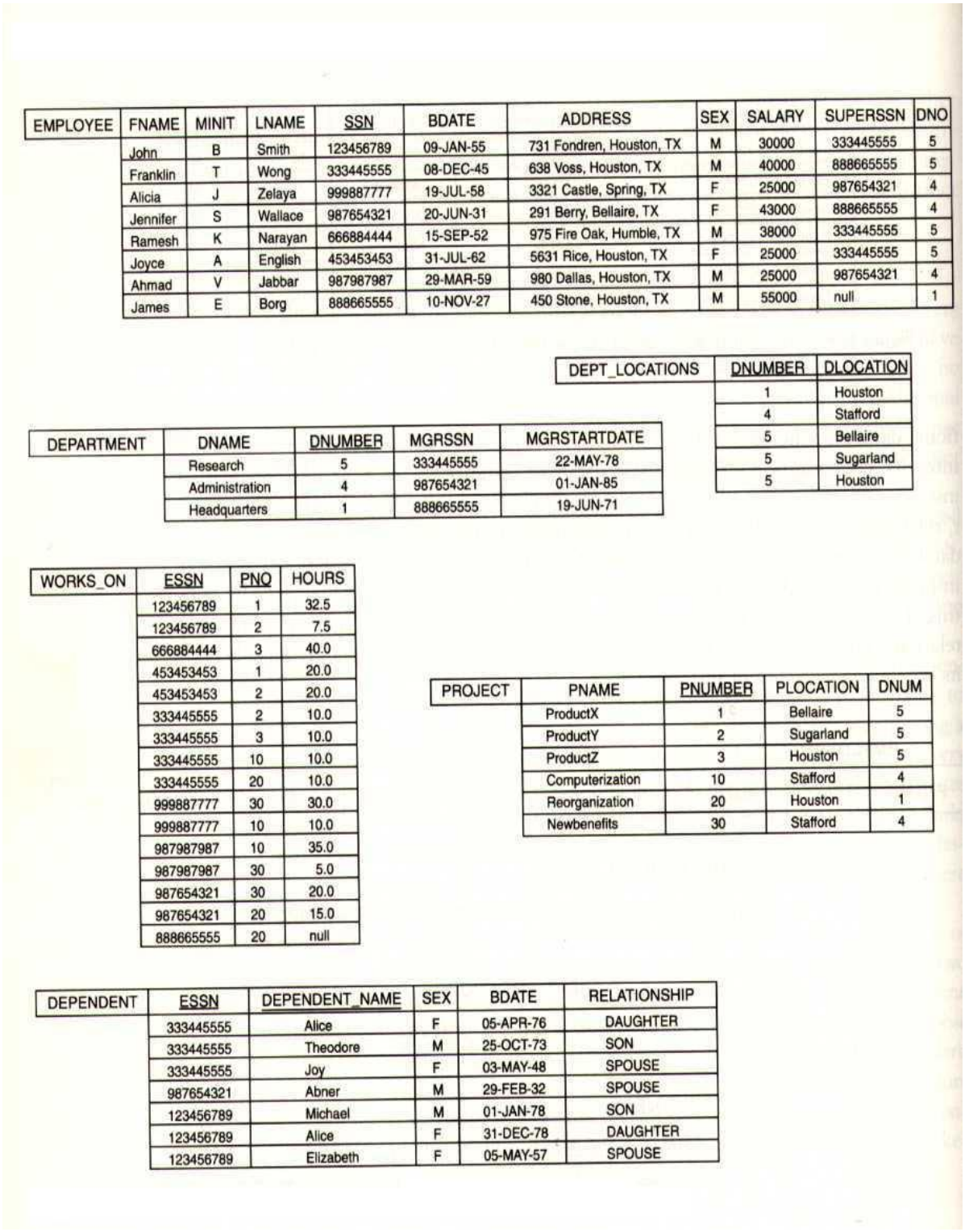
Componen la base de datos:

- **Esquema:** Es la descripción de la base de datos (metadatos)
- **Instancia de la base de datos:** Son los datos almacenados en la base de datos, en un instante particular en el tiempo



Ejemplo de esquema de una base de datos

METADATO



EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	09-JAN-55	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	08-DEC-45	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	19-JUL-58	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	20-JUN-31	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	15-SEP-52	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	31-JUL-62	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	29-MAR-59	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	10-NOV-27	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	22-MAY-78
Administration	4	987654321	01-JAN-85
Headquarters	1	888665555	19-JUN-71

WORKS_ON

ESSN	PNO	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	null

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

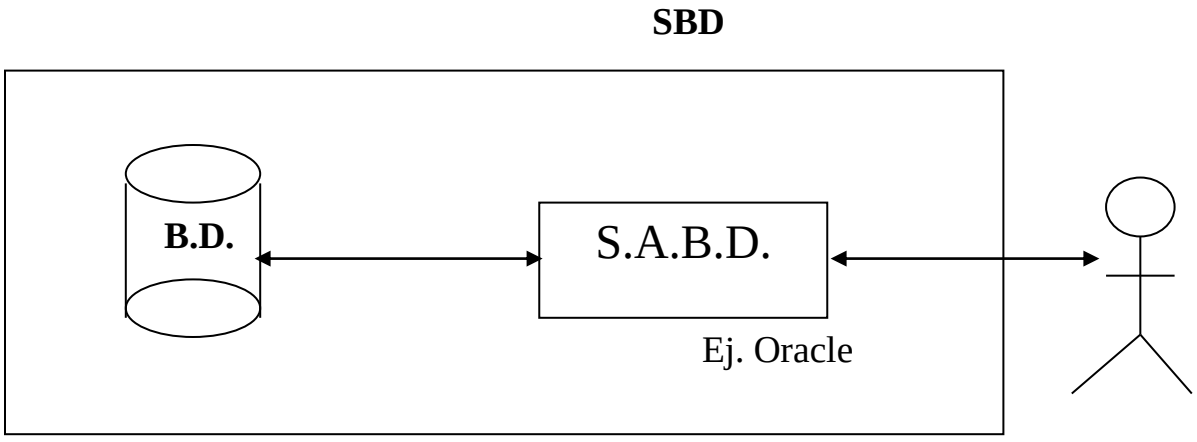
DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
333445555	Alice	F	05-APR-76	DAUGHTER
333445555	Theodore	M	25-OCT-73	SON
333445555	Joy	F	03-MAY-48	SPOUSE
987654321	Abner	M	29-FEB-32	SPOUSE
123456789	Michael	M	01-JAN-78	SON
123456789	Alice	F	31-DEC-78	DAUGHTER
123456789	Elizabeth	F	05-MAY-57	SPOUSE

Ejemplo de **instancia** de una base de datos

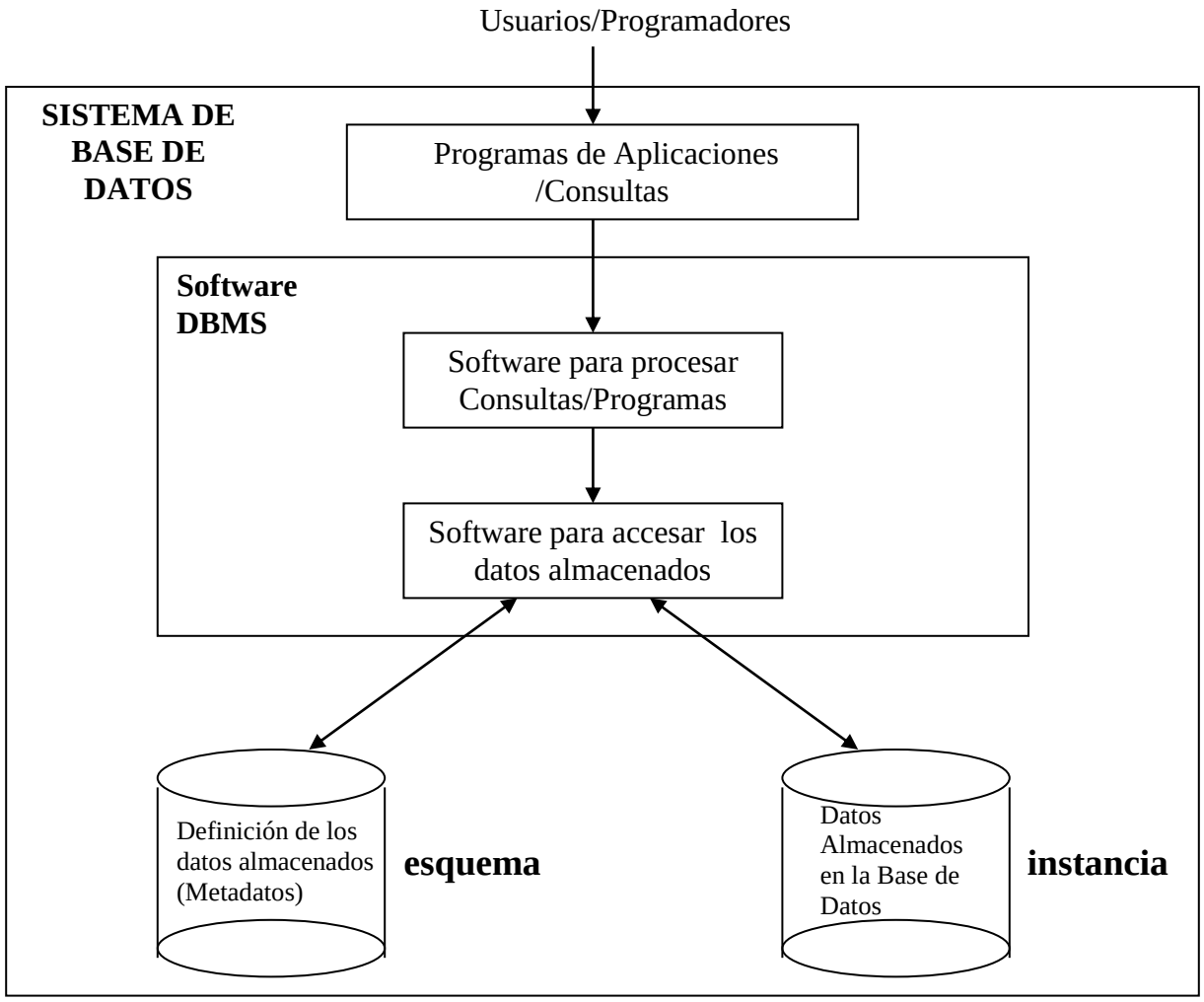
Sistemas de administración de bases de datos (SABD): Es un ambiente de programación orientado al manejo de grandes volúmenes de datos con propiedad de persistencia. Es un conjunto de programas que permiten a los usuarios crear y mantener una base de datos. Ejemplo: Oracle.

Sistemas de base de datos (SBD): SBD = BD + SABD



Sistema de base de datos

SABD = SGBD = DBMS



Un ambiente de sistema de bases de datos simplificado

1.2 Sistemas de procesamiento de archivos

- Necesidades de procesamiento de datos de departamentos individuales en la organización
- Se desarrollan nuevos programas, normalmente uno a la vez, para aplicaciones individuales, tales como remuneraciones, control de inventarios, etc.
- No hay un plan general que facilite el crecimiento de las aplicaciones
- Cada nueva aplicación es típicamente diseñada con su propio conjunto de archivos de datos. Muchos de los datos en estos nuevos archivos ya se encuentran presentes en archivos ya existentes para otras aplicaciones
- Cada programa de aplicación "posee" sus propios archivos de datos, y la lógica del programa es dependiente de los formatos y descripción de los datos.

Sus **desventajas** principales son:

a) Redundancia incontrolada: Cada aplicación tiene sus propios archivos

b) Datos inconsistentes: Producto de la redundancia se puede producir inconsistencia en los datos

c) Inflexibilidad: Si hay nuevos requerimientos, implica mucho trabajo

d) Compartición de datos es limitada: cada aplicación tiene sus propios datos y normalmente no se comparten

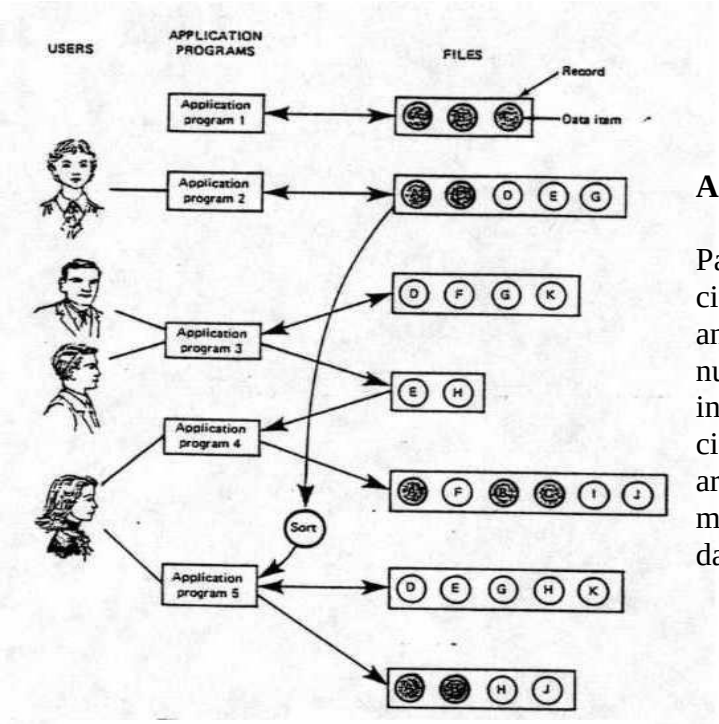
e) No hay tendencia a estándares

f) Baja productividad de los programadores: Cada definición de datos debía codificarse en el programa

g) Costo excesivo de mantención de programas: Cualquier cambio en el archivo implica cambios en el programa

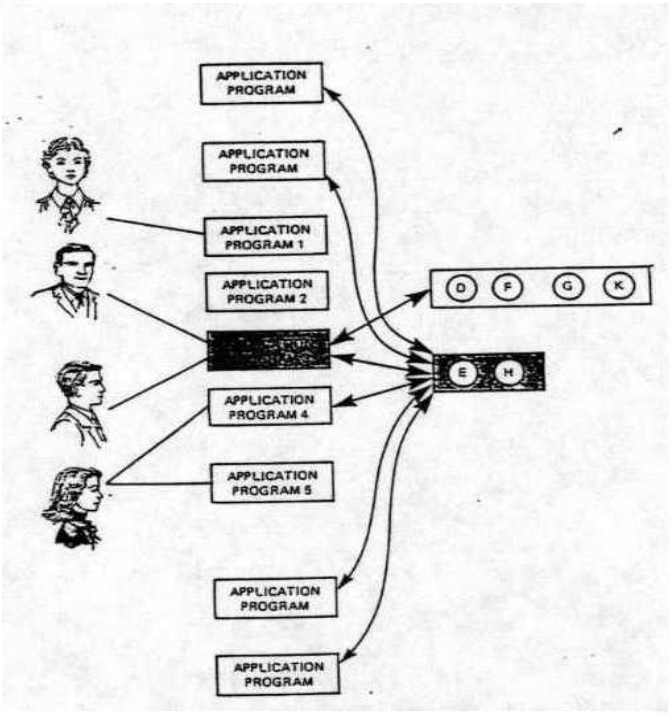
Existen paquetes de software que ayudan a evitar (o a minimizar) algunas de las desventajas mencionadas anteriormente, pero permanecen las dificultades principales de los sistemas de procesamiento de archivos:

- Redundancia de datos
- Baja compartición de datos
- Carencia de estándares
- Baja productividad



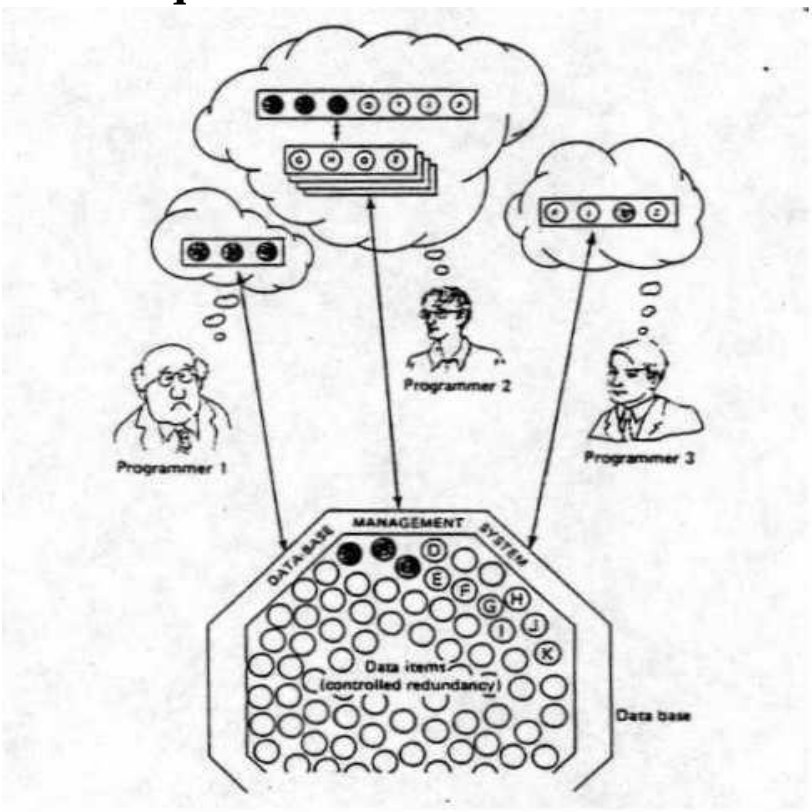
Ambiente de Archivo

Para cada nueva aplicación un programador o analista debe crear un nuevo archivo. Una instalación grande tiene cientos o miles de tales archivos, lo que genera mucha redundancia de datos.

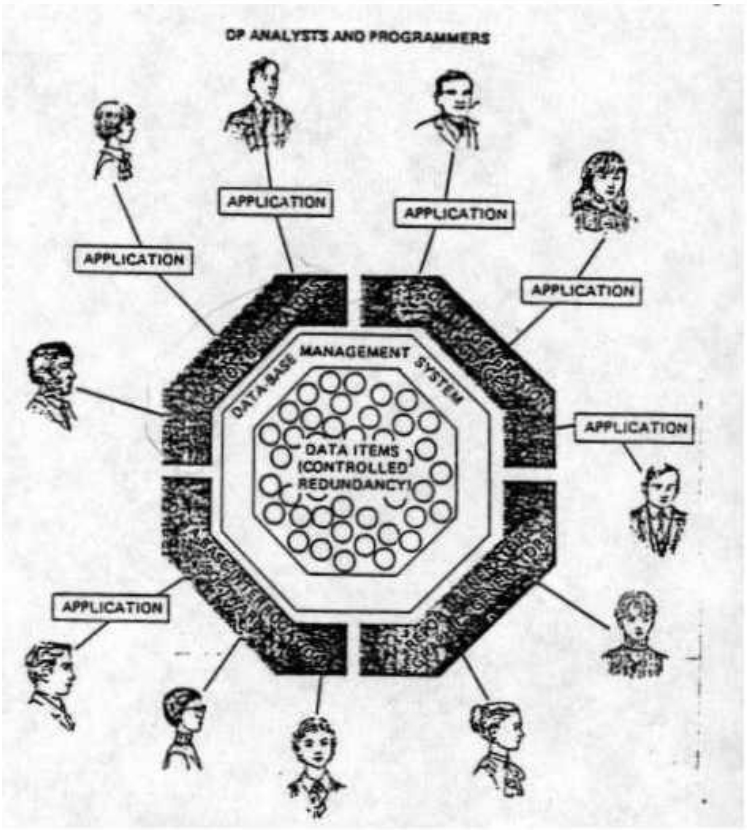


Un archivo puede ser usado por muchas aplicaciones. Cuando cambia una aplicación (programa de aplicación 3) y su archivo tiene que ser reestructurado, todos los programas que usan este archivo tienen que ser cambiados.
Aparentemente un cambio trivial en un ambiente de archivos, provoca una reacción en cadena de otros cambios que deberían hacerse.

1.3 Ventajas/Desventajas del uso del SABD frente al Enfoque de archivos



Con los sistemas administradores de bases de datos, los usuarios desconocen la estructura real de los datos almacenados.



Los lenguajes de alto nivel de bases de datos agilizan el proceso de desarrollo de aplicaciones y alivian las tareas de mantenimiento.

Sistemas administradores de bases de datos

Ventajas

1. Los datos se comparten (la información es un recurso de la organización)
2. Control de la redundancia de la información
3. Control de la consistencia en la información
4. Se mejoran los estándares con respecto a los datos
5. Mejora la seguridad de los datos (control de accesos)
6. Mejora la integridad de los datos (restricciones de integridad). Por ejemplo, un jefe no puede ganar menos que sus subordinados
7. Permite balancear requerimientos en conflicto
8. Mejora el tiempo de desarrollo de nuevas aplicaciones
9. Mejora la accesibilidad a los datos (“query languages”)
10. Economía de escala (producto de la asignación centralizada de los recursos)
11. Mejor control de acceso concurrente a los datos
12. Se dispone de procedimientos de respaldo (“Back – Up”) y recuperación (caídas del sistema y fallas del hardware)

Desventajas

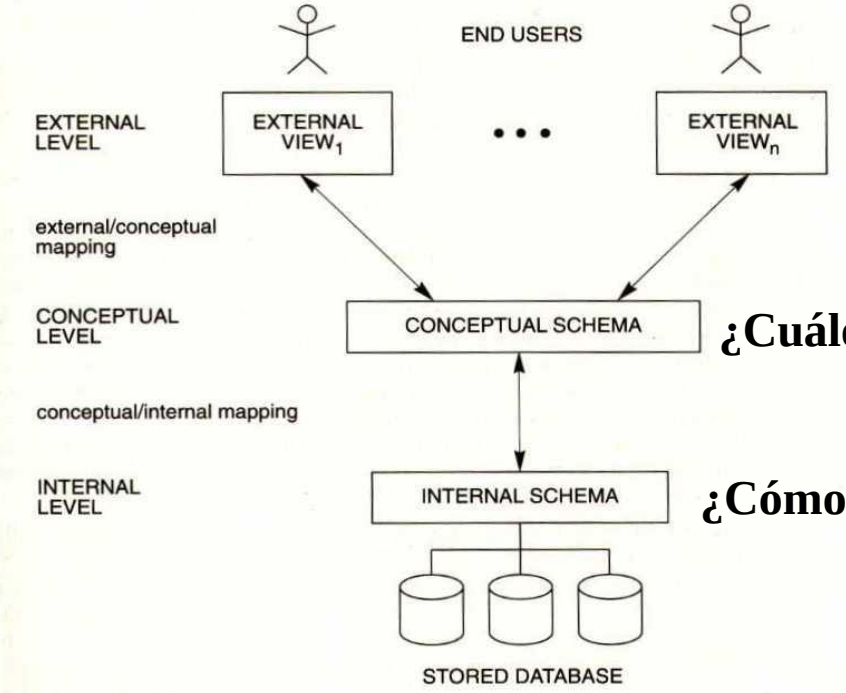
1. Alto costo de un DBMS
2. Mayor costo de hardware
3. Mayor costo de programación
4. Costo de conversión es alto
5. Procesamiento más lento de algunas aplicaciones
6. Se incrementa la vulnerabilidad (los recursos son centralizados)
7. Mayor dificultad en la recuperación (se debe determinar qué transacciones fueron completadas, cuáles no, etc)

1.4 Funciones críticas en un SABD

a) Soporte para atender múltiples usuarios en forma **concurrente** (compartir los datos)
b) **Seguridad** de los datos almacenados en la base de datos: **Autorización** (Control d^ accesos)
c) **Recuperación** de caídas del sistema
d) **Eficiencia** en el acceso a los datos
e) **Integridad** de los datos

1.5 Arquitectura de 3 esquemas para un SABD: Niveles de abstracción

Tiene como objetivo separar al usuario de las aplicaciones, de la base de datos física.



¿Cuáles son los datos?

¿Cómo se almacenan?

Figure 2.2 The three-schema architecture.

Arquitectura de 3 esquemas

Niveles (o vistas) de los datos (Niveles de abstracción)

- La mayoría de los SABD permiten que la base de datos de la organización sea vista en tres niveles de abstracción.
- Estos **tres niveles** son:
  - Nivel conceptual
  - Nivel interno
  - Múltiples modelos externos (views)

Nivel Interno

- Tiene un esquema interno, que describe la estructura de almacenamiento físico de la base de datos.
- El esquema interno utiliza un modelo de datos físico y describe todos los detalles del almacenamiento de datos y las rutas de acceso a la base de datos.

Nivel Conceptual

- Tiene un esquema conceptual que describe la estructura de toda la base de datos para una comunidad de usuarios
- Define el mundo real en un nivel abstracto, en un modelo de datos de alto nivel

- Se concentra en describir las entidades, los tipos de datos, las relaciones, las operaciones de los usuarios y las restriccciones
- El esquema conceptual oculta los detalles de las estructuras de almacenamiento físico (no contiene detalles de implementación).

Nivel de vista o externo

- Incluye una cierta cantidad de esquemas externos o vistas de usuario.
- Un esquema externo describe la parte de la base de datos en la que un grupo de usuarios en partyicular está interesado y le oculta el resto de la base de datos
- Cada vista del usuario es un subconjunto del modelo conceptual, definido y formateado de acuerdo a las necesidades del usuario.

El SABD provee los mecanismos para transformar de un nivel de abstracción a otro, intentando mantener una relativa independencia entre un nivel y otro.

Resumen del Modelo ANSI/SPARC:

- Esquema Externo
- Esquema Conceptual
- Esquema Interno

El esquema externo representa las visiones de los usuarios (como la realidad “es vista”).

El esquema conceptual representa la abstracción de “como la realidad es”.

El esquema interno se refiere a como esa realidad es representada en un computador.

Ejemplo:

- **Modelo Conceptual o Schema**

  Empleado (nombre, dirección, teléfono, depto, sueldo)

- **Esquema Externo**

  Subschema1: E1 (nombre, dirección, teléfono)

  Subschema2: E2 (nombre, depto, sueldo)

- **Esquema Interno**
  - Archivo Empleados

    Nombre : char [20]

    Dirección : char [40]

    Teléfono : char [10]

    Depto : char [15]

    Sueldo : REAL
  - Archivo índices por nombre
  - Archivo índices por depto

Indepencencia entre datos y programas

Independencia física

Es posible hacer cambios en la base de datos física (esquema interno) sin que haya que cambiar el esquema conceptual, por lo tanto tampoco es necesario cambiar los esquemas externos

Por ejemplo:

- Suministrar una ruta de acceso para mejorar la velocidad de recuperación ciertos registros.
- Reorganizar algunos archivos físicos.
- Cambiar la organización de un archivo de secuencial indexado a directo

Permite que el DBA afine la base de datos física, dejando que los programas de aplicación corran como si ningún cambio se hubiese efectuado.

Independencia lógica

Es posible modificar la base de datos conceptual sin modificar los subschemas o los programas de aplicación.

Por ejemplo:

- Agreagando un atributo atributos
- Una tabla se separa en dos para mejorar el rendimiento de ciertas consultas,
- Agregar otro tipo de entidades a la base de datos.

Permite que la base de datos cambie y evoluciones sin afectar las vistas o programas de los usuarios.

1.6 Lenguaje de base de datos en un SABD relacional

Considera dos aspectos:

- DDL: Lenguaje de definición de ddatos
- DML: Lenguaje de manipulación de datos

DML: Data Manipulation Language.
Lenguaje en el cual se especifican las operaciones de consulta o actualización de la base de datos.

Ejemplo de Consulta o Query: Listar todos los clientes de la II región que hayan comprado más de $100.000 durante el último año.

Usualmente, hay dos formas de accesar la base de datos:

a) **Lenguaje de base de datos autocontenido (Self-Contained Database Language)**
   SQL
b) **Lenguaje de base de datos inserto (Embedded Database Language) en un Lenguaje Huésped (anfitrión)**
   Ejemplo: C + SQL

SQL (STRUCTURED QUERY LANGUAGE)

a) **Lenguaje de Definicion de Datos (DDL):** Permite definir o eliminar objetos de la base de datos tales como tablas o vistas.

   Ejemplo: Alter, Create, Drop, Grant, Rename, Revoke

b) **Lenguaje de Manipulacion de Datos (DML):** Permite consultar y actualizar los datos existentes en la base de datos.

   Ejemplo: Select, Insert, Delete, Update

c) **Lenguaje de Control de Datos (DCL):** Permite controlar los cambios a los datos y a la base de datos.

   Ejemplo: Commit, Rollback, Savepoint (Transacción)

d) **Lenguaje inserto** (Embedded Language) (“Cursor”)

   Lenguaje de Base de Datos + Lenguaje de Programación

   SQL → **JAVA**

   SQLJ

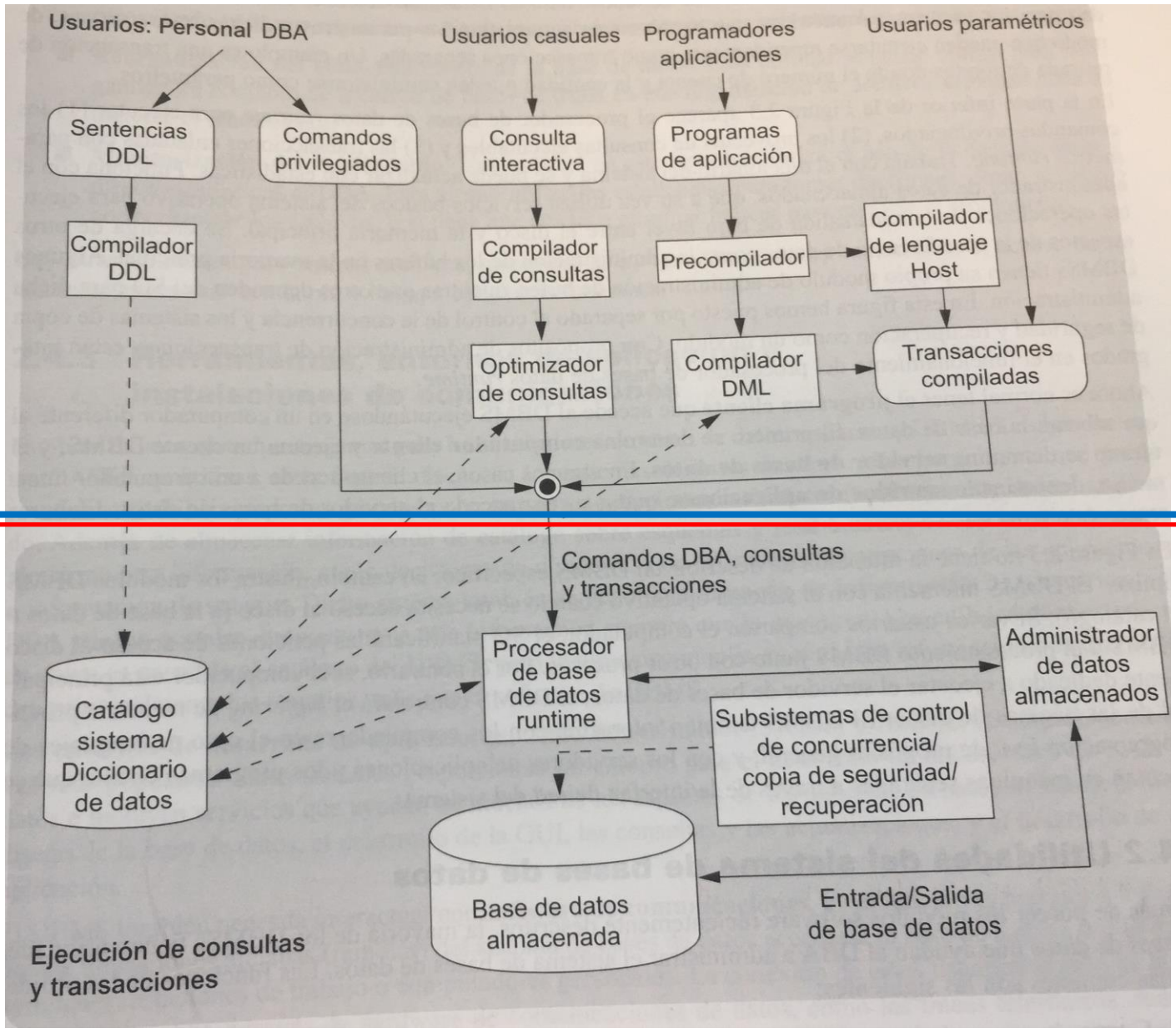
El resultado de una consulta es una tabla, que para accesarla fila por fila se utiliza un cursor (como leer un archivo convencional)

- **DDL:** Data Definition Language
- **DML:** Data Manipulation Language
- **DCL:** Data Control Language

1.7 Componentes de un DBMS (Ambiente operacional de un DBMS)

Módulos constituyentes de un SABD y sus interacciones

Diversos usuarios del entorno de base de datos y sus interfaces



Interior del SABD, responsable del almacenamiento de datos y el procesamiento de transacciones

Usuarios

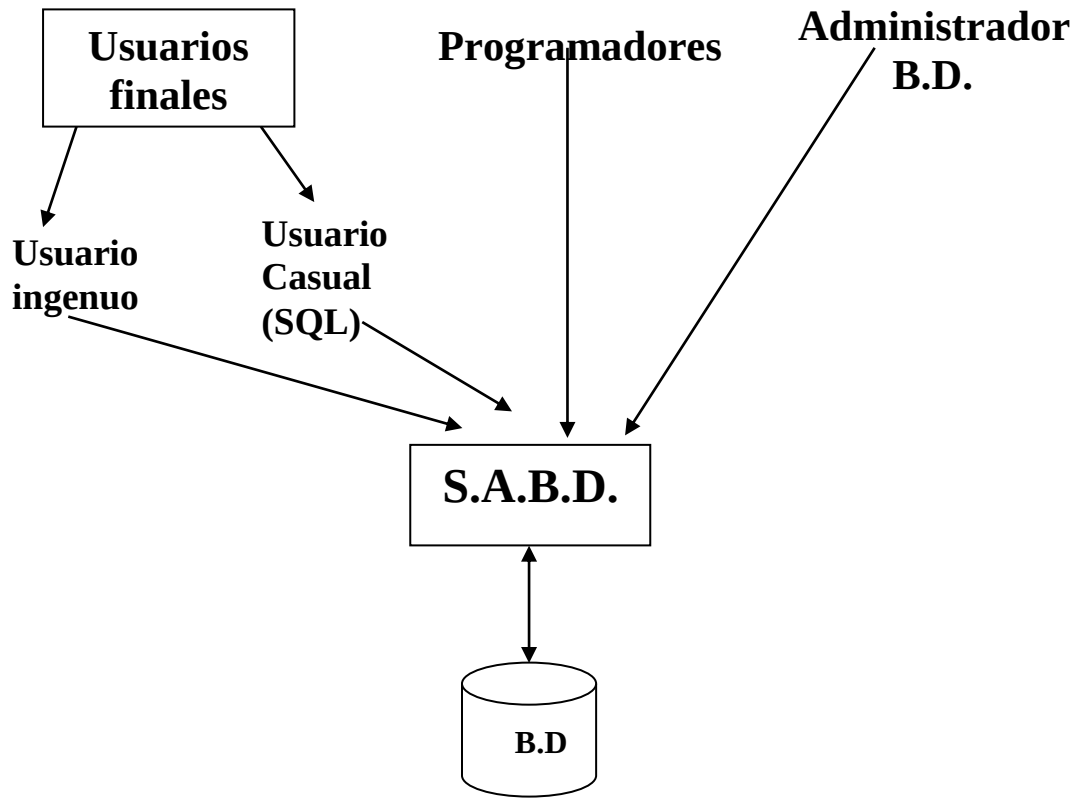
- **DBA**: Administrador de la base de datos. Responsable de:
  - El acceso autorizado a la base de datos
  - La coordinación y monitoreo de uso de la BD
  - La seguridad
  - Los tiempos de respuesta
  - La definición del esquema.
  - La definición de la estructura de almacenamiento y del método de acceso.
  - La modificación del esquema y de la organización física.
  - La especificación de las limitantes de integridad.

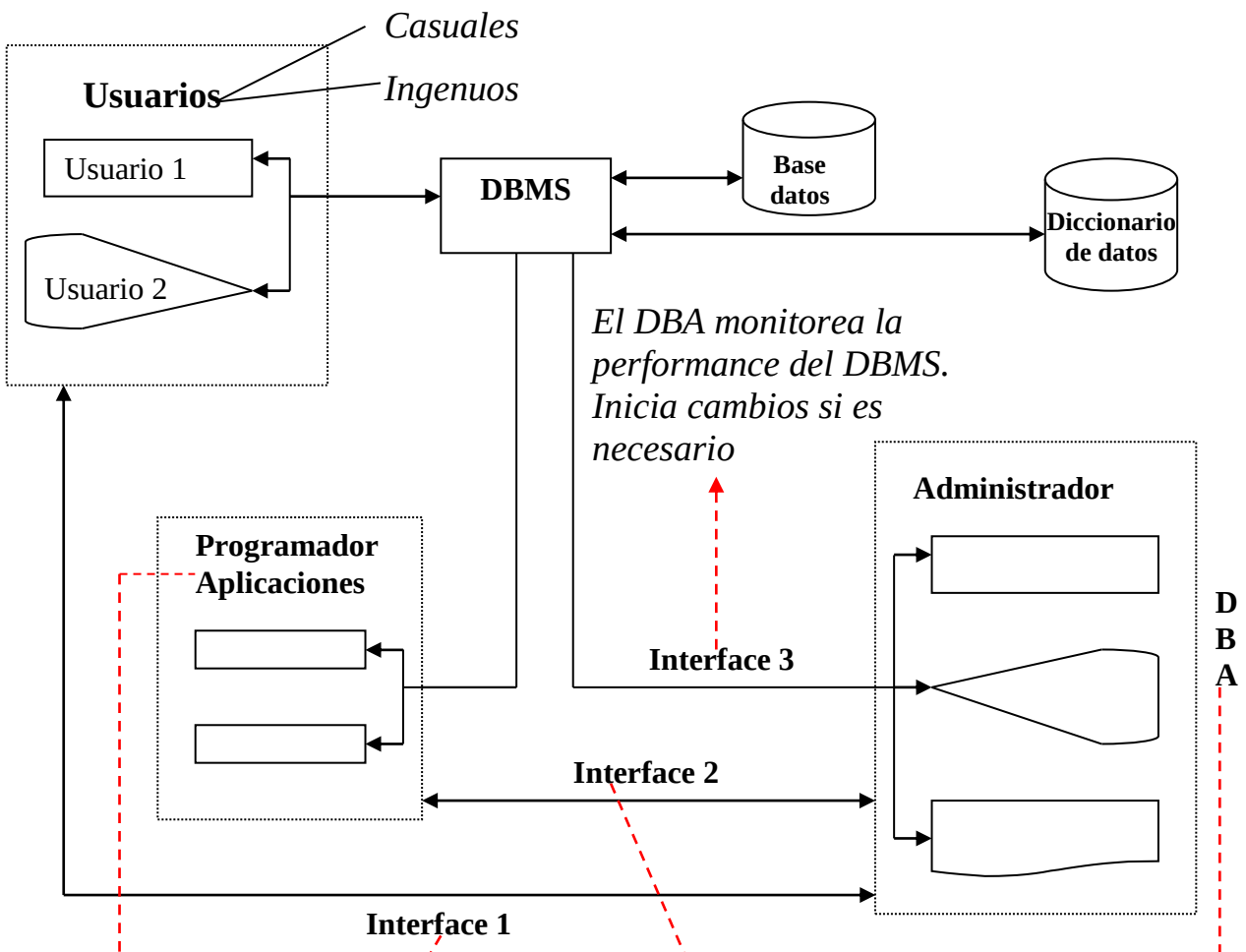
- **Programadores de aplicación**: Programan

- **Usuarios finales casuales**: Acceden ocasionalmente a la base de datos, pero pueden necesitar información diferente en cada momento. Trabajan con interfaces para formular consultas.

- **Usuarios finales paramétricos (ingenuo)**: Realizan las entradas de datos suministrando paramétros a las transacciones predefinidas. Ejemplo: Un cajero de un banco

Usuarios del S.A.B.D.





Ambiente operacional de un DBMS

Escribe el código para procesar los requerimientos de datos por parte de los usuarios

Persona a cargo de la mantención del sistema

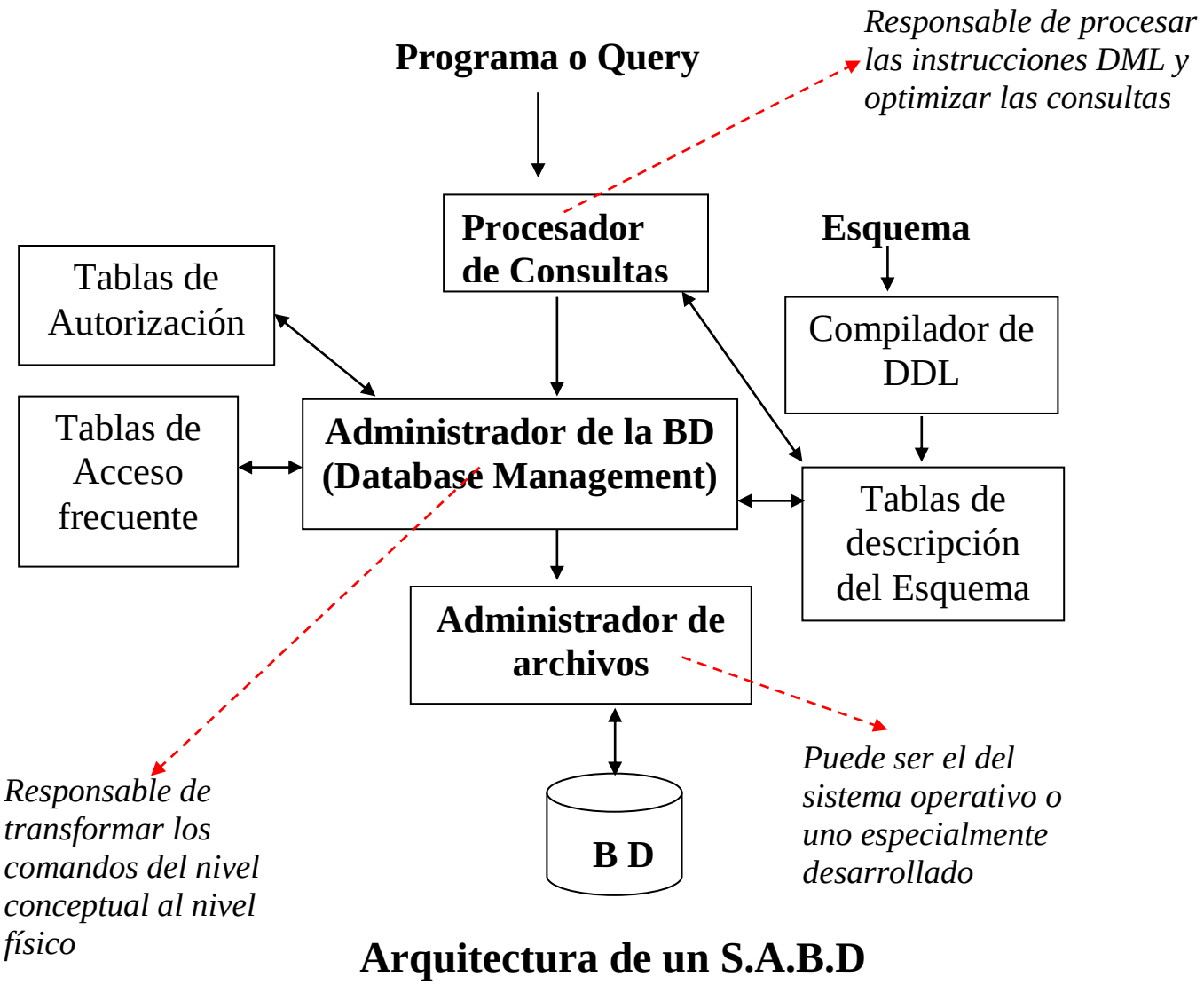
El DBA:

- *define derechos de los usuarios y responsabilidades.*
- *Mediador entre conflictos que pueden surgir.*
- *Capacita a los usuarios*

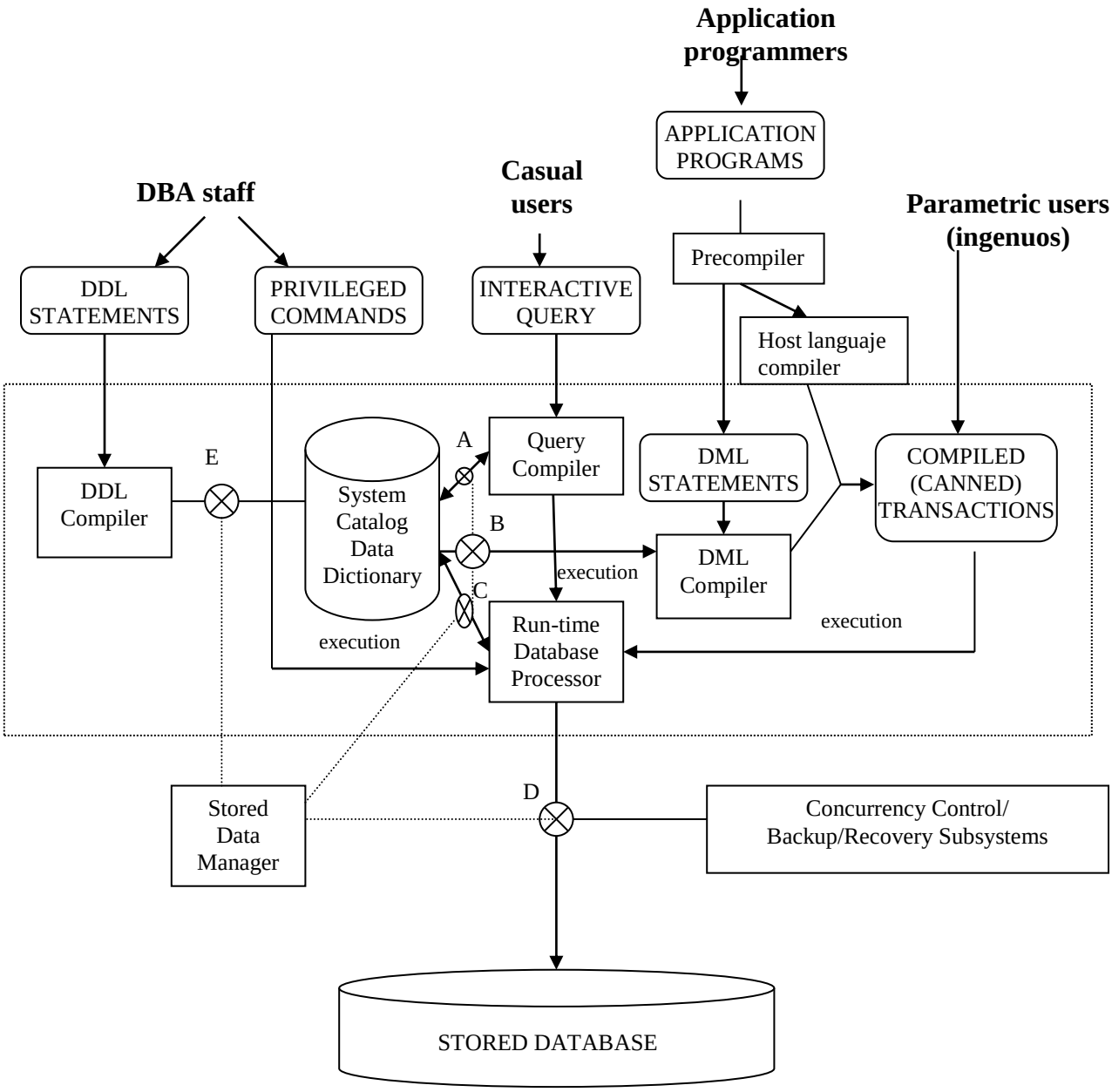
El DBA:

- *Controla todas las definiciones de datos y establece estándares para todos los programadores de aplicaciones que accesan la BD.*
- *Capacita a los programadores*

El ambiente operacional de un DBMS es un sistema integrado de hardware, software y personas que es diseñado para facilitar el almacenamiento, recuperación y control del recurso información.

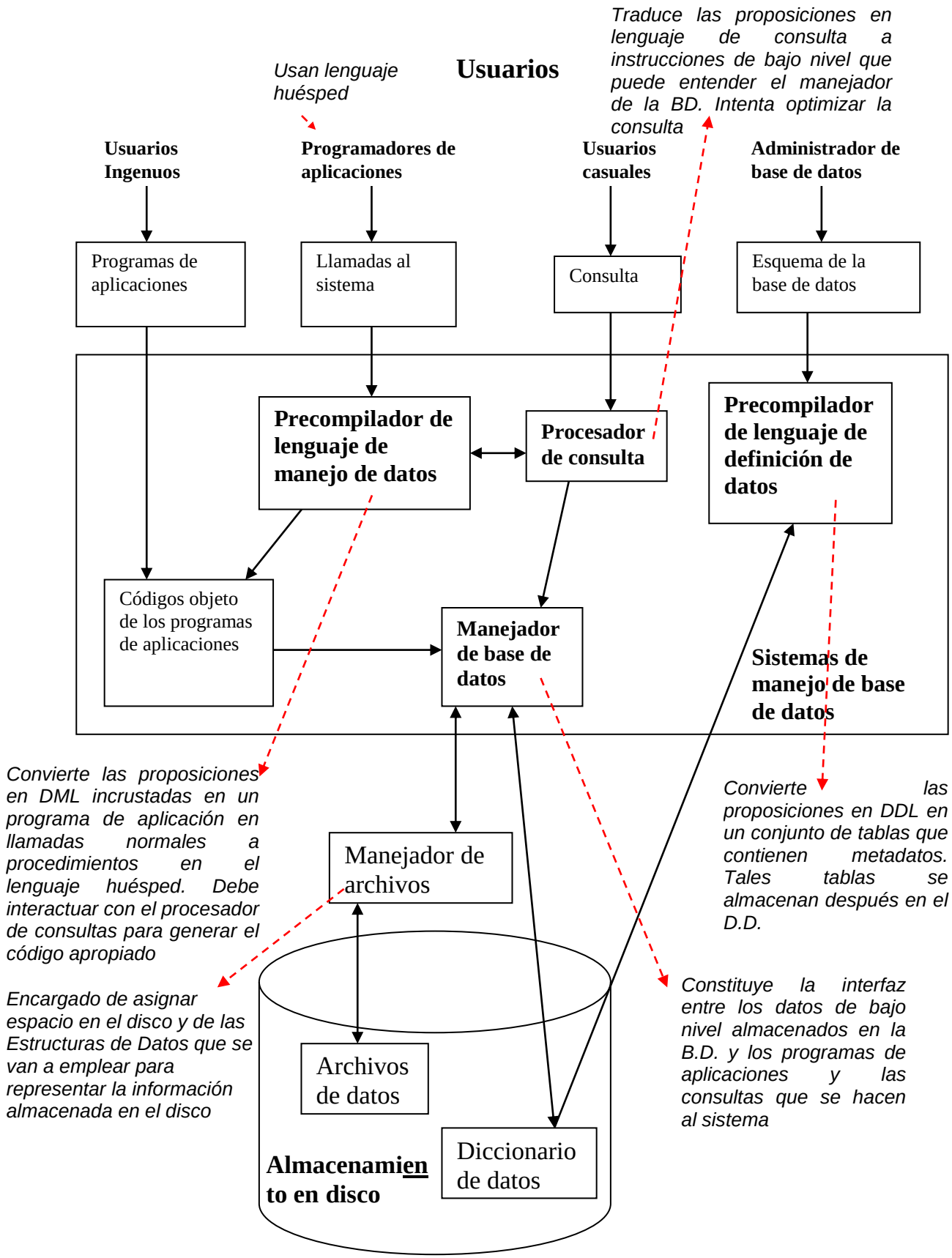


Arquitectura de un S.A.B.D



Las líneas punteadas muestran los accesos que están bajo el control del administrador de datos almacenados (“STORED DATA MANAGER”).

Componentes de un DBMS



Estructura del sistema

1.8 Bases de datos NoSQL

NoSQL (a veces llamado "no sólo SQL", para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL). Difieren del modelo clásico de SGBDR:

- No usan SQL como lenguaje principal de consultas.
- Los datos almacenados no requieren estructuras fijas como tablas
- Normalmente no soportan operaciones JOIN,
- No garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad) y habitualmente escalan bien horizontalmente.

Los sistemas de bases de datos NoSQL crecieron con las principales redes sociales, como Google, Amazon, Twitter y Facebook. Estas tenían que enfrentarse a desafíos con el tratamiento de datos que las tradicionales SGBDR no solucionaban.

Con el crecimiento de la web en tiempo real existía una necesidad de proporcionar información procesada a partir de grandes volúmenes de datos que tenían estructuras horizontales más o menos similares. Estas compañías se

dieron cuenta de que el rendimiento y sus propiedades de tiempo real eran más importantes que la coherencia, en la que las bases de datos relacionales tradicionales dedicaban una gran cantidad de tiempo de proceso

En ese sentido, a menudo, las bases de datos NoSQL están altamente optimizadas para las operaciones recuperar y agregar, y normalmente no ofrecen mucho más que la funcionalidad de almacenar los registros (p.ej. almacenamiento clave-valor). La pérdida de flexibilidad en tiempo de ejecución, comparado con los sistemas SQL clásicos, se ve compensada por ganancias significativas en escalabilidad y rendimiento cuando se trata con ciertos modelos de datos.

Las características comunes entre todas las implementaciones de bases de datos NoSQL suelen ser las siguientes:

- **Consistencia Eventual**: A diferencia de las bases de datos relacionales tradicionales, en la mayoría de sistemas NoSQL, no se implementan mecanismos rígidos de consistencia que garanticen que cualquier

cambio llevado a cabo en el sistema distribuido sea visto, al mismo tiempo, por todos los nodos y asegurando, también, la no violación de posibles restricciones de integridad de los datos u otras reglas definidas. En su lugar y para obtener un mayor rendimiento, se ofrece el concepto de "consistencia eventual", en el que los cambios realizados "con el tiempo" serán propagados a todos los nodos por lo que, una consulta podría no devolver los últimos datos disponibles o proporcionar datos inexactos, en contraposición a ACID (*Atomicity*, *Consistency*, *Isolation*, *Durability*), su analogía en las bases de datos relacionales.

- **Flexibilidad en el esquema:** En la mayoría de base de datos NoSQL, los esquemas de datos son dinámicos; es decir, a diferencia de las bases de datos relacionales en las que, la escritura de los datos debe adaptarse a tablas compuestas a su vez por filas y columnas y tipos de datos pre-definidos, en los sistemas NoSQL, cada registro (o documento, como se les suele llamar en estos casos) puede contener una información con diferente

forma cada vez, pudiendo así almacenar sólo los atributos que interesen en cada uno de ellos, facilitando el polimorfismo de datos bajo una misma colección de información. También se pueden almacenar estructuras complejas de datos en un sólo documento, como por ejemplo almacenar la información sobre una publicación de un blog (título, cuerpo de texto, autor, etc.) junto a los comentarios y etiquetas vertidos sobre el mismo, todo en un único registro.

- **Escalabilidad horizontal**: Por escalabilidad horizontal se entiende la posibilidad de incrementar el rendimiento del sistema añadiendo, simplemente, más nodos (servidores) e indicando al sistema cuáles son los nodos disponibles.
- **Estructura distribuida**: Generalmente los datos se distribuyen, entre los diferentes nodos que componen el sistema. Hay dos estilos de distribución de datos:
- **Tolerancia a fallos y Redundancia**: Pese a lo que cualquiera pueda pensar cuando se habla de NoSQL, no todas las tecnologías existentes bajo este paraguas usan el mismo modelo de datos ya que, al ser sistemas

altamente especializados, la idoneidad particular de una base de datos NoSQL dependerá del problema a resolver.

A menudo, las bases de datos NoSQL se clasifican según su forma de almacenar los datos, y comprenden categorías como:

- **Clave-valor**

  Son el modelo de base de datos NoSQL más popular, además de ser la más sencilla en cuanto a funcionalidad. En este tipo de sistema, cada elemento está identificado por una clave única, lo que permite la recuperación de la información de forma muy rápida. Se caracterizan por ser muy eficientes tanto para las lecturas como para las escrituras.

- **Base de datos Columnar (o Columna ancha):** En vez de "tablas", en las bases de datos de columna tenemos familias de columnas que son los contenedores de las filas. A diferencia de los RDBMS, no necesita conocer de antemano todas las columnas, cada fila no tiene por qué tener el mismo número de columnas. Este tipo de bases de

datos se adecuan mejor a operaciones analíticas sobre grandes conjuntos de datos.

- **Bases de datos documentales**

  Este tipo de base de datos almacena la información como un documento, usando habitualmente para ello una estructura simple como JSON, BSON o XML y donde se utiliza una clave única para cada registro. Este tipo de implementación permite, además de realizar búsquedas por clave–valor, realizar consultas más avanzadas sobre el contenido del documento. Son las bases de datos NoSQL más versátiles.

- **Bases de datos orientadas a grafos**

  Usadas para aquellos datos cuyas relaciones se pueden representar adecuadamente mediante un grafo. Los datos se almacenan en estructuras grafo con nodos (entidades), propiedades (información entre entidades) y líneas (conexiones entre las entidades)

Pese a todas las opciones proporcionadas por el auge de las bases de datos NoSQL, esto no significa la desaparición de las bases de datos de RDBMS ya que son tecnologías

complementarias. Estamos entrando en una era de persistencia políglota, una técnica que utiliza diferentes tecnologías de almacenamiento de datos para manejar las diversas necesidades de almacenamiento de datos.

1.9 Ejemplos de SABD

En la actualidad, existen multitud de SGBD y pueden ser clasificados según la forma en que administran los datos en:

- Relacionales (SQL)
- No relacionales (NoSQL)

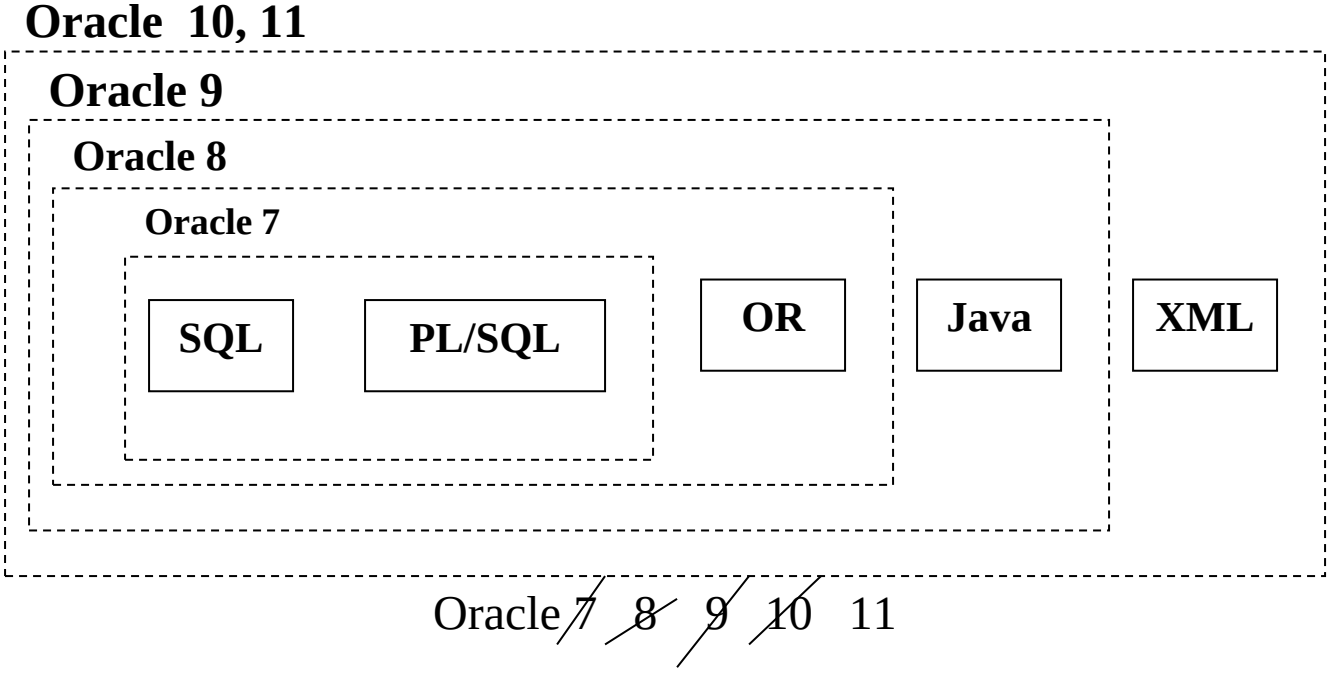
SABD Relacionales

- **ORACLE**

Oracle es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hace que sólo se vea en empresas muy grandes y multinacionales, por norma general.

Oracle se basa en la tecnología cliente/servidor, pues bien, para su utilización primero sería necesario la instalación de la herramienta servidor (Oracle 8i) y posteriormente podríamos atacar a la base de datos desde otros equipos con herramientas de desarrollo como Oracle Designer y Oracle Developer, que son las herramientas básicas de programación sobre Oracle.

Para desarrollar en Oracle utilizamos PL/SQL un lenguaje de 5ª generación, bastante potente para tratar y gestionar la base de datos, también por norma general se suele utilizar SQL al crear un formulario.



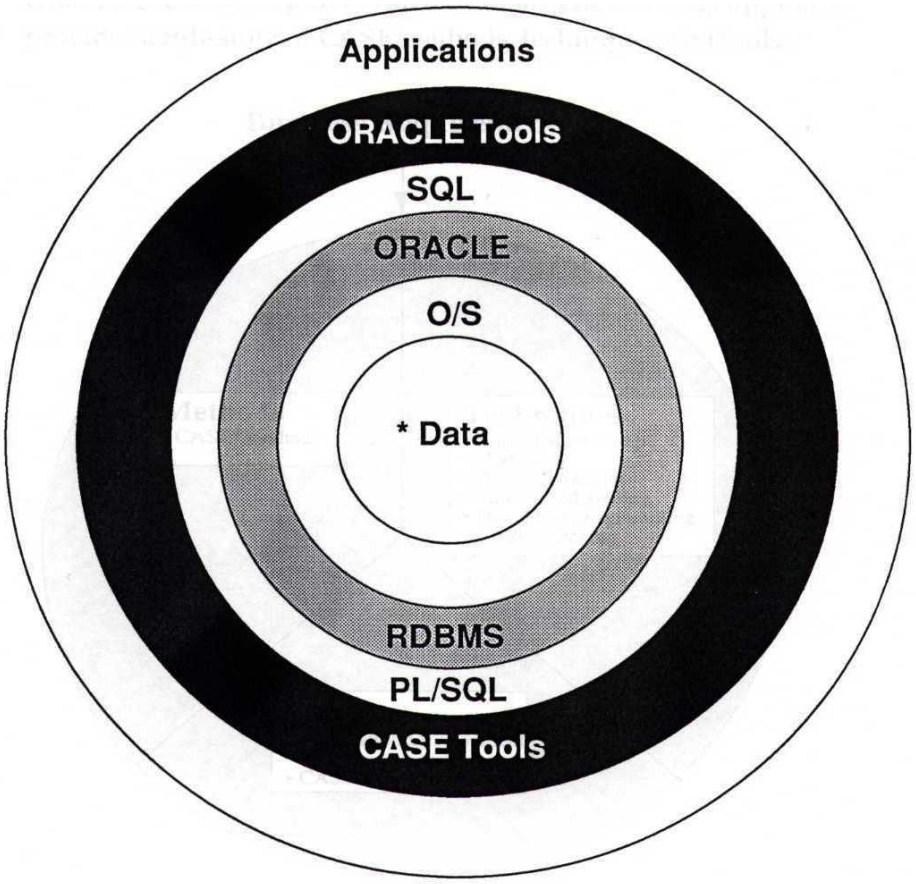
Oracle'Cloud (Oracle 12)

La última versión de la base de datos de Oracle ya está disponible en la nube

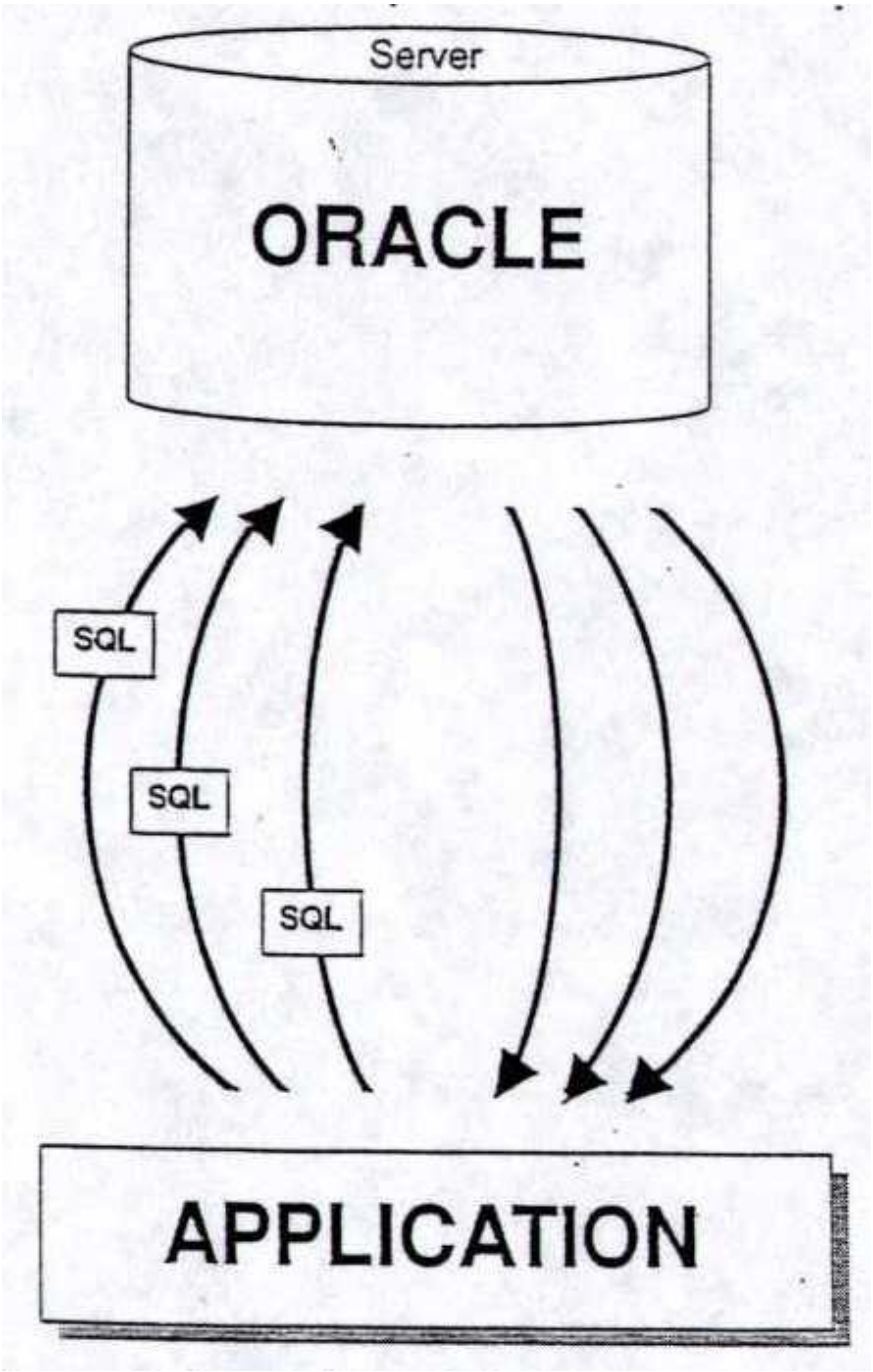
Ejemplo de SABD Oracle

Oracle Express
Business Intelligence:

- OLAP
- Data Warehousing
- Data Mining



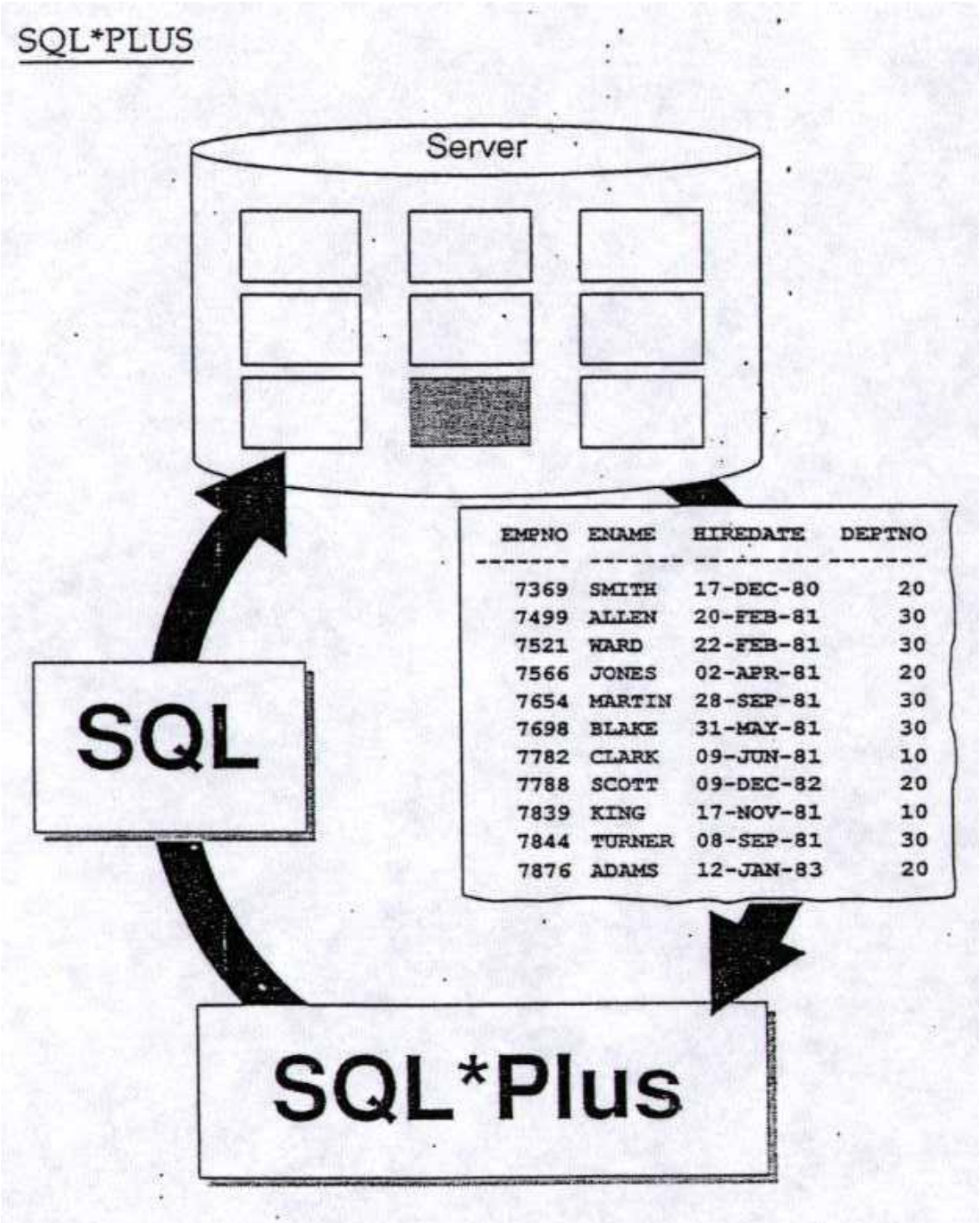
Arquitectura de ORACLE



Versiones estándares de SQL:

- SQL1 ANSI 1986
- SQL2 (SQL-92) ANSI 1992
- SQL3 Incluye conceptos de Orientación al Objeto

SQL: Lenguaje para la construcción de aplicaciones en ORACLE.



SQL *PLUS de ORACLE, una de las herramientas más simples para interactuar con la base de datos

SQL*PLUS es una herramienta de Oracle que tiene propiedades de formateo, no siendo su objetivo principal.

- **Access**

Es un sistema de gestión de bases de datos Relacional creado y modificado por Microsoft para uso personal de pequeñas organizaciones. Es un componente de la suite Microsoft Office aunque no se incluye en el paquete "básico".

- **MICROSOFT SQL SERVER**

Microsoft SQL Server es un sistema de gestión de bases de datos relacionales (SGBD) basada en el lenguaje Transact-SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Así de tener unas ventajas que a continuación se pueden describir.

- **MySQL**

Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones Se desarrolla como software libre.

- **INFORMIX**

Informix es una familia de productos RDBMS de IBM, adquirida en 2001 a una compañía (también llamada Informix o Informix Software) cuyos orígenes se remontan a 1980.

- **PostgreSQL**

Es un servidor de base de datos relacional orientada a objetos de software libre.

Sistemas Gestores de bases de datos No Relacionales (NoSQL)

Una base de datos no relacional (**NoSQL**) es aquella base de datos que:

- No requiere de estructuras de datos fijas como tablas
- No garantiza completamente las características ACID
- Escala muy bien horizontalmente.

Se utilizan en entornos distribuidos que han de estar siempre disponibles y operativos y que gestionan un importante volumen de datos.

Para la administración de este tipo de bases de datos, actualmente los principales sistemas gestores de bases de datos (SGBD NoSQL) son:

- **MongoDB**

Sistema Gestor de Bases de Datos no relacionales (SGBD NoSQL) más popular y utilizado actualmente.

Empresas como Google, Facebook, eBay, Cisco o Adobe utilizan MongoDB como Sistema Gestor de Bases de datos.

- **Redis**

Redis está basado en el almacenamiento clave-valor. Podríamos verlo como un vector enorme que almacena todo tipo de datos, desde cadenas, hashses, listas, etc.

- **Cassandra**

Al igual que Redis, Cassandra también utiliza almacenamiento clave-valor. Es un SGBD NoSQL distribuido y masivamente escalable.

Facebook, Twitter, Instagram, Spotify o Netflix utilizan Cassandra.

Dispone de un lenguaje propio para las consultas, denominado **CQL** (Cassandra Query Languaje).

Otros SGBD NoSQL

- Oracle NoSQL
- Azure Cosmos DB
- RavenDB
- ObjectDB
- Apache CouchDB
- Neo4j
- Google BigTable
- Apache Hbase
- Amazon DynamoDB