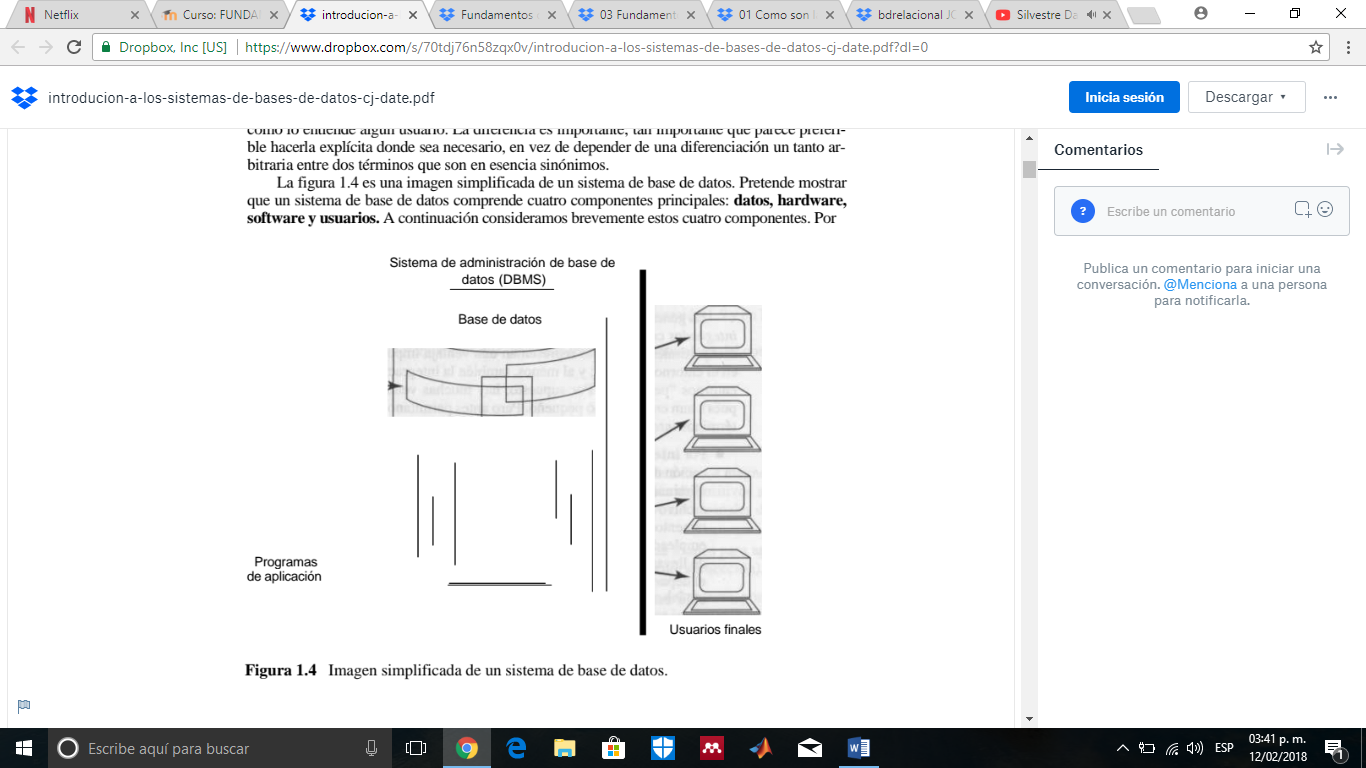
**1.-BASE DE DATOS**

FUNDAMENTOS DE BASE DE DATOS

Un **sistema gestor de bases de datos** (SGBD) consiste en una colección de datos inter-relacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada **base de datos**, contiene información relevante para una empresa. Los sistemas de bases de datos se diseñan para alma-cenar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para el almacenamiento de la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la seguridad de la información almacenada, en caso de caídas del sistema o intentos de accesos sin autorización. Si los datos están compartidos por varios usuarios, el sistema debe evitar posibles resultados anómalos.

(Silberschatz, Henry, & S., pág. 24)

Un sistema de bases de datos es básicamente un sistema computarizado para llevar registros. Es posible considerar a la propia base de datos como una especie de armario electrónico para archivar; es decir, es un depósito o contenedor de una colección de archivos de datos computarizados. Los usuarios del sistema pueden realizar una variedad de operaciones sobre dichos archivos. Por ejemplo:

■Agregar nuevos archivos vacíos a la base de datos;

■Insertar datos dentro de los archivos existentes;

■Recuperar datos de los archivos existentes;

■Modificar datos en archivos existentes;

■Eliminar datos de los archivos existentes;

■Eliminar archivos existentes de la base de datos

(Date C. , Transaccion , 2001, pág. 24)

**2.-OBJETIVO DE LAS BASES DE DATOS**

El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto *práctica* como *e*fi*ciente* Para las personas que lo usan para la recuperación y alma-cenamiento de la información*.*

(Silberschatz, Henry, & S., pág. 24)

**7.-METADATOS**

Un diccionario de datos contiene metadatos, es decir, datos acerca de los datos. El esquema de una tabla es un ejemplo de metadatos. Un sistema de base de datos consulta el diccionario de datos antes de leer o modifica los datos reales. Especificamos el almacenamiento y los métodos de acceso usados por el sistema de bases de datos por un conjunto de instrucciones en un tipo especial de LDD denominado lenguaje de almacenamiento y definición de datos. Estas instrucciones definen los detalles de implementación de los esquemas de base de datos, que se ocultan usualmente a los usuarios.

(sanchez , 2004, pág. 7)

**8.-DATO**

La palabra datos se deriva del vocablo latín para "dar"; por lo tanto, los datos en realidad son hechos dados, a partir de los cuales es posible inferir hechos adicionales.

(Date J. , 2001, pág. 13)

**9.-REDUNDANCIA DE DATOS**

Se llama así a los datos que se repiten continua e innecesariamente por las tablas de las bases de datos.

(Sánchez, 2004, pág. 23)

**10.-INTEGRIDAD DE DATOS**

El término integridad se refiere a la exactitud o corrección de los datos en la base de datos.

(J. Date, 2001, pág. 249)

**11.-PROPIEDADES**

Como acabamos de señalar, una entidad es cualquier objeto acerca del cual queremos registrar información. De donde se desprende que las entidades (incluidos los vínculos) poseen propiedades que corresponden a la información que deseamos registrar sobre ellas. Por ejemplo, los proveedores tienen localidades; las partes tienen pesos; los proyectos tienen prioridades; las asignaciones (de empleados a proyectos) tienen fechas de inicio, etcétera. Por lo tanto, dichas propiedades deben estar representadas en la base de datos. Por ejemplo, la base de datos podría incluir una tabla denominada V que represente a los proveedores y esa tabla podría incluir una columna de nombre CIUDAD que represente a las localidades de los proveedores .En general, las propiedades pueden ser tan simples o tan complejas como queramos. Por ejemplo, la propiedad "localidad del proveedor" es supuestamente bastante simple, ya que sólo consiste en un nombre de ciudad y puede ser representada en la base de datos por una simple cadena de caracteres. En contraste, un almacén podría tener una propiedad "plan de piso", que podría ser bastante compleja, consistir tal vez en todo un dibujo arquitectónico y en el texto descriptivo asociado. Al momento de la publicación de este libro, la mayoría de los productos de bases de datos estaban apenas logrando manejar propiedades complejas como el dibujo y el texto. Regresaremos a este tema más delante (en especial en el capítulo 5 y en la Parte VI); mientras tanto, en la mayoría de los casos (en donde signifique una diferencia) daremos por hecho que las propiedades son "simples" y que pueden ser representadas mediante tipos de datos "simples". Los ejemplos de dichos tipos "simples" incluyen números, cadenas de caracteres, fechas, horas, etcétera.

(Date C. , Propiedades, 2001, pág. 35)

**12.-INFORMACIÓN**

Toda la información de la base de datos debe estar representada explícitamente en el esquema lógico. Es decir, todos los datos están en las tablas.

(Sánchez, 2004, pág. 14)

**14.-MODELO DE DATOS**

Es una definición lógica, independiente y abstracta de los objetos, operadores y demás que en conjunto constituyen la máquina abstracta con la que interactúan los usuarios. Los objetos nos permiten modelar la estructura de los datos. Los operadores nos permiten modelar su comportamiento. Entonces, de manera útil, podemos distinguir al modelo de su implementación:

■La implementación de determinado modelo de datos es una realización física, en una máquina real, de los componentes de la máquina abstracta que en conjunto constituyen ese modelo. Resumiendo: El modelo es aquello que los usuarios tienen que conocer; la implementación es lo que los usuarios no tienen que conocer. Nota: Como puede ver, la distinción entre modelo e implementación es en realidad sólo un caso especial (uno muy importante) de la conocida distinción entre lógico y físico. Sin embargo, por desgracia muchos de los sistemas de bases de datos actuales (incluso aquellos que dicen ser relaciónales) no hacen estas distinciones con tanta claridad como debieran. De hecho, parece no haber un buen entendimiento de estas distinciones y de la importancia de hacerlas. Como con-secuencia, a menudo hay una brecha entre los principios de las bases de datos (la forma en que los sistemas de bases de datos deberían funcionar) y la práctica de las bases de datos (la forma en que realmente funcionan). En este libro nos enfocamos principalmente en los principios; aun-que es justo advertirle que cuando comience a utilizar un producto comercial, podría llevarse algunas sorpresas desagradables. Para concluir esta sección, debemos mencionar el hecho de que en realidad el término mo-delo de datos es utilizado en la literatura con dos significados muy distintos. El primero es como se describió anteriormente. El segundo es como un modelo de los datos persistentes de alguna empresa en particular (por ejemplo, la compañía manufacturera KnowWare Inc. que mencionamos anteriormente en esta sección). La diferencia entre ambos significados puede ser caracterizada como sigue:

■En el primer sentido, un modelo de datos es como un lenguaje de programación (aunque en cierto modo abstracto) cuyos elementos pueden ser usados para resolver una amplia variedad de problemas específicos, pero que en sí y por sí mismos no tienen una conexión directa con ninguno de estos problemas específicos.

■En el segundo sentido, un modelo de datos es como un programa específico escrito en es el lenguaje. En otras palabras, un modelo de datos que toma las características que ofrece algún modelo como el primero y las aplica a cierto problema específico. Puede ser visto como una aplicación específica de algún modelo con el primer significado. De aquí en adelante, en este libro usaremos el término modelo de datos sólo en el primer sentido, excepto cuando se indique lo contrario.

(Date C J, 2001, pág. 36)

**15.-Jerárquias**

En ellas se organiza la información se organiza con un jerarquía en la que la relación entre las entidades de este modelo siempre es del tipo **padre / hijo**. De esta forma hay una serie de nodos que contendrán atributos y que se relacionarán con nodos hijos de forma que puede haber más de un hijo para el mismo padre (pero un hijo sólo tiene un padre).

Las entidades de este modelo se llaman **segmentos** y los atributos **campos**. La forma visual de este modelo es de árbol invertido, en la parte superior están los padres y en la inferior los hijos.

(Sanchez , Jerarquias, 2004, pág. 7)

**18.-En red**

Se trata de un modelo que se utilizó durante mucho tiempo. Organiza la información en **registros** y **enlaces**. Los registros representan las entidades del modelo entidad / relación. En los registros se almacenan los datos utilizando **atributos**. Los enlaces permiten relacionar los registros de la base de datos.

El modelo en red más aceptado es el llamado **codasyl**, que durante mucho tiempo se ha convertido en un estándar. Las bases de datos en red son parecidas a las jerárquicas sólo que en ellas puede haber más de un padre. En este modelo se pueden representar perfectamente relaciones varios a varios. Pero su dificultad de manejo y complejidad hace que se estén abandonando completamente.

(Sanchez , En red, 2004, pág. 8)

**21.-Relacionales**

Los datos se muestran en forma de tablas y relaciones. Este es el modelo que se comenta en el presente documento. De hecho es el claramente más popular.

(Sanchez , Relacionales, 2004, pág. 8)

**24.-ORIENTADAS A OBJETOS**

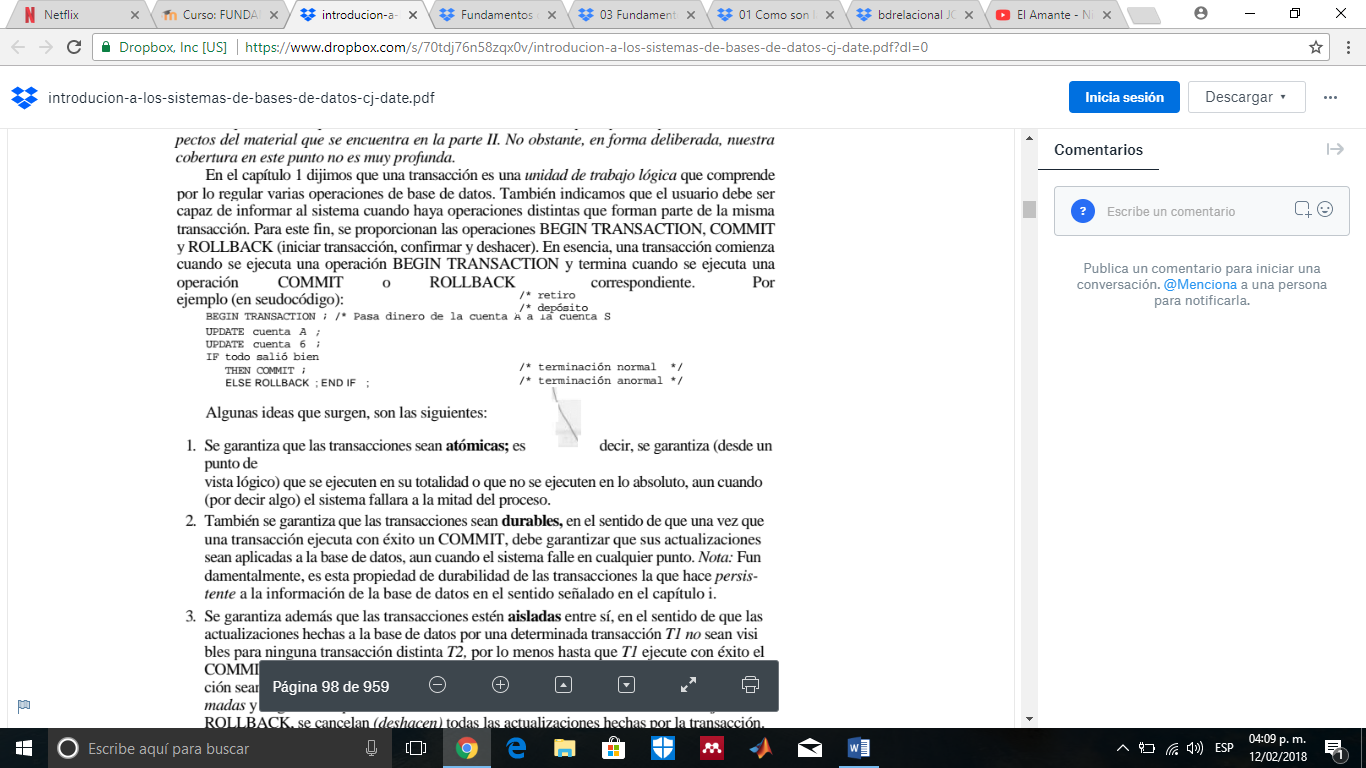
Desde la aparición de la programación orientada a objetos (POO u OOP) se empezó a pensar en bases de datos adaptadas a estos lenguajes. En estos lenguajes los datos y los procedimientos se almacenan juntos. Esta es la idea de las bases de datos orientadas a objetos.

A través de esta idea se intenta que estas bases de datos consiguen arreglar las limitaciones de las relacionales. Por ejemplo el problema de la herencia, tipos definidos por el usuario, disparadores almacenables en la base de datos, soporte multimedia... Se supone que son las bases de datos de tercera generación (la primera fue las bases de datos en red y la segunda las relacionales), lo que significa que el futuro parece estar a favor de estas bases de datos. Pero siguen sin reemplazar a las relacionales (aunque cada vez hay más)

(Sanchez, Orientadas de objeto, pág. 8)

**30.-TRANSACCIÓN**

Es una unidad de trabajo lógica que comprende por lo regular varias operaciones de base de datos. También indicamos que el usuario debe ser capaz de informar al sistema cuando haya operaciones distintas que forman parte de la misma transacción. Para este fin, se proporcionan las operaciones BEGIN TRANSACTION, COMMIT y ROLLBACK (iniciar transacción, confirmar y deshacer). En esencia, una transacción comienza cuando se ejecuta una operación BEGIN TRANSACTION y termina cuando se ejecuta una operación COMMIT o ROLLBACK correspondiente. Por ejemplo (en seudocódigo):



Algunas ideas que surgen, son las siguientes:

1. Se garantiza que las transacciones sean atómicas; es decir, se garantiza (desde un punto de vista lógico) que se ejecuten en su totalidad o que no se ejecuten en lo absoluto, aun cuando (por decir algo) el sistema fallara a la mitad del proceso.

2. También se garantiza que las transacciones sean durables, en el sentido de que una vez que una transacción ejecuta con éxito un COMMIT, debe garantizar que sus actualizaciones sean aplicadas a la base de datos, aun cuando el sistema falle en cualquier punto. Nota: Fundamentalmente, es esta propiedad de durabilidad de las transacciones la que hace persistente a la información de la base de datos en el sentido señalado en el capítulo i.

3. Se garantiza además que las transacciones estén aisladas entre sí, en el sentido de que las actualizaciones hechas a la base de datos por una determinada transacción T1 no sean visibles para ninguna transacción distinta T2, por lo menos hasta que T1 ejecute con éxito el COMMIT. La operación COMMIT hace que las actualizaciones efectuadas pop una transacción sean visibles para otras transacciones; se dice que dichas actualizaciones están confirmadas y se garantiza que nunca sean canceladas. Si en vez de ello la transacción ejecuta un ROLLBACK, se cancelan (deshacen) todas las actualizaciones hechas por la transacción. En este último caso, el efecto es como si nunca se hubiese realizado la transacción.

4. Se garantiza (por lo regular) que la ejecución intercalada de un conjunto de transacciones concurrentes sea seriable, en el sentido que produzca el mismo resultado que se obtendría si se ejecutaran esas mismas transacciones, una a la vez, en un cierto orden serial no especificado.

(Date C. , Transaccion , 2001, pág. 17)

**31.-ESQUEMA CANÓNICO**

Representa los datos en un formato más cercano al del ordenador.

(Sánchez, 2004, pág. 15)

**33.- USUARIOS FINALES**

Los usuarios finales son las personas cuyos trabajos requieren acceso a la base de datos para realizar consultas, actualizaciones e informes; la base de datos existe principalmente para ser utilizada.

(RAMEZ ELMASRI, 2007, pág. 14)

**34.- PROGRAMADORES DE APLICACIONES**

Los **programadores de aplicaciones** implementan esas especificaciones como

programas; después, verifican, depuran, documentan y mantienen esas transacciones enlatadas. Dichos analistas y programadores deben familiarizarse con todas las posibilidades proporcionadas por el DBMS al objeto de desempeñar sus tareas.

(RAMEZ ELMASRI, 2007, pág. 15)

**35.- ADMINISTRADOR DE BASE DE DATOS**

Para acomodar estos sistemas, muchas empresas crearon el puesto de DBA (Administrador de base de datos) o, incluso, departamentos completos de administración para supervisar y controlar las actividades del ciclo vital de la base de datos.

(RAMEZ ELMASRI, 2007, pág. 346)

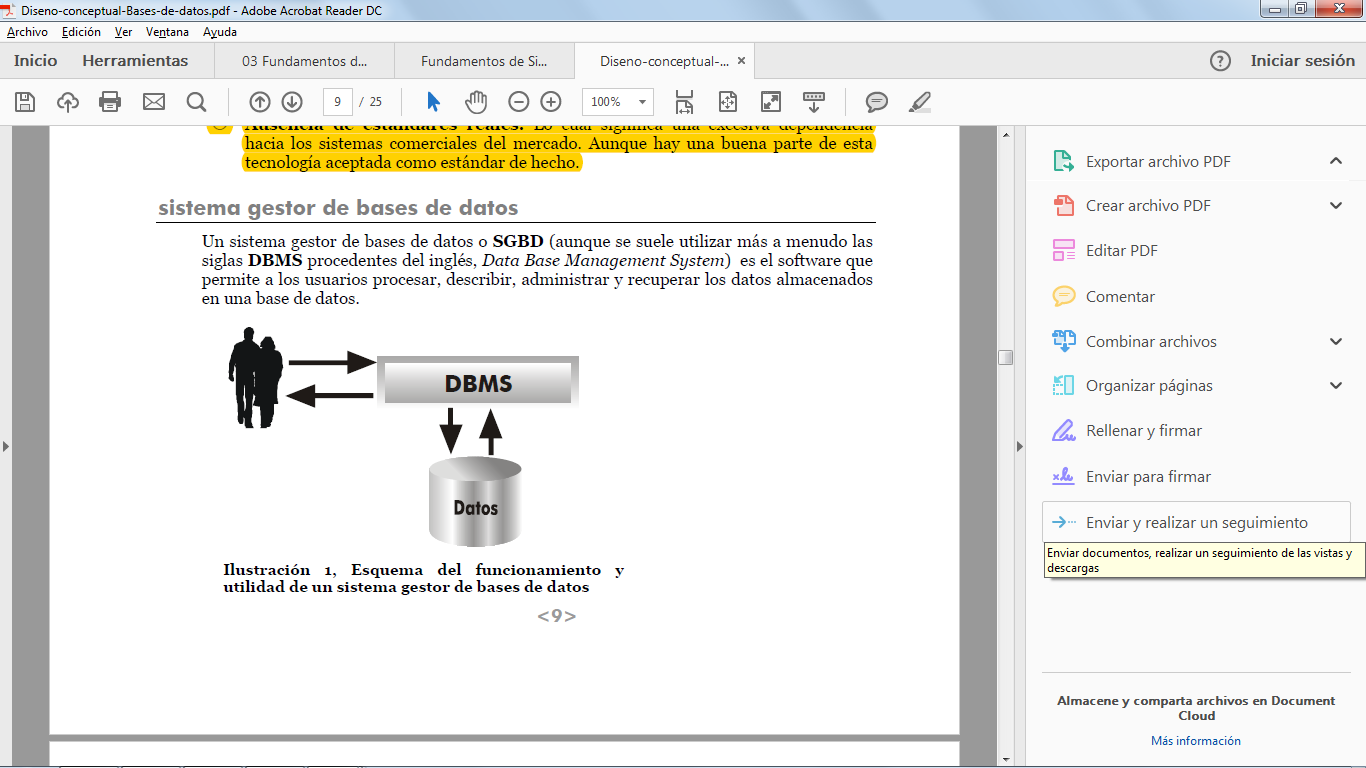
**37.- SISTEMA GESTOR DE BASE DE DATOS**

Un sistema gestor de bases de datos o **SGBD** es el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos.

(Sánchez, 2004, pág. 9)

**38.- ARQUITECTURA DEL SGBD**

Ilustración 1, Esquema del funcionamiento y utilidad de un sistema gestor de bases de datos



(Sánchez, 2004, pág. 9)

**39.- Estructura Lógica**

La estructura lógica sirve para que las aplicaciones puedan utilizar los elementos de la base de datos sin saber realmente cómo se están almacenando. Es una estructura que permite idealizar a la base de datos. Sus elementos son objetos, entidades, nodos, relaciones, enlaces,...

(Sánchez, 2004, pág. 8)

**40.- Estructura Física**

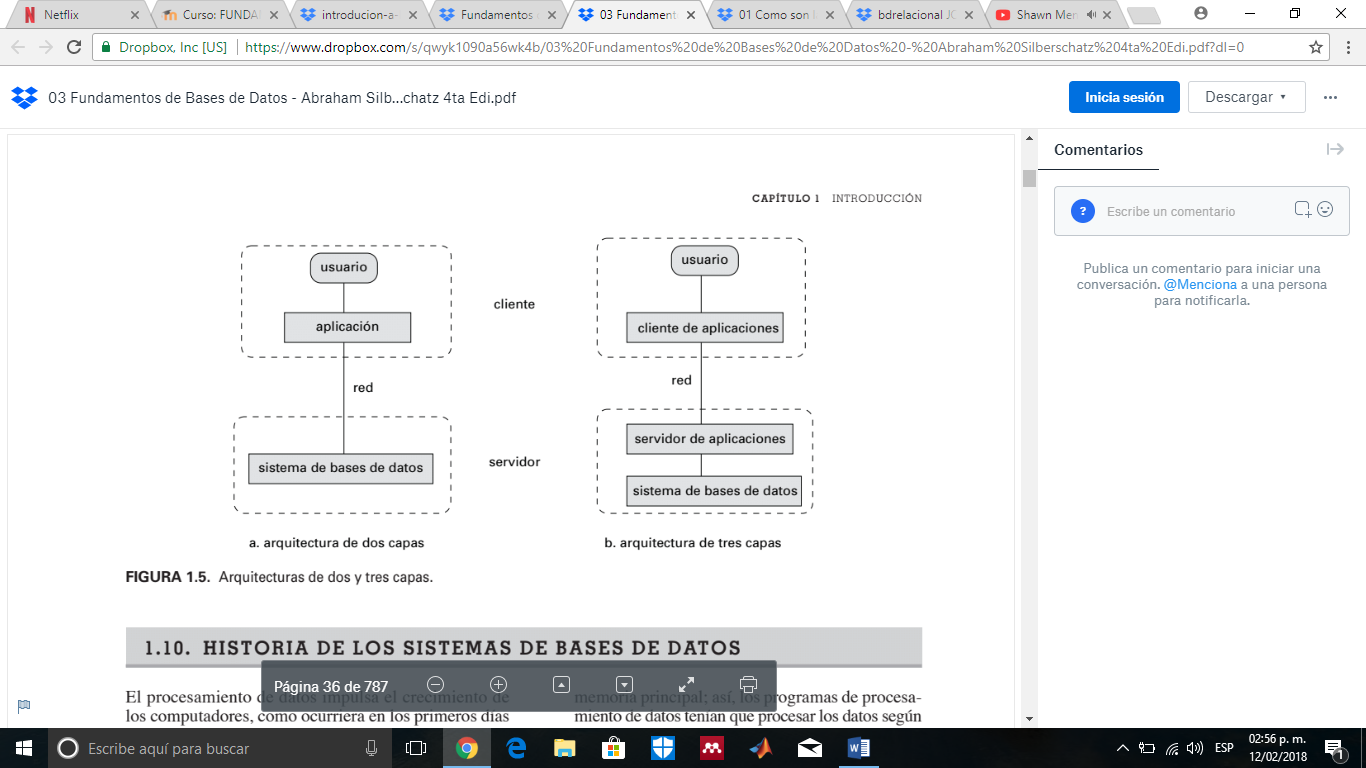
Es la estructura de los datos tan cual se almacenan en las unidades de disco. La correspondencia entre la estructura lógica y la física se almacena en la base de datos (en los metadatos).

(Sánchez, 2004, pág. 8)

**41.- ARQUITECTURA DE APLICACIONES**

La mayoría de usuarios de un sistema de bases de datos no están situados actualmente junto al sistema de bases de datos, sino que se conectan a él a través de una red. Se puede diferenciar entonces entre las máquinas clien-te, en donde trabajan los usuarios remotos de la base de datos, y las máquinas servidor, en las que se ejecuta el sistema de bases de datos. Las aplicaciones de bases de datos se dividen usual-mente en dos o tres partes, como se ilustra en la Figu-ra 1.5. En una arquitectura de dos capas, la aplicación se divide en un componente que reside en la máquina cliente, que llama a la funcionalidad del sistema de bases de datos en la máquina servidor median-te instrucciones del lenguaje de consultas. Los están-dares de interfaces de programas de aplicación como ODBC y JDBC se usan para la interacción entre el cliente y el servidor. En cambio, en una arquitectura de tres capas, la máquina cliente actúa simplemente como frontal y no contiene ninguna llamada directa a la base de datos. En su lugar, el cliente se comunica con un servidor de aplicaciones, usualmente mediante una interfaz de formularios. El servidor de aplicaciones, a su vez, se comunicación el sistema de bases de datos para acceder a los datos. La lógica de negocio de la aplicación, que establece las acciones a realizar bajo determinadas condiciones, se incorpora en el servidor de aplicaciones, en lugar de ser distribuida a múltiples clientes. Las aplicaciones de tres capas son más apropiadas para grandes aplicaciones, y para las aplicaciones que se ejecutan en World Wide Web.

(Date C. , 2001, pág. 36)



**LOS TRES NIVELES DE LA ARQUITECTURA**

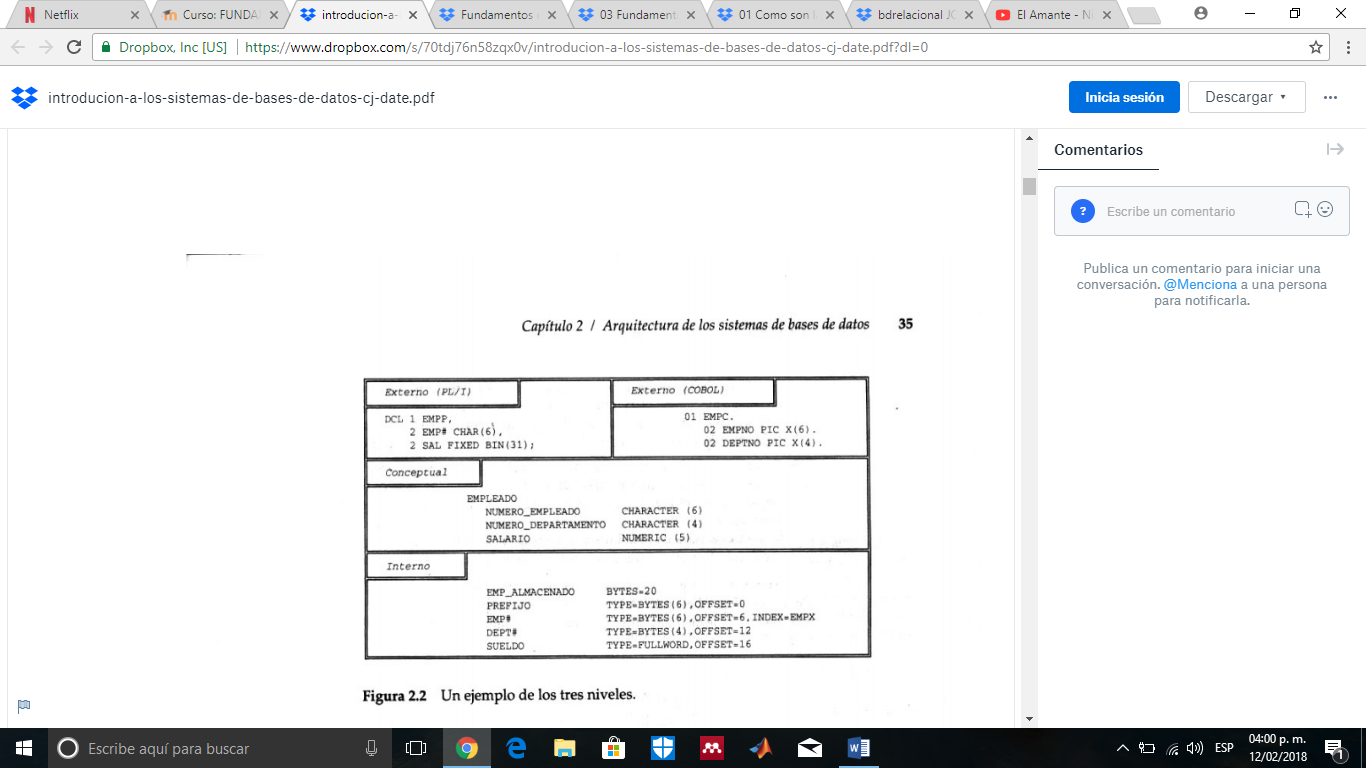
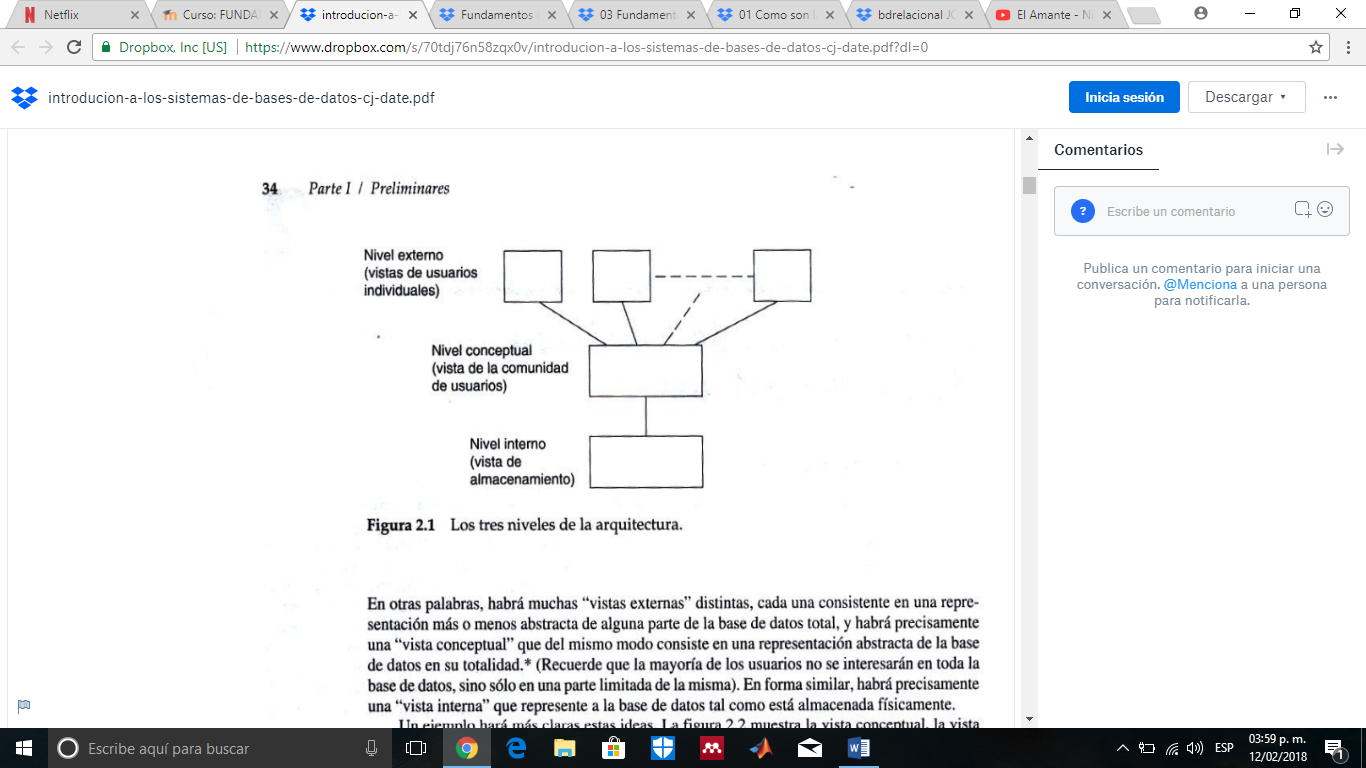
La arquitectura ANSI/SPARC se divide en tres niveles, conocidos como interno, conceptual y externo, respectivamente (vea la figura 2.1). Hablando en términos generales:

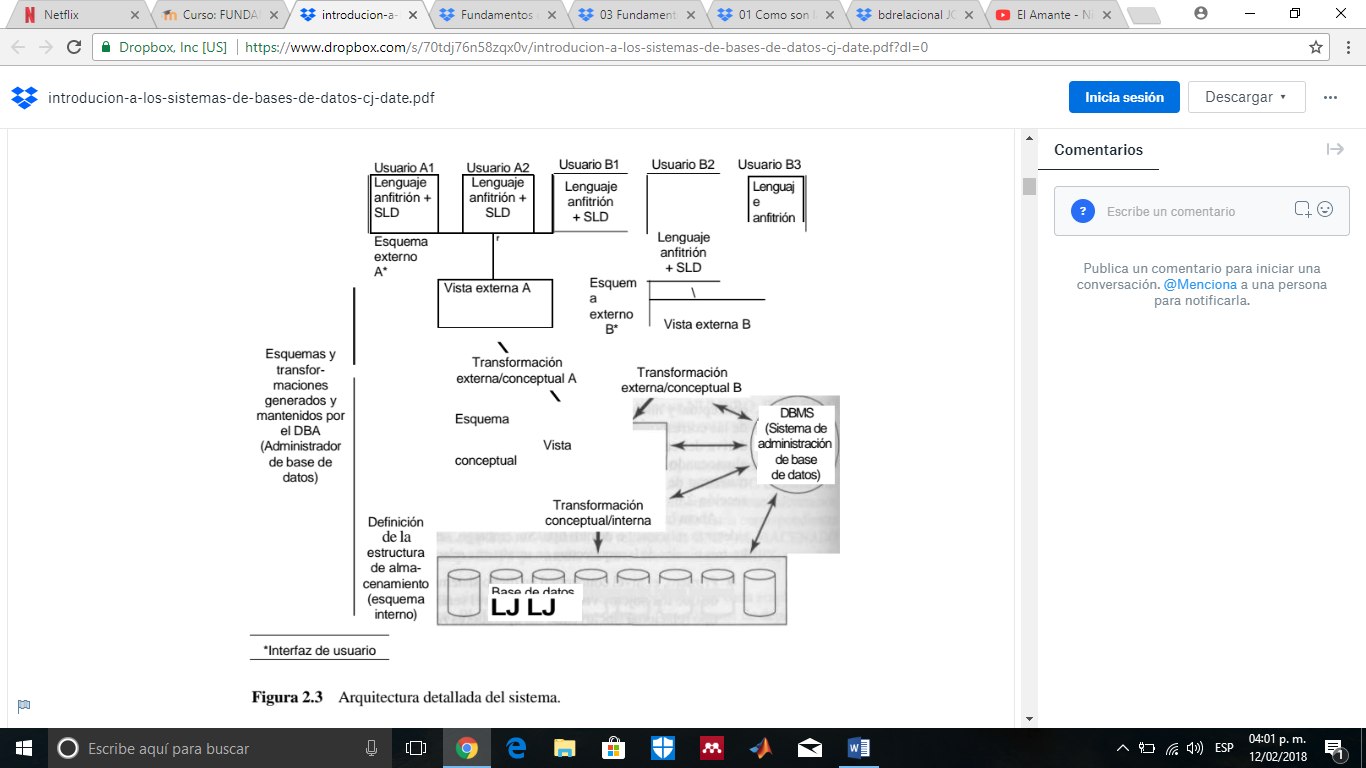
■El nivel interno (también conocido como el nivel físico) es el que está más cerca del almacenamiento físico; es decir, es el que tiene que ver con la forma en que los datos están almacenados físicamente.

■ 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.





**42.-EL NIVEL EXTERNO**

El nivel externo es el nivel del usuario individual. Como expliqué en el capítulo 1, 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 (a) el código de máquina, el lenguaje ensamblador y los lenguajes como PL/I pueden ser vistos como tres "generaciones" de lenguajes anteriores, y (b) 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 ensamblador 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 (como expliqué en el capítulo 1).

(Date C J, 2001, pág. 37)

**43.-EL NIVEL CONCEPTUAL**

La vista conceptual es una representación de todo el contenido de la información de la base de datos, de nuevo (al igual que con la vista externa) en una forma un poco abstracta comparada con la forma 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 pretende 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 pudieran 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, más 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.

Entonces, la vista conceptual es una vista del contenido total de la base de datos, y el esquema conceptual es una definición de esa vista. Sin embargo, sería engañoso dar por hecho que el esquema conceptual no es nada más que un conjunto de definiciones muy similar a las definiciones que se encuentran (por ejemplo) en un programa COBOL actual. Las definiciones del esquema conceptual pretenden incluir muchas características adicionales, como las restricciones de seguridad y de integridad mencionadas en el capítulo 1. Algunas autoridades van más lejos al sugerir que el objetivo final del esquema conceptual es describir toda la empresa; no sólo los da-tos como tales, sino también la forma en que son utilizados, la forma en que fluyen de un punto a otro dentro de la empresa, su función en cada punto, lo controles de auditoría u otros que se aplican en cada punto, etcétera [2.3]. Sin embargo, debemos enfatizar que en la actualidad ningún sistema soporta realmente un esquema conceptual de cualquier cosa que se aproxime a este grado de amplitud; en la mayoría de los sistemas existentes, el "esquema conceptual" es en realidad algo más que una simple unión de todos los esquemas externos individuales más ciertas restricciones de seguridad y de integridad. Aunque en verdad es posible que los sistemas futuros sean mucho más sofisticados en cuanto al soporte del nivel conceptual.

(Date C J, 2001, pág. 39)

**44.-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 (y que seguiremos utilizando). 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 almacena-miento 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 sólo define los di-versos tipos de registros almacenados sino que especifica también qué índices existen, cómo están representados los campos almacenados, en qué secuencia están dichos registros, etcétera. El esquema interno está escrito utilizando otro lenguaje más de definición de datos: el DDL interno. Nota: En este libro usaremos normalmente los términos más intuitivos "estructura de almacenamiento" o "base de datos alma-cenada" en lugar de "vista interna", así como el término "definición de la estructura de almacena-miento" en lugar de "esquema interno".

Para terminar, señalamos que, en ciertas situaciones excepcionales, a los programas de aplicación —en particular, las aplicaciones de utilería se les podrían permitir operar directamente en el nivel interno en vez del nivel externo. Sobra decir que no es recomen-dable 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).

(Date C. , 2001, pág. 40)

**45.-MODELO CONCEPTUAL**

Los modelos de datos de alto nivelo conceptuales ofrecen conceptos muy cercanos a como muchos usuarios perciben los datos. Los modelos de datos conceptuales utilizan conceptos como entidades, atributos y relaciones.

(Ramez E. , 2007, pág. 29)

**46.-MODELO FÍSICO**

Mientras que los modelos de datos de bajo nivelo físicos ofrecen conceptos que describen los detalles de cómo se almacenan los datos en el computador. Los conceptos ofrecidos por los modelos de datos de bajo nivel están pensados principalmente para los especialistas en computadores, no para los usuarios finales normales.

(Ramez E. , 2007, pág. 29)

**48.-ALMACENAMIENTO SECUNDARIO**

Esta categoría incluye los discos magnéticos, los discos ópticos y las cintas. Las unidades de disco duro se clasifican como almacenamiento secundario, mientras que los medios removibles o extraíbles están considerados como almacenamiento terciario, Estos dispositivos normalmente tienen una gran capacidad, cuestan poco y proporcionan un acceso más lento a los datos que los dispositivos de almacenamiento principales.

(Ramez & Shamkant, 2007, pág. 390)

**49.- SISTEMA OPERATIVO**

El sistema operativo accede a los datos tras traducir las órdenes dadas por el DBMS.

(Sánchez, 2004, pág. 11)

**50.- INDEPENDENCIA LÓGICA DE DATOS.**

Es la capacidad de cambiar el esquema conceptual sin tener que cambiar los esquemas externos o los programas de aplicación. Es posible cambiar el esquema conceptual para expandir la base de datos (añadiendo un tipo de registro o un elemento de datos), para cambiar las restricciones o para reducir la base de datos (eliminando un tipo de registro o un elemento de datos). En el último caso, no deben verse afectados los esquemas externos que sólo se refieren a los datos restantes. Por ejemplo, el esquema externo de la Figura 1.5(a) no debe verse afectado por cambiar el archivo INFORME\_CALlF (o tipo de registro) de la Figura 1.2 por el mostrado en la Figura 1.6(a). Sólo es necesario cambiar la definición de la vista y los mapeados en un DBMS que soporta la independencia lógica de datos. Una vez que el esquema conceptual sufre una reorganización lógica, los programas de aplicación que hacen referencia a las estructuras de esquema externo deben funcionar como antes. En el esquema conceptual se pueden introducir cambios en las restricciones sin que se vean afectados los esquemas externos o los programas de aplicación.

(Ramez E. , independencia logica de datos, 2007, pág. 32)

**51.- INDEPENDENCIA FÍSICA DE DATOS.**

Es la capacidad de cambiar el esquema interno sin que haya que cambiar el esquema conceptual. Por tanto, tampoco es necesario cambiar los esquemas externos. Puede que haya que realizar cambios en el esquema interno porque algunos archivos físicos fueran reorganizados (por ejemplo, por la creación de estructuras de acceso adicionales) de cara a mejorar el rendimiento de las recuperaciones o las actualizaciones. Si en la base de datos permanecen los mismos datos que antes, no hay necesidad de cambiar el esquema conceptual. Por ejemplo, el suministro de una ruta de acceso para mejorar la velocidad de recuperación de los registros de sección (véase la Figura 1.2) por semestre y año no debe requerir modificar una consulta del tipo "listar todas las secciones ofrecidas en otofío de 2004", aunque el DBMS ejecutará la consulta con más eficacia utilizando la ruta de acceso nueva.

(Ramez E. , 2007, pág. 33)

**52.- SUBLENGUAJE DE DATOS**

Siempre que hay comandos DML, de alto o de bajo nivel, incrustados en un lenguaje de programación de propósito general, ese lenguaje se denomina host, y el DML sublenguaje de datos”.

(Ramez & Shamkant, 2007, pág. 34)

**53.-LDD**

Lenguaje de definición de datos (DDL, data definition lan-guage), para definir los dos esquemas. El DBMS tendrá un compilador DDL cuya función es procesar las sentencias DDL a fin de identificar las descripciones de las estructuras del esquema y almacenar la descripción del mismo en el catálogo del DBMS. En los DBMSs donde hay una clara separación entre los niveles conceptual e interno, se utiliza DDL sólo para especificar el esquema conceptual.

(Date C. , 2001, pág. 34)

**54.- LMD**

Se utiliza un lenguaje de manipulación de datos (DML) para especificar las recuperaciones y las actualizaciones de la base de datos. Los DMLs pueden ser de alto nivel (orientados a conjunto, no procedimental es) o de bajo nivel (orientados a registros, procedimentales). Un DML de alto nivel se puede incrustar en un lenguaje has y también se puede utilizar como un lenguaje independiente; en el último caso se denomina con frecuencia lenguaje de consulta.

(Date C. , 2001, pág. 34)

**55.-LDV**

El lenguaje de definición de vistas (VDL, víew definítion langllage), a fin de especificar las vistas de usuario y sus mapeados al esquema conceptual, pero en la mayoría de los DBMSs se utiliza el DDL para definir tanto el esquema conceptual como el externo. En los DBMSs relacionales se utiliza SQL actuando como VDL para definir las vistas de usuario o de aplicación como resultado de las consultas predefinida.

(Date C. , 2001, pág. 34)

**56.-LDS**

El SDL era un componente de las primeras versiones de SQL, pero se ha eliminado del lenguaje para mantenerlo únicamente en los niveles conceptual y externo.

(Date C. , 2001, pág. 34)