CONSULTA Y MANIPULACIÓN DE LOS DATOS

5.1.- Álgebra Relacional

El modelo relacional es la forma de representar los datos y manipular dicha representación considerando la integridad de los datos. Estos, en el modelo relacional se representan mediante Relaciones y un procedimiento para su manipulación es el Álgebra Relacional.

El *Álgebra relacional* es un lenguaje de consulta procedimental.

Un *lenguaje de consulta* es aquel que el usuario utiliza para solicitar información de la base de datos. Este tipo de lenguajes puede ser *procedimental* o *no procedimental*. En el primer caso, el usuario indica la secuencia de operaciones que debe realizar sobre la base de datos para obtener el resultado deseado, mientras que en un lenguaje no procedimental el usuario describe la información que desea sin dar el procedimiento para determinarla.

Para realizar las consultas a la base de datos, el álgebra relacional dispone de un conjunto de operadores y una operación de asignación.

La *operación de asignación* asigna el valor de alguna expresión del álgebra a una relación nombrada.

Los *operadores* toman una o dos relaciones como entrada y producen una nueva operación como salida.

En álgebra relacional se definen ocho tipos de operadores básicos:

- *Operaciones de Conjuntos:* **Unión, Intersección, Diferencia y Producto Cartesiano**.
- *Operaciones relacionales:* **Selección, Proyección, Reunión y División**.

Además de estos operadores, se define el operador Renombrar que permite cambiar el nombre de los atributos de una relación.

5.1.1.- Operador Renombrar (ρ)

Cambia el nombre de los atributos que sean necesarios en una relación antes de realizar una operación que pueda llevar a una relación con una cabecera en la que aparezcan dos atributos con el mismo nombre.

A partir de una relación especificada crea una nueva copia de ésta en la que sólo se han modificado los nombres de aquellos atributos que se quieren renombrar. La sintaxis, para una relación de nombre **R**, es:

$$\mathbf{R}\ \rho_{\text{Atributos Nuevos}}\ \textbf{(Atributos originales)}$$

5.1.2.- Operaciones de conjuntos

Se dice que dos Relaciones son ***compatibles respecto a la unión*** sí y sólo sí sus cabeceras son idénticas. Esto implica que:

1.- Las dos tienen el mismo conjunto de nombres de atributos.

2.- Los atributos correspondientes se definen sobre el mismo dominio.

Se dice que dos Relaciones son ***compatibles respecto al producto*** sí y sólo sí sus cabeceras son disjuntas, esto es, no contienen nombres de atributos iguales.

Unión (∪): La unión de dos relaciones R_1 y R_2 compatibles respecto a la unión es una nueva relación **R** cuya cabecera es idéntica a la de las dos relaciones y cuyo cuerpo está formado por todas las tuplas pertenecientes a R_1 , a R_2 o a las dos. Esto es, está formado por todas las tuplas que aparecen en cualquiera de las dos relaciones especificadas. Su sintaxis es:

$$R = R_1 \cup R_2$$

Intersección (∩): La intersección de dos relaciones R_1 y R_2 compatibles respecto a la unión es una nueva relación **R** cuya cabecera es idéntica a la de las dos relaciones y cuyo cuerpo está formado por todas las tuplas pertenecientes tanto a R_1 , como a R_2 o a las dos. Esto es, está formado por todas las tuplas que aparecen en las dos relaciones especificadas. Su sintaxis es:

$$R = R_1 \cap R_2$$

Diferencia (-): La diferencia de dos relaciones R_1 y R_2 compatibles respecto a la unión es una nueva relación **R** cuya cabecera es idéntica a la de las dos relaciones y cuyo cuerpo está formado por todas las tuplas pertenecientes a R_1 pero no a R_2. Esto es, está formado por todas las tuplas de la primera relación que no aparecen en la segunda. Su sintaxis es:

$$R = R_1 - R_2$$

Producto Cartesiano (x): El producto cartesiano de dos relaciones R_1 y R_2 compatibles respecto al producto es una nueva relación **R** cuya cabecera es una combinación de las cabeceras de R_1 y R_2 y cuyo cuerpo está formado por el conjunto de todas las tuplas **t** tales que **t** es la combinación de la tupla t_1 perteneciente a R_1 y la tupla t_2 perteneciente a R_2. Esto es, está formado por todas las combinaciones posibles de tuplas, una de cada una de las dos relaciones. Su sintaxis es:

$$R = R_1 \times R_2$$

5.1.2.1.- Propiedades de los operadores de conjuntos

La Unión, la intersección y el producto cartesiano gozan de la propiedad asociativa y de la conmutativa, pero no así la diferencia. Así si R_1 , R_2 y R_3 son relaciones arbitrarias (que cumplen con la compatibilidad respecto de la operación a realizar) se tendrá:

Propiedad Asociativa:

Unión .- $(R_1 \cup R_2) \cup R_3 \Leftrightarrow R_1 \cup (R_2 \cup R_3) \Leftrightarrow R_1 \cup R_2 \cup R_3$

Intersección.- $(R_1 \cap R_2) \cap R_3 \Leftrightarrow R_1 \cap (R_2 \cap R_3) \Leftrightarrow R_1 \cap R_2 \cap R_3$

Producto Cartesiano .- $(R_1 \times R_2) \times R_3 \Leftrightarrow R_1 \times (R_2 \times R_3) \Leftrightarrow R_1 \times R_2 \times R_3$

Propiedad Conmutativa:

Unión .- $R_1 \cup R_2 \Leftrightarrow R_2 \cup R_1$

Intersección.- $R_1 \cap R_2 \Leftrightarrow R_2 \cap R_1$

Producto Cartesiano .- $R_1 \times R_2 \Leftrightarrow R_2 \times R_1$

5.1.3.- Operaciones Relacionales

Selección (σ): La selección de tuplas de una relación R es otra relación con la misma cabecera que R y cuyo cuerpo está formado por las tuplas de R que verifican una *condición* entre atributos.

En la condición pueden aparecer operadores de comparación (=, <>, >=, etc.) y booleanos (and, or, not). Los atributos que aparecen en la condición deben estar definidos sobre el mismo dominio. La sintaxis del operador es:

$$\sigma_{\text{Condición}}(\mathbf{R})$$

Proyección (Π): La proyección de la Relación R según los atributos A_1, A_2, ... , A_n es otra relación que tiene por cabecera los atributos indicados y en cuyo cuerpo aparecen todas las tuplas de R restringidas a dichos atributos, eliminando tuplas repetidas. La sintaxis del operador es:

$$\Pi_{A1,\, a2,\, \ldots An}(\mathbf{R})$$

Reunión: La reunión de dos relaciones específicas es otra relación que contiene todas las posibles combinaciones de tuplas, una de cada una de las dos relaciones, tales que las dos tuplas participantes de la combinación satisfacen una condición especificada. Se consideran dos tipos de reunión:

Reunión natural (*): Sea ($\mathbf{A_1}$, $\mathbf{A_2}$,..., $\mathbf{A_n}$, $\mathbf{B_1}$, $\mathbf{B_2}$,..., $\mathbf{B_m}$) la cabecera de la relación $\mathbf{R_1}$ y ($\mathbf{B_1}$, $\mathbf{B_2}$,..., $\mathbf{B_m}$, $\mathbf{C_1}$, $\mathbf{C_2}$,..., $\mathbf{C_p}$) la cabecera de la Relación $\mathbf{R_2}$ estando los atributos del mismo nombre definidos en el mismo dominio.

Si se consideran los tres atributos compuestos **A**, **B**, **C**, la reunión natural de $\mathbf{R_1}$ y $\mathbf{R_2}$ es una relación con la cabecera (**A**, **B**, **C**) y un cuerpo formado por todas las tuplas **t** (**A:a**, **B:b**, **C:c**) tales que **t** es la combinación de la tupla $\mathbf{t_1}$ que aparece en $\mathbf{R_1}$ con el valor **a** en **A** y el valor **b** en **B**, y la tupla $\mathbf{t_2}$ que aparece en $\mathbf{R_2}$ con el valor **b** en **B** y el valor **c** en **C**. La sintaxis del operador es:

$$\mathbf{R_1 * R_2}$$

Si las dos relaciones no tienen atributos en común, la reunión natural coincide con el producto cartesiano:

$$\mathbf{R_1 * R_2} \Leftrightarrow \mathbf{R_1 \times R_2}$$

Goza de las propiedades asociativa y conmutativa:

Asociativa: $\mathbf{(R_1 * R_2)* R_3 \Leftrightarrow R_1 *(R_2 * R_3) \Leftrightarrow R_1 * R_2 * R_3}$

Conmutativa: $\mathbf{R_1 * R_2 \Leftrightarrow R_2 * R_1}$

Reunión Theta (|x|): Permite reunir dos relaciones en función de una condición diferente de la igualdad. Sean las relaciones $\mathbf{R_1}$ y $\mathbf{R_2}$ compatibles respecto al producto y sea θ un operador. La reunión theta de la relación $\mathbf{R_1}$ según el atributo **A** con la relación $\mathbf{R_2}$ según el atributo **B** es una relación con la misma cabecera que el producto cartesiano de $\mathbf{R_1}$ y $\mathbf{R_2}$ y un cuerpo formado por el conjunto de todas las tuplas **t** tales que **t** pertenece al producto cartesiano si la evaluación de la condición **A** θ **B** resulta verdadera. Los atributos **A** y **B** deben estar definidos sobre el mismo dominio y la operación θ debe ser aplicable a ese dominio. La sintaxis del operador es:

$$\sigma_{A\,\theta\,B}(R_1 \times R_2) \Leftrightarrow R_1 |x|_{A\,\theta\,B} R_2$$

División (÷): Sea (A_1, A_2,..., A_n, B_1, B_2,..., B_m) la cabecera de la relación R_1 y (B_1, B_2,..., B_m) la cabecera de la Relación R_2 estando los atributos del mismo nombre definidos en el mismo dominio.

Si se consideran los atributos compuestos **A** y **B**, la división de R_1 (dividendo) entre R_2 (divisor), es otra relación con la cabecera (**A**) y un cuerpo formado por el conjunto de todos los valores de R_1 en el atributo **A**, cuyos valores correspondientes en el atributo **B** incluyen a todos los valores del atributo **B** en la relación R_2.

La división toma dos relaciones, una binaria y otra unaria, y construye una relación formada por todos los valores de un atributo de la relación binaria que concuerdan , en el otro atributo, con todos los valores en la relación unaria. La sintaxis del operador es:

$$R_1 \div R_2$$

5.1.4.- Operaciones adicionales

Ampliación (α): Toma una relación **R** y crea otra nueva relación con un atributo mas que la original cuyos valores se obtienen evaluando alguna expresión de cálculo escalar. La sintaxis del operador es:

$$R\ \alpha_{\text{Cálculo Escalar}}\ \textbf{(nombre atributo)}$$

Resumen (Ω): Permite incorporar operaciones de agregados (cuenta, suma, promedio, máximo, mínimo, etc.). A partir de una relación **R** y de una lista de atributos, obtiene otra relación en cuya cabecera aparecen los atributos de R especificados y un nuevo atributo, con el nombre indicado, siendo los valores de éste último el resultado de evaluar la expresión de agregados. La sintaxis del operador es:

$$R\ \textbf{(lista atributos)}\ \Omega_{\text{Cálculo Agregados}}\ \textbf{(nombre atributo)}$$

División generalizada (÷): Dadas la relación R_1 con la cabecera (**A, B**) y la relación R_2 con la cabecera (**B, C**) donde los atributos **A, B, C** pueden ser compuestos, la división generalizada produce una relación que tiene como cabecera (**A, C**) y un cuerpo formado por todas las tuplas (**A:a, C:c**) tales que aparece una tupla (**A:a, B:b**) en R_1 para todas las tuplas (**B:b, C:c**) que aparecen en R_2.

Si **C** está vacío la operación se reduce a la *división* entre R_1 y R_2

Si **A** está vacío la operación se reduce a la *división* entre R_2 y R_1

Si **B** está vacío la operación degenera en el *producto cartesiano* de R_1 y R_2

Al ser la división un caso particular de la división generalizada utilizan la misma simbología. La sintaxis del operador es:

$$R_1 \div R_2$$

5.1.5.- Asignación relacional

El objetivo de esta operación es asignar, mediante una ***etiqueta***, el valor de alguna expresión algebraica. La asignación se utiliza en consultas que requieren una expresión algebraica extensa.

Etiqueta ← Expresión

5.2.- Normalización

El proceso de normalización consiste en seguir una serie de pasos o normas, al definir una Base de Datos Relacional que, tras ser aplicadas, se obtienen los datos agrupados en diferentes relaciones, de forma tal que la estructura obtenida es óptima para su implementación, gestión y aplicación desde diferentes aplicaciones futuras. Se dice que una relación está en una forma normal cuando satisface un conjunto de restricciones impuestas por dicha norma. El proceso de normalización parte de las formas normales definidas por E. F. Codd, creador de las Bases de Datos Relacionales, en 1970. Inicialmente se definieron tres formas normales (1FN, 2FN, 3FN). Debido a ciertas anomalías detectadas en ciertas bases de datos, se definió una forma normal mas completa (Forma de Boyce y Codd) y, posteriormente Fagin definió la 4FN y la 5FN.La normalización se basa en que "*los datos son independientes de las aplicaciones que los gestionan*" y su objetivo es "*obtener el mayor número de relaciones posible, dejando en cada una de ellas los atributos imprescindibles para representar a la entidad (objeto) o a la relación entre entidades a la que hace referencia la relación mediante la conexión de sus claves*" .Las ventajas que se obtienen tras la normalización de los datos para su eficaz gestión son:

- *Facilidad de uso*.- Los datos están agrupados en relaciones que identifican claramente un objeto o una relación.

- *Flexibilidad*.- La información que necesitan los usuarios puede obtenerse de las relaciones relacionales o las relaciones mediante operaciones de álgebra relacional, uniendo relaciones, seleccionando sus valores, proyectándolos, etc.

- *Precisión*.- Las interrelaciones entre relaciones consiguen mantener información diferente relacionada con toda exactitud.

- *Seguridad*.- Los controles de acceso para consultar o actualizar información (tanto en relaciones como en atributos) son mucho mas sencillos de implementar.

- *Independencia de datos*.- Los programas no están ligados a las estructuras, con lo que se consigue aumentar la base de datos añadiendo nuevos atributos o nuevas relaciones sin que afecten a los programas que las usan.

- *Claridad*.- La representación de la información es clara y sencilla para un usuario: son relaciones simples.

- *Facilidad de gestión*.- Los lenguajes manipulan la información en forma sencilla, al estar los datos basados en el álgebra y cálculo relacional.

- *Mínima redundancia*.- La información no estará duplicada innecesariamente dentro de las estructuras.

- *Máximo rendimiento de las aplicaciones*.- Sólo se trata aquella información que va a ser de utilidad en cada aplicación concreta.

El proceso de normalización se realiza después del análisis detallado del mundo real objeto de la base de datos, estableciendo las interrelaciones y las restricciones existentes entre los datos, sus agrupaciones y el modelo Entidad - Relación subsiguiente.

5.3.- Primera Forma Normal (1FN)

Se dice que una relación está en 1FN (Primera Forma Normal) si y sólo si los valores que componen cada atributo de una tupla son atómicos. Esto es: Se dice que una relación está en 1FN sí y sólo si cada atributo de la relación toma un único valor del dominio correspondiente.

5.3.1.- Dependencia Funcional

Las dependencias funcionales determinan una manera de definir restricciones en un esquema relacional.

Dada una relación R que contiene los atributos X e Y se dice que Y depende funcionalmente de X ($X \rightarrow Y$) sí y sólo sí en todo momento cada valor de X tiene asociado un solo valor de Y. Esto es lo mismo que decir que si dos tuplas de R tienen el mismo valor para su atributo X forzosamente han de tener el mismo valor para el atributo Y.

A veces, para determinar el valor de un atributo es preciso conocer el valor de varios atributos, así puede ser que si el atributo Y no depende funcionalmente del atributo X ($X \not\rightarrow Y$) y simultáneamente el atributo Y tampoco depende funcionalmente del atributo Z ($Z \not\rightarrow Y$) sin embargo Y dependa funcionalmente de la composición de los atributos X y Z ($X \bullet Z \rightarrow Y$).

Dependencias completa y parcial: Dado un atributo compuesto X formado por los atributos X1 y X2 se dice que el atributo Y tiene una dependencia funcional completa con respecto a X si:

$$X1 \not\rightarrow Y \; ; \; X2 \not\rightarrow Y \; ; \; Y \not\rightarrow X \quad \text{pero} \quad X \rightarrow Y$$

Por el contrario, dado un atributo compuesto X formado por los atributos X1 y X2 se dice que el atributo Y tiene una dependencia funcional parcial con respecto a X si:

$$X1 \rightarrow Y \quad o \quad X2 \rightarrow Y \; y \; X \rightarrow Y$$

Axiomas de Armstrong: Son un conjunto de reglas que permiten deducir implicaciones lógicas en un conjunto de dependencias funcionales.

Si se considera una relación relacional o una relación T, de la cual son atributos X, Y, Z, W:

Reflexividad: Si los valores del conjunto de atributos Y están incluidos o son iguales a un conjunto de atributos X se verifica que Y depende funcionalmente de X:

$$\text{Si } Y \subset X \Rightarrow X \rightarrow Y$$

Aumentatividad: Si el conjunto de atributos Y depende funcionalmente de X, la dependencia se mantiene si se añade a ambos conjuntos el mismo atributo:

$$\text{Si } X \rightarrow Y \Rightarrow X \bullet Z \rightarrow Y \bullet Z$$

Dependencia transitiva: Si en la relación T existen las siguientes dependencias funcionales

$$X \rightarrow Y \; ; \; Y \rightarrow Z \; ; \; Y \not\rightarrow X$$

Se dice que Z tiene una dependencia transitiva respecto X a través de Y:

$$X \rightarrow Z$$

Unión o aditividad: Si Y depende de X y además Z también depende de X la composición de Y y Z depende también de X:

Si $X \rightarrow Y$ y $X \rightarrow Z \Rightarrow X \rightarrow Y \bullet Z$

Pseudo-transitividad: Si el atributo Y depende funcionalmente de X y el atributo Z depende funcionalmente del atributo compuesto W•Y se verifica entonces que Z depende funcionalmente del atributo compuesto W•X:

$$\text{Si } \mathbf{X \rightarrow Y} \text{ y } \mathbf{W \bullet Y \rightarrow Z} \Rightarrow \mathbf{W \bullet X \rightarrow Z}$$

Descomposición o proyectividad: Si **Y** depende funcionalmente de **X** y los valores de **Z** están incluidos en **Y** entonces **Z** depende funcionalmente de **X**:

$$\mathbf{X \rightarrow Y} \text{ si } \mathbf{Z \subset Y} \Rightarrow \mathbf{X \rightarrow Z}$$

5.4.- Segunda Forma Normal (2FN)

Se dice que una relación se encuentra en 2FN si y sólo si cumple las condiciones siguientes:

- Se encuentra en la Primera Forma Normal.
- Todo atributo secundario (es decir, todo atributo de la relación que no pertenece a la clave principal) tiene una dependencia funcional completa de la clave principal completa.

Dado que la clave principal puede ser una clave compuesta, una relación no estará en la 2FN si algún atributo de ella depende funcionalmente de una parte de la clave principal pero no de la clave completa

5.4.1.- Teorema de la Segunda Forma Normal

Sea una Relación formada por los atributos **A**, **B**, **C**, **D** con clave primaria compuesta por los atributos **A** y **B**. Si se cumple que:

$$\mathbf{A \rightarrow D}$$

Entonces la Relación puede descomponerse en dos Relaciones Relación1 y Relación2 con los atributos respectivos:

RELACIÓN1 (**A**, **D**)

RELACIÓN2 (**A**, **B**, **C**)

5.4.2.- Dependencia Funcional Transitiva

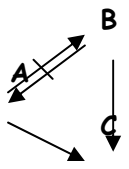
Anteriormente, dentro de los Axiomas de Armstrong, se ha establecido la dependencia funcional transitiva cuya definición formal es: Dados tres subconjuntos distintos de atributos **A**, **B** y **C** pertenecientes a una relación T, tales que cumplen las condiciones:

$$\mathbf{A \rightarrow B} \qquad \text{y} \qquad \mathbf{B \not\rightarrow A}$$

Se dice que **C** tiene una *dependencia funcional transitiva* con **A** o que es *transitivamente dependiente* de **A** si se cumple:

$$\mathbf{B \rightarrow C}$$

La representación gráfica de dicha dependencia funcional transitiva será:



Por lo tanto el atributo (o subconjunto de atributos) **C** será transitivamente dependiente del atributo (o subconjunto de atributos) **A** si puede determinarse por diferentes caminos, en particular uno directamente, y otro, mediante un atributo (o subconjunto de atributos) **B**.

5.5.- Tercera Forma Normal (3FN)

Se dice que una relación está en 3FN sí y sólo si cumple las siguientes condiciones:

- Se encuentra en la Segunda Forma Normal.
- Ningún atributo no primario es transitivamente dependiente de cada posible clave (primaria o candidatas) de la relación.

Esto quiere decir que no existe ningún atributo no principal que dependa transitivamente de alguna de las claves de la relación.

5.5.1.- Teorema de la Tercera Forma Normal

Sea una Relación formada por los atributos **A**, **B**, **C** con clave primaria formada por el atributo **A**. Si se cumple que:

$$B \rightarrow C$$

Entonces la Relación puede descomponerse en dos Relaciones Relación1 y Relación2 con los atributos respectivos:

RELACIÓN1 (**A**, **B**)

RELACIÓN2 (**B**, **C**)

5.6.- Descomposición de relaciones

La transformación de una relación que se encuentra en una determinada forma normal en otra relación cuya forma normal es superior se realiza por medio del operador *proyección* del álgebra relacional.

Así, por ejemplo la relación:

RELACIÓN(Campo1, Campo2, Campo3)

es tal que se encuentra en 1FN por que su único atributo no principal (Campo3) no depende totalmente de la clave (agregación de Campo1 y de Campo2), sino de parte de ella (por ejemplo, Campo3 solamente depende de Campo2), puede llevarse a una forma normal más avanzada descomponiéndola mediante proyecciones, obteniendo así varias relaciones:

$$\textbf{RELACIÓN1} = \Pi_{\text{Campo1, Campo2}}\ \textbf{(RELACIÓN)}$$

$$\textbf{RELACIÓN2} = \Pi_{\text{Campo2, Campo3}}\ \textbf{(RELACIÓN)}$$

Estando ambas relaciones en una forma normal superior. En concreto en 3FN ya que la combinación natural **RELACIÓN1 * RELACIÓN2** mediante el atributo común Campo2 devuelve la relación original **RELACIÓN**

5.6.1.- Descomposición sin pérdida de información

Se dice que una descomposición se ha realizado sin pérdida de información cuando la combinación natural de las proyecciones resultantes devuelve la relación original.

La condición necesaria y suficiente para que una descomposición se produzca sin pérdida de información es que el *atributo común de las dos relaciones sea clave, al menos en una de ellas*.

5.6.2.- Descomposición sin pérdida de dependencias funcionales

Las dependencias funcionales recogen la semántica del mundo real por lo que es conveniente conservarlas en el proceso de descomposición

5.6.3.- Descomposición en proyecciones independientes

La descomposición de una relación R en un conjunto de relaciones $\{R_i\}$ se dice que se ha realizado en proyecciones independientes si no ha habido pérdida de información ni pérdida de dependencias funcionales.

Se trata de la mejor descomposición ya que las relaciones resultantes son equivalentes a la relación original y, en ellas se han eliminado las anomalías de inserción, modificación y borrado.

Toda relación en 3FN puede descomponerse sin pérdida de información ni de dependencias funcionales.

5.7.- Forma Normal de Boyce – Codd

Se define como *determinante* en una Relación a un atributo del cual depende funcionalmente de manera completa cualquier otro atributo de la Relación

Se dice que una Relación está en la Forma Normal de Boyce – Codd (FNBC) si y sólo sí todo determinante de ella es una clave candidata.

En la práctica, muy pocas Relaciones que se encuentren en la 3FN no están en FNBC. Este tipo de tablas son aquellas en las que se dan las siguientes circunstancias:

Existen varias claves candidatas

Las claves candidatas son compuestas

Las claves candidatas se solapan, esto es, tienen por lo menos un atributo común.

5.7.1.- Teorema de Boyce-Codd

Sea una Relación formada por los atributos **A**, **B**, **C**, **D** con claves candidatas compuestas **(A, B)** y **(B, C)** tal que:

$$A \longleftrightarrow C$$

Entonces la Relación puede descomponerse en cualquiera de las dos siguientes maneras:

RELACIÓN1 (**A**, **C**)

RELACIÓN2 (**B**, **C**, **D**)

O bien:

RELACIÓN1 (**A**, **C**)

RELACIÓN2 (**A**, **B**, **D**)

5.8.- Dependencias multivaluadas

Dada una relación con los atributos **A**, **B**, **C** se dice que se cumple en ella una dependencia multivaluada

A → B

Si y sólo si el conjunto de valores correspondiente a un par dado (valor de **A** , valor de **C**) en la Relación depende sólo del valor de **A** y es independiente del valor de **C**.

La dependencia multivaluada **A → B** se cumple sí y sólo sí también se cumple **A → C**

5.8.1.- Cuarta Forma Normal (4FN)

Una Relación se encuentra en 4FN sí y sólo sí:

Está en FNBC

No existen dependencias multivaluadas

5.8.2.- Teorema de Fagin

Dada la relación formada por los atributos A, B, C con las siguientes dependencias multivaluadas:

A → B

A → C

Entonces la relación puede descomponerse en dos relaciones:

RELACIÓN1 (**A**, **B**)

RELACIÓN2 (**A**, **C**)

5.9.- Dependencias de Reunión

Una D*ependencia de Reunión* es una restricción en una Relación. Se dice que una Relación satisface la dependencia de reunión (X, Y, … , Z) sí y sólo sí la Relación es igual a la reunión de sus proyecciones según X, Y, … , Z . Donde X, Y, … , Z son subconjuntos del conjunto de atributos de la Relación.

5.9.1.- Quinta Forma Normal (5FN)

Una Relación se encuentra en 5FN sí y sólo sí toda dependencia de reunión en la Relación es una consecuencia de las claves candidatas, esto es, la relación estará en 5FN si está en 4FN y no existen restricciones impuestas por el creador de la Base de Datos.

5.10.- Metodología de diseño de una Base de Datos

Como ejemplo de aplicación se establece con un ejemplo la metodología a seguir en el diseño de una base de datos:

La Comunidad de Madrid desea guardar información sobre los alojamientos rurales que existen en dicha comunidad. Para ello decide crear una base de datos que recoja las siguientes consideraciones:

Un alojamiento rural se identifica por un nombre ("Villa Aurora", "Las Rosas", etc..), tiene una dirección, un teléfono y una persona de contacto que pertenece al personal del alojamiento.

En cada alojamiento trabajan una serie de personas que se identifican por un código de personal. Se requiere conocer el nombre completo, la dirección y el NIF. Aunque en el alojamiento trabajen varias personas, una persona sólo puede trabajar en un alojamiento.

Los alojamientos se alquilan por habitaciones y se desea conocer cuántas habitaciones componen el alojamiento y de qué tipo (individuales, dobles, triples) es cada una de estas habitaciones, si poseen cuarto de baño y el precio.

En alguno de estos alojamientos se realizan actividades multiaventura organizadas para huéspedes (senderismo, bicicleta de montaña, etc..). Estas actividades se identifican por un código. Es de interés saber el nombre de la actividad, la descripción y el nivel de dificultad de dicha actividad (de 1 a 10).

Estas actividades se realizan un día a la semana, por ejemplo, en la casa "Villa Aurora" se practica el senderismo los jueves y se desea guardar esta información. Pero puede haber algún día en el que no se practique ninguna actividad.

Diseñar la Base de Datos indicada de forma normalizada.

1º DISEÑO CONCEPTUAL: MODELO ENTIDAD /RELACIÓN

1 Paso: *Elaborar las listas de conceptos candidatos a ser entidades e interrelaciones indicando los conceptos que no se sabe como catalogar*. Un análisis del enunciado indica:

Entidades:	Interrelaciones:
ALOJAMIENTOS	**TRABAJA_EN**
PERSONAL	**ALQUILAN**
HABITACIONES	
ACTIVIDADES	**REALIZAN**
¿DIAS_SEMANA?	

Las entidades e interrelaciones suelen estar explícitamente indicadas en el universo definido por el cliente.

2 Paso: *Construir una matriz ENTIDADES / ENTIDADES que representen las relaciones junto con el tipo de correspondencia*. Para ello se analizan los supuestos indicados en el enunciado, así como los supuestos implícitos o de sentido común:

Supuestos del enunciado:

En un ALOJAMIENTO pueden TRABAJAR varias PERSONAS

Cada ALOJAMIENTO puede ALQUILAR varias HABITACIONES

Cada ALOJAMIENTO puede REALIZAR varias ACTIVIDADES

Supuestos implícitos

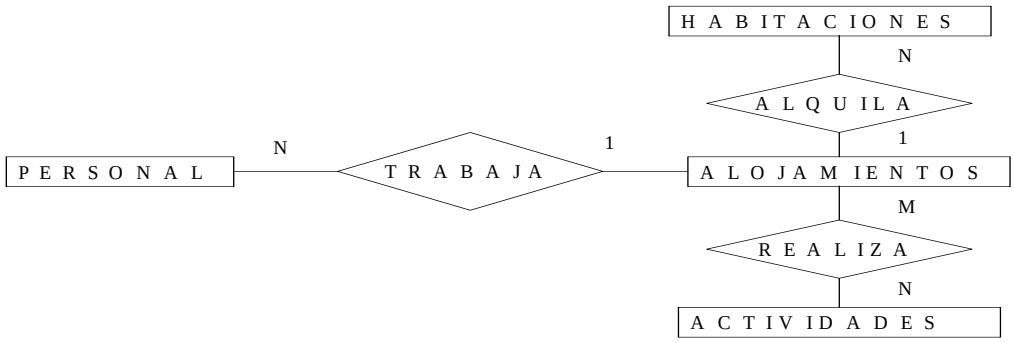
Ninguno

La matriz obtenida será:

	ALOJAMIENTOS	PERSONAL	HABITACIONES	ACTIVIDADES
ALOJAMIENTOS	-	X	ALQUILAN (1:N)	REALIZAN (M:N)
PERSONAL	TRABAJA EN	-	X	X

	(1:N)			
HABITACIONES	X	X	-	X
ACTIVIDADES	X		X	-

3 Paso *Realizar una versión preliminar del diagrama* ENTIDAD / RELACIÓN:



4 Paso *Análisis de las Cardinalidades Mínimas*. Anteriormente se han definido las cardinalidades máximas. En cuanto a las cardinalidades mínimas se tiene:

TRABAJAN: Una PERSONA sólo puede trabajar en un ALOJAMIENTO

ALQUILAN: Una HABITACIÓN sólo puede estar ALQUILADA (pertenece) a un ALOJAMIENTO.

REALIZAN una ACTIVIDAD: puede realizarse en varios ALOJAMIENTOS.

5 Paso *Análisis de redundancias*. En el ejemplo anterior al no existir ciclos en el diagrama ENTIDAD / RELACION no existen problemas de redundancias.

Por tanto puede indicarse que el diagrama ENTIDAD / RELACION establecido es válido.

2º DISEÑO LÓGICO: MODELO RELACIONAL

1 Paso: *Identificación de las relaciones básicas*. De la lectura del enunciado

Un alojamiento rural se identifica por un nombre ("Villa Aurora", "Las Rosas", etc..), tiene una dirección, un teléfono y una persona de contacto que pertenece al personal del alojamiento.

ALOJAMIENTOS (Nombre-A, Dirección, Teléfono, Contacto*)

En principio Nombre-A identifica la clave primaria y Contacto se señala con * indicando que puede tener valores nulos, permitiendo la posibilidad de que, cuando se dé de alta un nuevo alojamiento no esté definida todavía la persona asignada.

En cada alojamiento trabajan una serie de personas que se identifican por un código de personal. Se requiere conocer el nombre completo, la dirección y el NIF. Aunque en el alojamiento trabajen varias personas, una persona sólo puede trabajar en un alojamiento.

PERSONAL (Código-P, Nombre-P, Dirección, *NIF*)

Se asigna como clave primaria a Código-P y como clave alternativa a *NIF*.

Los alojamientos se alquilan por habitaciones y se desea conocer cuántas habitaciones componen el alojamiento y de qué tipo (individuales, dobles, triples) es cada una de estas habitaciones, si poseen cuarto de baño y el precio.

HABITACIONES (Nombre-A, Nº-Hab, Tipo, Baño*, Precio*)

La clave primaria será el número de habitación "Nº-Hab" junto con el alojamiento "Nombre-A" donde está ubicada. Como puede darse el caso de que no se suministre información sobre si tiene baño o no y sobre su precio, ambos atributos pueden ser nulos. Además,

se desea conocer cuántas habitaciones componen el alojamiento

por lo que será preciso ampliar la relación ALOJAMIENTOS con el atributo Número-H que indica el número de habitaciones que hay en el alojamiento:

ALOJAMIENTOS (Nombre-A, Dirección, Teléfono, Contacto*, Número-H)

En alguno de estos alojamientos se realizan actividades multiaventura organizadas para huéspedes (senderismo, bicicleta de montaña, etc..). Estas actividades se identifican por un código. Es de interés saber el nombre de la actividad, la descripción y el nivel de dificultad de dicha actividad (de 1 a 10).

Este párrafo establece la necesidad de la relación

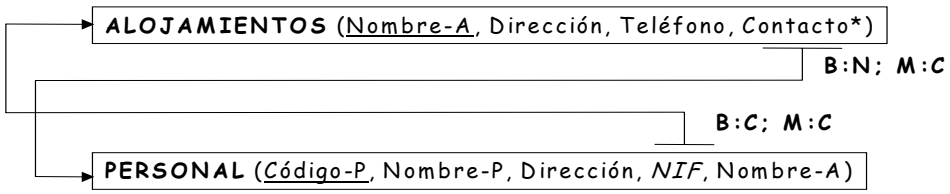
ACTIVIDADES (Código-Ac, Nombre-Ac, Descripción, Nivel)

Restricciones semánticas.

Un alojamiento rural se identifica por un nombre ("Villa Aurora", "Las Rosas", etc..), tiene una dirección, un teléfono y una persona de contacto que pertenece al personal del alojamiento.

En cada alojamiento trabajan una serie de personas pero una misma persona sólo puede trabajar en un alojamiento por lo que debe introducirse el atributo Nombre-A en la relación PERSONAL que será clave ajena de esta relación y referenciará a la relación ALOJAMIENTOS.

Además la persona de contacto pertenece al alojamiento por lo que "Contacto" será en la relación ALOJAMIENTOS clave ajena que referenciará a la relación PERSONAL



Opciones de Borrado y Modificación.-

Si se elimina un alojamiento de la base de datos, automáticamente ha de eliminarse todo el personal que trabaja en él, el borrado es por tanto en Cascada (B:C) y lo mismo ocurre con las modificaciones (M:C)

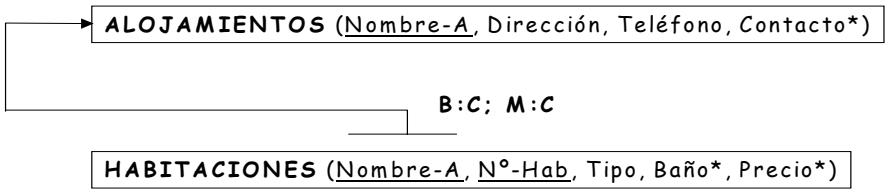
Sin embargo si se elimina una persona, la relación ALOJAMIENTOS no debe verse afectada y si la persona eliminada es la de contacto, como el atributo contacto admite valores nulos, puede considerarse el borrado como puesta a nulos (B:N) . La modificación sin embargo debe hacerse en cascada, pues si se modifica la persona de contacto en PERSONAL, debe modificarse también la relación ALOJAMIENTOS (M:C)

Otras reglas no contempladas.-

Debe controlarse en la definición de la clave el hecho de que en un alojamiento trabaje al menos una persona.

Los alojamientos se alquilan por habitaciones y se desea conocer cuántas habitaciones componen el alojamiento

Los alojamientos disponen de una serie de habitaciones por lo que éstas dependen de los alojamientos en el sentido de que para hacer referencia a una habitación es preciso conocer el nombre del alojamiento donde está incluída. Nombre-A es por tanto la clave ajena de la relación HABITACIONES que referencia a la relación ALOJAMIENTOS:



Opciones de borrado y modificación.-

Si se elimina o modifica un alojamiento deben eliminarse o modificarse todas las habitaciones de dicho alojamiento por lo que tanto el borrado como la modificación han de ser en cascada (B:C; M:C).

Otras reglas no contempladas.-

Como se desea conocer el número de habitaciones que tiene cada alojamiento debe contemplarse el hecho de que cada vez que se dé de alta o se elimine una habitación, automáticamente se incremente o decremente en uno el valor del atributo Número-H.

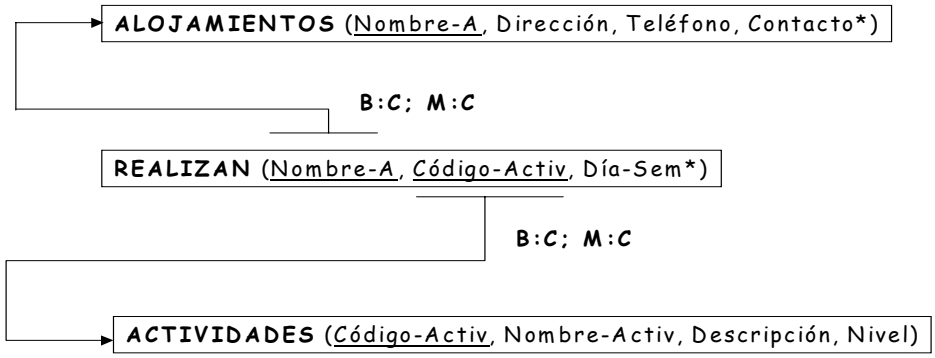
En alguno de estos alojamientos se realizan actividades multiaventura organizadas para huéspedes (senderismo, bicicleta de montaña, etc..).

Estas actividades se realizan un día a la semana, por ejemplo, en la casa "Villa Aurora" se practica el senderismo los jueves y se desea guardar esta información. Pero puede haber algún día en el que no se practique ninguna actividad.

Dado que cada alojamiento puede proporcionar varias actividades multiaventura y además una misma actividad puede desarrollarse en varios alojamientos, existirá una nueva relación: REALIZAN cuya clave primaria será la composición de las claves primarias de ALOJAMIENTOS y de ACTIVIDADES. Cada una a su ve será clave ajena que referencia a la relación de que proceden. Además

Estas actividades se realizan un día a la semana

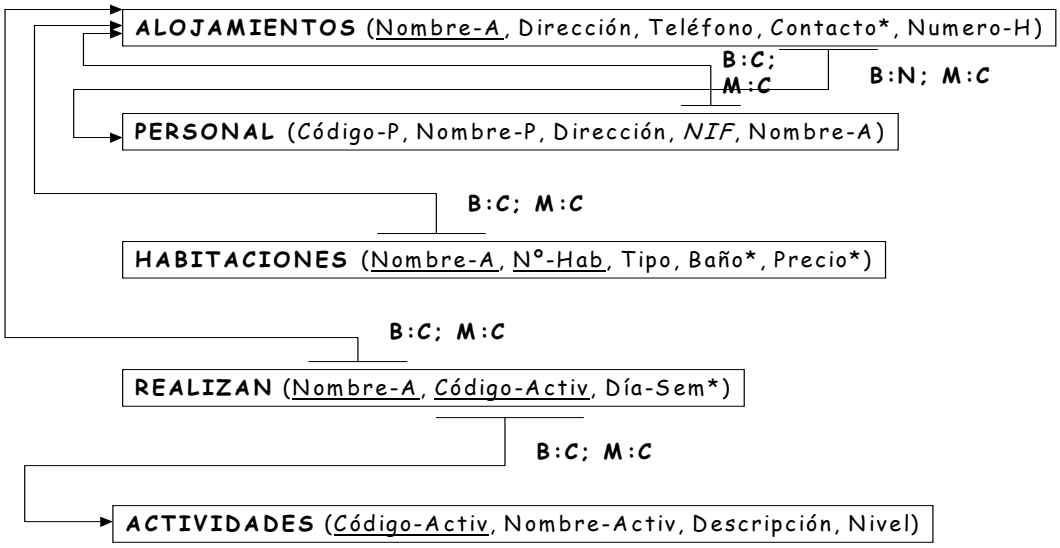
Debe introducirse el atributo "Día-Sem" en la relación REALIZAN este atributo puede tener valores nulos para contemplar el hecho de que una actividad propuesta en un cierto alojamiento todavía no esté configurada.



Opciones de borrado y modificación.-

Cada vez que se elimine o se modifique una actividad se debe eliminar o modificar la tupla correspondiente de la relación intermedia, por lo tanto, tanto el borrado como la modificación será en cascada.

En resumen, el esquema relacional será el siguiente:



3º NORMALIZACION DE LAS RELACIONES

1 Primera Forma Normal

Las cinco relaciones se encuentran en la 1FN dado que los valores de todos sus respectivos atributos son atómicos.

2 Dependencias funcionales.

ALOJAMIENTOS (Nombre-A, Dirección, Teléfono, Contacto*, Número-H)

Nombre-A → Dirección

Nombre-A → Teléfono

Nombre-A → Contacto

Nombre-A → Número-H

Todos los atributos dependen directamente de la clave primaria

PERSONAL (Código-P, Nombre-P, Dirección, *NIF,* Nombre-A)

Código-P → Nombre-P

Código-P → Dirección

Código-P ↔ *NIF*

Código-P → Nombre-A

Todos los atributos dependen directamente de la clave primaria

HABITACIONES (Nombre-A, Nº-Hab, Tipo, Baño*, Precio*)

(Nombre-A, Nº-Hab) → Tipo

(Nombre-A, Nº-Hab) → Baño

(Nombre-A, Nº-Hab) → Precio

Todos los atributos dependen directamente de la clave primaria, la cual es compuesta

REALIZAN (Nombre-A, Código-Activ, Día-Sem*)

(Nombre-A, Código-Activ) → Sia-Sem

Todos los atributos dependen directamente de la clave primaria, la cual es compuesta

ACTIVIDADES (Código-Activ, Nombre-Activ, Descripción, Nivel)

Código-Activ → Nombre-Activ

Código-Activ → Descripción

Código-Activ → Nivel

Todos los atributos dependen directamente de la clave primaria

3 Recubrimiento Mínimo

Para cada relación debe determinarse que el conjunto de dependencias del apartado anterior son dependencias elementales, no existe ningún atributo extraño en ninguna de ellas y no existen dependencias redundantes. Para ello

Dependencias elementales

Debe comprobarse que:

- *En todas las dependencias se cumple que tienen un único atributo implicado*
- *Todas las dependencias son plenas, esto es, ningún subconjunto del atributo implicante (determinante) puede implicar al atributo implicado*
- *Ninguna dependencia es trivial, es decir, ningún atributo del determinante aparece como implicado*

Todas las dependencias son elementales.

Atributos Extraños

Se dice que el atributo A es extraño en la dependencia (X,A) → Y cuando se cumple que X → Y sin que A tome parte de la dependencia.

Ninguna dependencia de las indicadas anteriormente tiene atributos extraños.

Dependencias Redundantes

Para cada una de las relaciones indicadas no existen dependencias redundantes

Por tanto, el recubrimiento mínimo de cada relación es el conjunto de las dependencias indicadas anteriormente.

4 Claves Primarias

Para cada relación se han definido ya las claves primarias

5 Segunda Forma Normal

Para cada relación todos los atributos tiene una dependencia funcional completa de su respectiva clave, por lo que las cinco relaciones están en 2FN

6 Tercera Forma Normal

Para cada relación no existen atributos que tengan una dependencia funcional transitiva de su respectiva clave, por lo que las cinco relaciones están en 3FN

7 Forma Normal de Boyce Codd

Unicamente la relación **PERSONAL** no está en la FNBC ya que el atributo NIF determinante que no forma parte de la clave primaria, por lo que, en virtud del teorema de Boyce Codd puede descomponerse en dos recomponerse en dos relaciones:

PERSONAL (Código-P, Nombre-P, Dirección, *NIF,* Nombre-A)

PERSONAL-ASIG (Código-P, Nombre-A)

PERSONAS (Código-P, Nombre-P, Dirección, NIF,)

4º IMPLEMENTACION DE LA BASE DE DATOS

1 Creación de dominios.

```
CREATE DOMAIN Nombre_Valido    CHAR(30);
CREATE DOMAIN Tipo_Codigo      CHAR(3);
CREATE DOMAIN Tipo_Bano        CHAR(2);
CHECK (VALUE IN ("SI","NO"));
```

2 Creación de tablas

```
CREATE TABLE Alojamientos (
Nombre_A      Nombre_Valido,
Direccion     CHAR(25)          NOT NULL,
Telefono      CHAR(9)           NOT NULL,
Contacto      Tipo_Codigo,
Numero_H      INTEGER           NOT NULL,
PRIMARY KEY (Nombre_A)
);

CREATE TABLE Personal_Asig (
Codigo_P      Tipo_Codigo,
Nombre_A      Nombre_Valido     NOT NULL
PRIMARY KEY (Codigo_P),
FOREIGN KEY (Nombre_A) REFERENCES Alojamiento
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Personas (
Codigo_P      Tipo_Codigo,
Nombre_P      Nombre_Valido     NOT NULL,
Direccion     CHAR(25)          NOT NULL,
NIF           CHAR(10)          NOT NULL
PRIMARY KEY (NIF),
FOREIGN KEY (Nombre_P) REFERENCES Personal_Asig
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

ALTER TABLE Alojamientos
ADD FOERIGN KEY (Contacto) REFERENCES Personal-Asig
ON DELETE SET NULL
ON UPDATE CASCADE;

CREATE TABLE Habitaciones (
Nombre_A      Nombre_Valido,
No_Hab        Integer(3)        NOT NULL,
Tipo          Char(1)           NOT NULL,
Bano          Tipo_Bano,
Precio        INTEGER,
PRIMARY KEY (Nombre_A, No_Hab),
FOREIGN KEY (Nombre_A) REFERENCES Alojamientos
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

CREATE TABLE Actividades (
```

```
Codigo_Activ   Tipo_Codigo,
Nombre_Activ Nombre_Valido        NOT NULL,
Descripcion    CHAR(50)           NOT NULL,
Nivel          CHAR(2)            NOT NULL,
PRIMARY KEY (Codigo_Activ),
CHECK (Nivel BETWEEN 1 AND 10)
);

CREATE TABLE Realizan (
Nombre_A       Nombre_Valido,
Codigo_Activ   Tipo_Codigo,
Dia_Sem        CHAR(10)
PRIMARY KEY (Codigo_Activ, Nombre_A),
FOREIGN KEY (Codigo_Activ) REFERENCES Actividades
        ON UPDATE CASCADE
        ON DELETE CASCADE,
FOREIGN KEY (Nombre_A) REFERENCES Alojamientos
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

2 Consultas Principales:

Nombre y descripción de las actividades que se realizan en el alojamiento denominado "La Huerta"

SQL

```
SELECT A.Nombre_Activ, A.Descripcion
FROM Actividades A, Realizan B
WHERE Nombre_A = "La Huerta" AND A.Codigo_Activ = B.Codigo_Activ
```

Algebra Relacional

$\Pi_{Nombre\_Activ,\ Descripcion}((ACTIVIDADES) * \Pi_{Codigo\_Activ}(\sigma_{Nombre\_Activ= \text{"La Huerta"}}(REALIZAN))$

Nombre de los alojamientos que tienen habitaciones dobles y realizan actividades de senderismo

SQL

```
SELECT DISTINCT Nombre_A
FROM Actividades A, Realizan B, Habitaciones C
WHERE A.Nombre_A = "Senderismo" AND C.Tipo = "Doble"  AND
(B.Codigo_Activ = A.Codigo_Activ AND B.Nombre.A = C.Nombre_A)
```

Algebra Relacional

$\Pi_{Nombre\_A}(\sigma_{Tipo="Doble"}(HABITACIONES)) * (\Pi_{Nombre\_A}((REALIZAN) * \Pi_{Codigo\_Activ}(\sigma_{Nombre\_Activ= \text{"Senderismo"}}(ACTIVIDADES)))$

5.11.- Concepto y objetivos de los SGBD

Los sistemas informáticos tradicionales se denominan **sistemas orientados hacia procesos** debido a que, en ellos, el interés se centra mas en los tratamientos que reciben los datos (los cuales se almacenan en ficheros que son diseñados para una determinada aplicación).

Este planteamiento produce, además de una ocupación inútil de la memoria secundaria, un aumento en el tiempo de proceso al repetirse los mismos controles y operaciones con los distintos ficheros. Además pueden producirse graves inconsistencias en dichos sistemas como consecuencia de actualizar los mismos datos, cuando se encuentran en distintos ficheros, de forma no simultánea en todos ellos.

Por otro lado, la dependencia de los datos respecto del soporte físico que los contiene y de los programas que los utilizan da lugar a una falta de flexibilidad y de adaptabilidad frente a posibles cambios, lo que repercute en el rendimiento del sistema informático.

Frente a esta situación, y para lograr una gestión mas racional del conjunto de los datos surge un nuevo enfoque del tratamiento de los datos, que se apoya en el concepto de "*base de datos*", en la cual éstos son recogidos y almacenados, al menos lógicamente, una sola vez, con independencia de los tratamientos que se efectúen sobre ellos.

Las ventajas de los sistemas de bases de datos son, entre otras, las siguientes:

Independencia de los datos respecto a los tratamientos y viceversa, lo que evita un importante esfuerzo de reprogramación de las aplicaciones cuando se producen cambios en los datos.

Coherencia de los resultados, con lo que se elimina el inconveniente de las divergencias en los resultados debidas a actualizaciones no simultáneas de todos los ficheros-

Mejor disponibilidad de los datos para el conjunto de los usuarios junto con una mayor "transparencia" respecto a la información existente.

Mayor valor informativo debido a que los distintos elementos están interrelacionados.

Documentación de la información mejor y mas normalizada, dado que ésta se encuentra integrada en los datos.

Mayor eficiencia en la recogida, validación y entrada de los datos al sistema.

Por su parte, las bases de datos presentan una serie de inconvenientes que deben valorarse antes de tomar una decisión relativa a un cambio de orientación en el sistema de información (SI). Entre estos inconvenientes destacan:

Instalación costosa

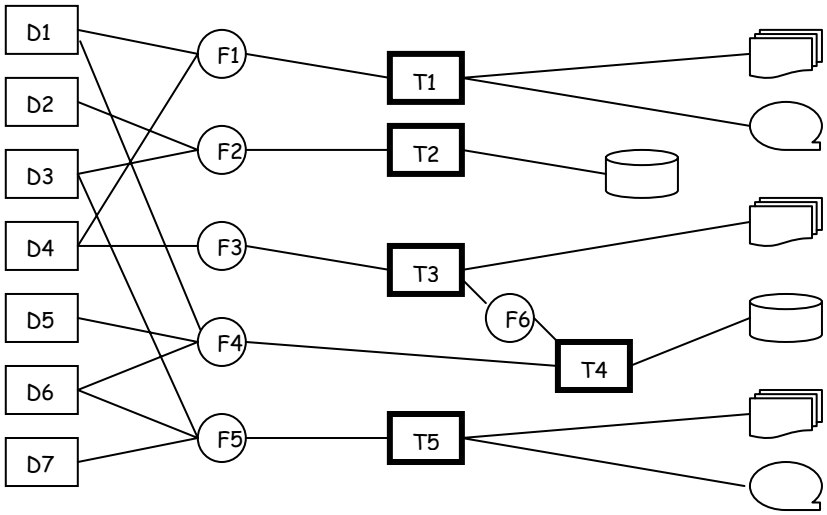
Personal especializado

Falta de rentabilidad a corto plazo

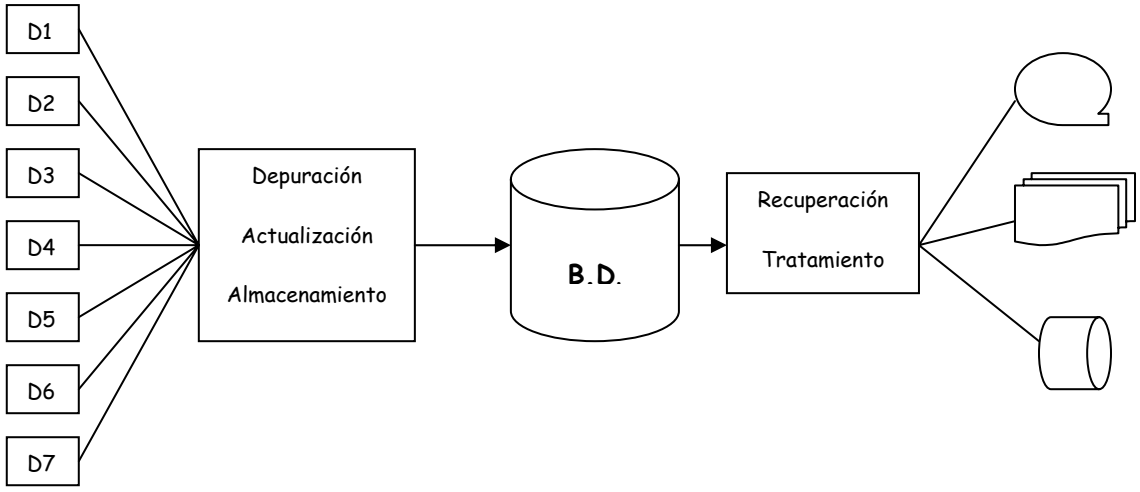
Desfase entre teoría y práctica

Las bases de datos no constituyen pues únicamente una nueva tecnología mas o menos avanzada, sino que nacen de una concepción distinta de los sistemas de información y, por ello, tienen una influencia decisiva en las estructuras y en la organización de su entorno. Si no se tiene en cuenta esta premisa, la mayoría de las ventajas de las bases de datos no se harán realidad y se acentuarán sus inconvenientes y problemas.

De acuerdo con lo que se acaba de indicar, puede definirse la base de datos como " una colección o depósito de datos integrados, con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción (única para cada tipo de datos) han de estar almacenadas junto con los mismos. Los procedimientos de actualización y recuperación, comunes y bien determinados, habrán de ser capaces de conservar la seguridad (integridad, confidencialidad y disponibilidad) del conjunto de los datos"



Sistemas orientados hacia procesos

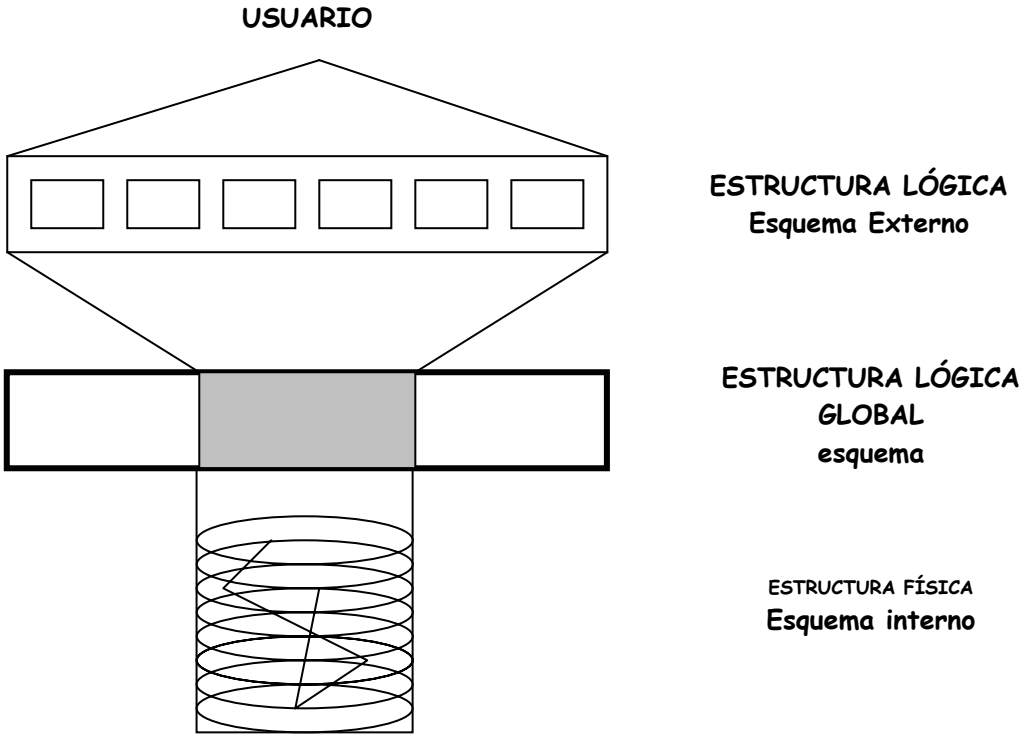


Sistemas de Bases de datos

5.12.- Niveles de abstracción de una Base de Datos

En los sistemas de información se puede observar la existencia de dos estructuras distintas:

- *Lógica, vista por el usuario*
- *Física, forma en la que se encuentran los datos en el sistema de almacenamiento*



En las bases de datos aparece un nuevo nivel de abstracción que se denomina de distintas maneras: *nivel conceptual, estructura lógico global, esquema,* etc. . Esta estructura intermedia pretende una representación global de los datos interpuesta entre las estructuras física y lógica y que es independiente, tanto del equipo como de cada usuario en particular.

Estos tres niveles de abstracción en una base de datos corresponden a otras tantas maneras distintas de contemplar ésta. La estructura lógica de usuario o esquema externo es la visión que de la base de datos tiene cada usuario en particular, la estructura lógica global responde al enfoque del conjunto de la empresa (lo que podría ser la visión del administrador de la base de datos) y la estructura física o esquema interno es la forma en la que se organizan los datos en el almacenamiento físico.

5.12.1.-Estructura Lógica de Usuario (Esquema Externo)

Debido a que un esquema externo es la visión que tiene de la base de datos un usuario en particular, en dicho esquema deberán encontrarse reflejados solo aquellos datos e interrelaciones que necesite dicho usuario. También habrán de especificarse las restricciones de uso (insertar o borrar datos, acceso a los mismos, etc.).

5.12.2.- Estructura Lógica Global (Esquema Conceptual)

Por ser el esquema conceptual la visión global de los datos, deberá incluirse la descripción de todos y cada uno de los datos e interrelaciones entre éstos, las restricciones de integridad y confidencialidad, etc.

5.12.3.- Estructura Física (Esquema Interno)

Aunque el contenido del esquema interno es muy dependiente de cada SGBD pueden distinguirse tres clases de aspectos que deben especificarse en él:

Estrategia de almacenamiento. Apartado que incluye la determinación del espacio de almacenamiento para el conjunto de los datos, así como las relaciones entre los distintos espacios de almacenamiento. La estrategia de emplazamiento de los datos seguida para optimizar tiempo y espacio de memoria secundaria así como otros aspectos, como el tratamiento de los desbordamientos, etc.

Caminos de acceso. Se incluyen en este apartado la especificación de claves primarias y secundarias así como los índices o punteros e incluso claves de ordenación.

Miscelánea. Además de los aspectos citados habría que incluir otros varios como técnicas de compresión de datos, criptografiado de datos, traslación o correspondencia del esquema interno al esquema conceptual, técnicas de ajuste y de optimización, etc.

5.13.- El Sistema Gestor de la Base de Datos

La base de datos, como depósito único de los datos de toda la organización, debe ser capaz de atender las necesidades de los distintos tipos de usuarios que interactúan con ella. Puede definirse un Sistema Gestor de la Base de Datos (SGBD) como: "*Un conjunto coordinado de programas, procedimientos, lenguajes, etc., que suministra, tanto a los usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos almacenados en la base, manteniendo su seguridad*".

Debido a la diversidad usuarios con necesidades cambiantes a lo largo del tiempo, es imprescindible dotar al sistema de la adecuada flexibilidad para que pueda atender las exigencias de todos los usuarios y para que sea capaz de responder a los cambios a un coste no excesivo, es decir, el SGBD debe ser diseñado de forma tal que optimice las ventajas que se han indicado para una base de datos.

Las funciones esenciales de un SGBD son:

Función de descripción o de definición. Esta función debe permitir al administrador de la base especificar los elementos de datos que la integran , su estructura, las relaciones que existen entre ellos, las reglas de integridad semántica, los controles a efectuar antes de autorizar el acceso a la base, etc. Esta función se lleva a cabo mediante el Lenguaje de Descripción o de Definición de Datos (LDD) propio de cada SGBD y debe suministrar los medios para definir las tres estructuras de datos – externa, lógica global e interna -, especificando las características de los datos a cada uno de estos niveles.

Función de manipulación. Permite a los usuarios de la Base buscar, añadir, suprimir o modificar los datos de la misma, siempre de acuerdo con las especificaciones y las normas de seguridad establecidas por el administrador. Esta función se realiza mediante el Lenguaje de Manipulación de Datos (LMD) que facilita los instrumentos necesarios para la realización de estas tareas.

Función de utilización. Reúne todas las interfaces que necesitan los diferentes usuarios para comunicarse con la base y proporciona un conjunto de procedimientos para el administrador entre los que se encuentra el Lenguaje de Control de Datos (LCD). Además, en la mayoría de los SGBD existentes en el mercado existen funciones de servicio, como cambiar la capacidad de los ficheros, obtener estadísticas de utilización, cargar archivos, etc., y otras relacionadas con la seguridad física - copias de seguridad, rearranque en caso de caída del sistema, etc. – y protección frente a accesos no autorizados.

En resumen, en el cuadro adjunto se presentan las funciones esenciales de un Sistema Gestor de la Base de Datos (SGBD):

DESCRIPCIÓN

***Permite describir*:**
Los elementos de la base de datos con
Su estructura
Sus Interrelaciones
Sus Validaciones
A tres niveles
Externo
Lógico Global
Interno
Mediante un Lenguaje de Definición de Datos (LDD)

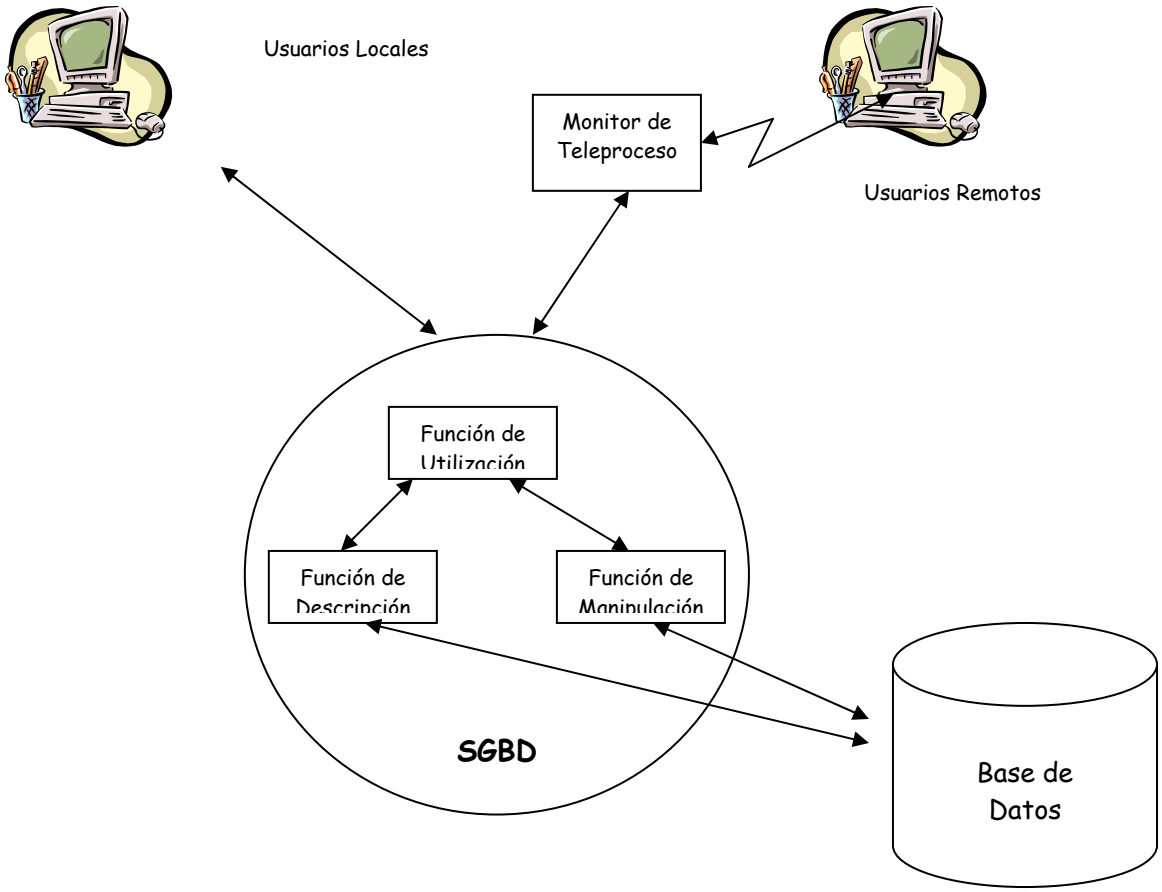
MANIPULACIÓN

Permite
Buscar
Añadir
Suprimir
Modificar
Datos de la base
Mediante un Lenguaje de Manipulación de Datos (LMD)
Lo cual supone
Definir un criterio de selección (responsabilidad del usuario)
Definir la estructura externa a recuperar (responsabilidad del usuario)
Acceder a la estructura física (responsabilidad del sistema)

CONTROL

Reúne las Interfaces de los Usuarios
Suministra procedimientos para el Administrador
Mediante un Lenguaje de Control de Datos (LCD)

Las interrelaciones existentes entre estas funciones, el conjunto de los datos y los usuarios se encuentra representado en la figura adjunta.



5.14.- Interacción del usuario con el SGBD: Lenguajes

Debido a las distintas funciones a las distintas funciones a realizar por el SGBD se hace necesario disponer de diferentes lenguajes y procedimientos que permitan la comunicación con la base de datos, tanto dirigidos a las funciones indicadas (definición, manipulación o control) como dirigidos a los diferentes tipos de usuarios a de procesos a realizar.

La tipología de los lenguajes de un SGBD se expresa en el cuadro adjunto:

Por tipo de función { Definición; Manipulación

Por tipos de Usuarios y de Aplicaciones { Informáticos; Finales { Aplicaciones formalizables; Aplicaciones no formalizables

Como se ve, las distintas características del proceso y del usuario determinan el tipo de lenguaje a realizar. En general, los usuarios informáticos, como el diseñador de la base, el administrador, analistas, programadores, etc., requerirán medios potentes y flexibles con los cuales consigan definir, administrar , extraer o manipular los datos de la base.

Normalmente se apoyarán en un lenguaje de programación que están habituados a manejar ("***Lenguaje Anfitrión***"), para lo cual deberá permitir hacer llamadas desde un programa de aplicación al SGBD.

El conjunto de sentencias de manipulación del SGBD que pueden ser llamadas desde un lenguaje de programación permitiendo el acceso a la base de datos, se suele denominar ***sublenguaje de datos*** o también ***lenguaje huésped*** o ***lenguaje embebido***.

Los SGBD admiten , en general, varios lenguajes de tipo anfitrión para manipular datos (Cobol, Ensamblador, Fortran, PL/I, Basic, Pascal, C, etc,.) . Así mismo, la práctica totalidad de los SGBD admiten lenguajes de 4ª generación que permiten el acceso a la base de datos, mediante sentencias embebidas en dicho lenguaje y escritas en un lenguaje de datos como SQL.

El usuario final, por su parte, requerirá medios simples para comunicarse con la base, lo que puede conseguirse mediante un lenguaje de manipulación autocontenido, que tenga una sintaxis sencilla, pero potente como para soportar demandas de información muy variadas o por medio de tratamientos parametrizados que suelen presentarse al usuario en forma de *menús*.

La estructura y la sintaxis de estos tipos de lenguajes dependen de cada SGBD. Para modelo de datos en red, las normas Codasyl proponen especificaciones concretas de la sintaxis para los lenguajes de descripción y manipulación de los datos. Para modelos de datos relacionales el SQL es un estándar muy extendido que proporciona estas facilidades.

5.14.1.- Lenguajes de Definición de Datos

Los instrumentos que permiten al administrador de la BD describir los datos con facilidad y precisión, especificando sus distintas estructuras es lo que se denomina *Lenguaje de Definición de Datos (LDD)* . Suelen ser lenguajes autocontenidos y no necesitan apoyarse en ningún lenguaje de programación. El SGBD deberá facilitar los medios para describir la estructura lógica global, para hacer las especificaciones relativas a la estructura interna y para declarar las estructuras externas que sean requeridas para el desarrollo de distintas aplicaciones.

5.14.1.1.- Lenguajes de definición de la estructura lógica global

Desde el punto de vista lógico global el administrador debe disponer de un instrumento de descripción que permita asignar nombres a los campos, a los agregados de datos, a los registros, etc. estableciendo sus longitudes y sus características así como las relaciones entre estos elementos, especificar los identificadores e indicar restricciones semánticas que se han de aplicar a los diferentes objetos descritos.

5.14.1.2.- Lenguajes de definición de la estructura lógica interna

En teoría, el propio SGBD debería conseguir automáticamente la optimización del almacenamiento y recuperación de los datos y encargarse, a partir de la estructura lógica global, de definir la estructura interna adecuada sin intervención del usuario (administrador).

Para ello, habría que suministrar al SGBD las informaciones precisas sobre volúmenes, crecimiento previsto, tipos de registros mas accedidos, con indicaciones del número medio de accesos, relación entre actualizaciones y consultas, etc.

En la práctica, puede mejorarse sensiblemente la eficiencia si el administrador especifica características respecto a la estructura física, por lo que deberá disponer de un

lenguaje de definición de la estructura interna o, simplemente, deberá dar valores a ciertos parámetros.

En muchos SGBD se suministra automáticamente por defecto una estructura interna, que es la que el sistema considera mas adecuada para la estructura lógica global definida, aunque el administrador deberá ajustar posteriormente dicha estructura interna para conseguir una mayor eficiencia.

5.14.1.3.- Lenguajes de definición de las estructuras externas

El SGBD debe poner a disposición de los usuarios los medios necesarios para recuperar o actualizar los datos contenidos en la base de datos, de acuerdo con la visión lógica o estructura externa (vista) que precise cada aplicación.

Al definir una estructura externa es preciso darle un nombre e indicar qué datos y qué interrelaciones de la estructura lógica global se encontrarán en la misma. Cuando se desee utilizar un esquema externo ya definido se podrá hacer referencia al mismo invocando su nombre desde el lenguaje de manipulación.

5.14.2.- Lenguajes de manipulación de datos

Para cumplir los objetivos asignados a la función de manipulación debe disponerse de lenguajes que ofrezcan a los usuarios la posibilidad de referirse a determinados conjuntos de datos, que cumplan ciertas condiciones (*criterio de selección*) como que un atributo que tenga un determinado valor, o un conjunto de atributos y valores que satisfagan cierta expresión lógica. Además del criterio de selección, es preciso indicar la estructura externa que se desea actualizar o recuperar.

Una vez especificados el criterio de selección y los datos a actualizar o recuperar el SGBD debe ocuparse de acceder al correspondiente soporte físico de donde se extraerán los datos definidos para su transferencia a un dispositivo de salida , o, si se trata de una actualización, en donde se insertarán, modificarán o borrarán los datos.

Pero al igual que el programador precisa de un lenguaje de manipulación que se embeba en un lenguaje de programación, el usuario no informático deberá disponer de también de un instrumento análogo (mucho mas sencillo) que le permita comunicarse con la base y extraer de ella o introducir en ella las informaciones que precise. Para ello, los SGBD suelen disponer de lenguajes autocontenidos para que, desde un terminal y en modo interactivo, el usuario pueda acceder a la base y manipular los datos almacenados en ella sin necesidad de apoyarse en un lenguaje de programación.

La mayoría de los SGBD utilizan como lenguaje de manipulación de datos el Lenguaje Estructurado de Consultas o SQL.

Atendiendo a su utilización, los LMD pueden ser *procedmentales* (lenguajes en los que, además de *qué* se quiere es preciso indicar el algoritmo que establece *cómo* obtenerlo) o *no procedimental* (lenguaje en el que basta decir *qué* se quiere sin explicar *cómo* se obtiene)

Atendiendo a la forma en que se recuperan o actualizan datos, los LMD pueden ser *Navegacionales* (recuperando o actualizando los datos registro a registro) o *Especificacionales* (actuando sobre conjuntos de registros)

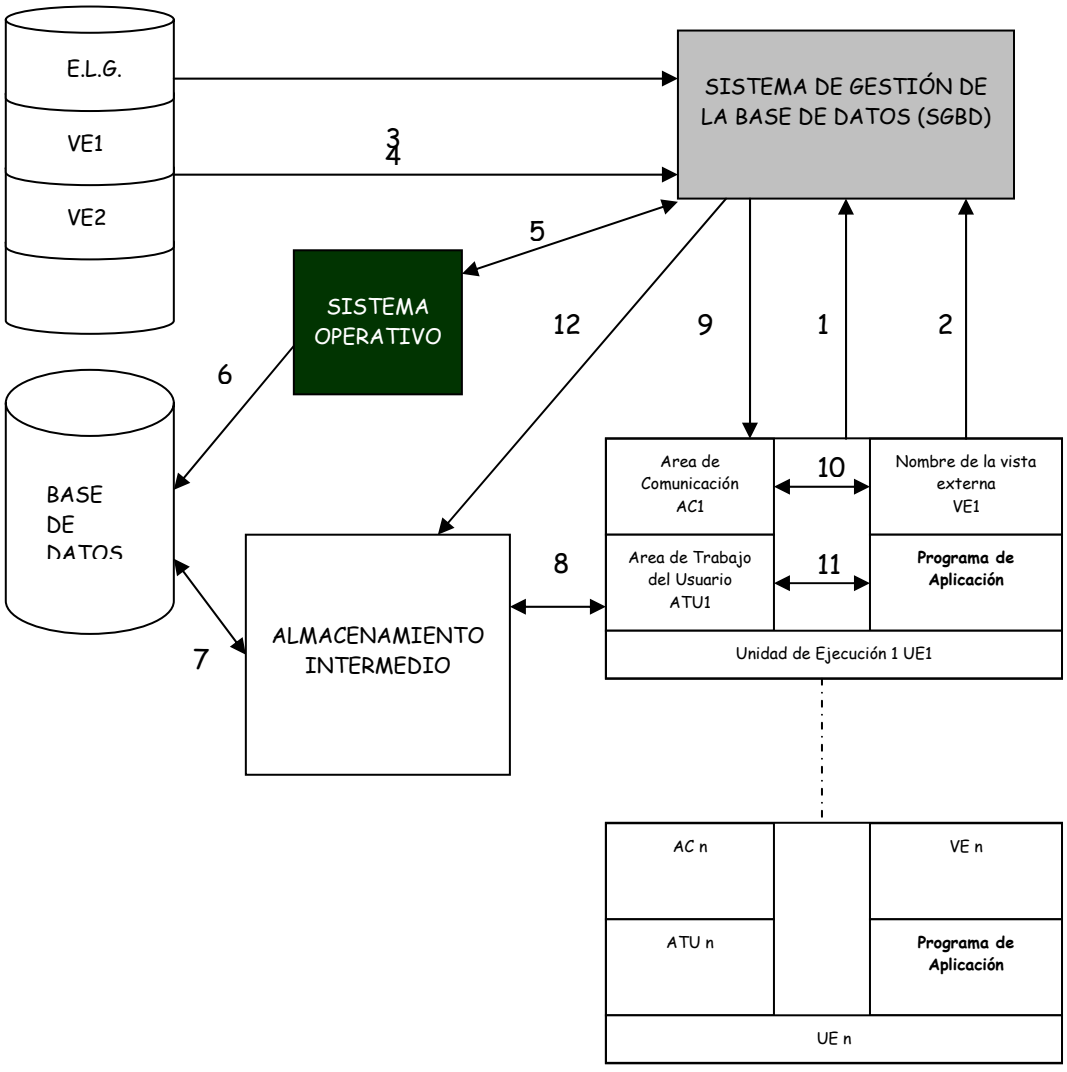
Por último, atendiendo al momento en que actúan sobre la BD, los LMD pueden ser *diferidos* (actuando en procesos por lotes) o *conversacionales* (actuando de modo interactivo con la BD).

5.15.- Integración del SGBD con el Sistema Operativo

El SGBD constituye un subsistema del sistema informático y, en particular, es un subsistema del software. Su funcionamiento, por tanto, estará muy interrelacionado con el de otros componentes del software y especialmente con el Sistema Operativo.

Aunque no es posible un estudio pormenorizado de dicho funcionamiento debido a la diversidad de SGBD y la diversidad del equipo físico en el que se apoya, si puede obtenerse una visión general analizando aquellos aspectos comunes a la mayoría de los SGBD actualmente operativos.

La diferencia entre el modo de acceso a un fichero y a una base de datos se centra en que, en el primer caso, el programa de aplicación accede al fichero por medio del subsistema de gestión de datos del Sistema Operativo, que es quien contiene los métodos de acceso.



Sin embargo, cuando se trata de una base de datos el programa de aplicación (que incluye en su lenguaje anfitrión el LMD embebido) se dirige al SGBD, el cual, a través del Sistema Operativo, accede a la base de datos.

La interacción, en un entorno concurrente, entre el SGBD, el Sistema Operativo y los Programas de Aplicación se muestra en la figura adjunta.

Por cada Programa de Aplicación (PA) que se está ejecutando, existe una Unidad de Ejecución (UE) donde se encuentra el Área de Trabajo del Usuario (ATU) con sus Áreas de Entrada y Salida (E/S) y un Área de Comunicación con el SGBD (AC) destinada a recibir los mensajes y la información de control procedente del SGBD. Desde el programa de aplicación se hace referencia a la Vista Externa (VE) permitida a tal programa. En la biblioteca del sistema se encuentran almacenados, además de los datos, la estructura lógica global y la estructura interna, así como las vistas externas que serán llamadas por los programas de aplicación de los usuarios.

El flujo de datos e instrucciones entre estos elementos es el siguiente:

1º.- Se produce una llamada desde una unidad de ejecución al SGBD (flecha 1); en la llamada se ha de hacer referencia a la vista externa implicada (flecha 2)

2º.- El SGBD analiza la llamada y completa los argumentos con la información de la vista externa a la que se ha hecho referencia en la llamada, así como con la información correspondiente a la estructura lógica global y la estructura interna con ella relacionada; esta información se encuentra previamente almacenada en los ficheros del sistema , desde donde pasa al SGBD (flechas 3 y 4).

3º.- Una vez comprobado el derecho del Programa de Aplicación (PA) a utilizar esta vista , y después de verificar su corrección, el SGBD traduce la llamada en las correspondientes órdenes para los métodos de acceso del Sistema Operativo (flecha 5).

4º.- El Sistema Operativo accede al soporte secundario (disco) donde se encuentran los datos (flecha 6)

5º.- Los datos a recuperar pasan del soporte donde se encuentra almacenada la base de datos al área de almacenamiento intermedio (buffers), y, si se tratase de una inserción o modificación pasarían en sentido contrario (flecha 7).

6º.- Los datos son transferidos desde el área de almacenamiento intermedio al área de trabajo del usuario de la unidad de ejecución desde donde se hizo la llamada (flecha 8) [o en sentido contrario si se hizo una inserción o una modificación], realizándose las correspondientes transformaciones entre las representaciones de los datos.

7º.- El SGBD, una vez terminada la operación de manipulación pasa al área de comunicación los indicadores de estado (flecha 9), en éstos se señala si la operación ha acabado satisfactoriamente o no, al tiempo que se dan otras informaciones sobre la operación realizada.

8º.- El Programa de Aplicación revisa el estado de los indicadores, que se encuentran en el área de control de la unidad de ejecución desde la que se efectuó la llamada, y toma las decisiones oportunas (flecha 10).

9º.- Los datos, que se encuentran en el área de E/S de la correspondiente unidad de ejecución, en el caso de que la operación haya terminado satisfactoriamente, ya pueden ser utilizados por el Programa de Aplicación (flecha 11).

5.16.- El administrador de la base de datos

Es el responsable del diseño, control y administración de la base de datos. Realmente se trata de una función, por lo que puede ser desempeñada por una persona, o mas bien, por un equipo de personas dependiendo de la envergadura del proyecto.

La figura del administrador está reconocida en los diferentes grupos de estandarización (Codasyl, ANSI//SPARC, etc.) definiendo su función en tres niveles o con tres cometidos distintos:

- *Administrador de la Empresa se encarga del diseño conceptual y lógico de la base de datos*
- *Administrador de la Base de Datos es el responsable de las funciones de diseño físico, mantenimiento, seguridad, etc. además de la labor de ajuste o refinamiento a efectos de eficiencia.*
- *Administrador de Aplicaciones tiene a su cargo la creación de vistas o esquemas externos que necesitan los programadores de aplicaciones para escribir los programas de consulta y actualización de la base de datos*

En general, las principales responsabilidades del Administrador de la Base de Datos (ABD):

- *La estructura de la Base de Datos .- El ABD es el responsable de determinar qué información se almacenará en la base de datos después de haber analizado los requisitos de los distintos usuarios.*
- *La descripción conceptual y lógica de la base de datos.- Es el responsable de realizar el diseño conceptual de la BD para, después, adecuar la estructura conceptual a un SGBD específico, y, por tanto, también a un modelo convencional de datos concreto.*
- *La descripción física de la Base de Datos.- Debe encontrar una estructura interna que soporte el sistema lógico y los objetivos de diseño consiguiendo la máxima eficiencia de los recursos de la máquina.*
- *Las especificaciones y vistas – o subesquemas - para los programas.- Del estudio de los requisitos de los usuarios deberá obtener un conjunto de necesidades en cuanto a procesos a realizar sobre los datos; con esta información definirá las vistas externas y las normativas necesarias para los programas de aplicación. La función de programación no es responsabilidad del ABD.*
- *Los estándares.- Fijará las normas por las que se va a regir la Base de Datos en cuanto a su documentación, metodologías de diseño, etc.*
- *La estrategia de transición.- Será responsable de fijar la estrategia de transición del sistema existente, informatizado o no, al nuevo sistema de información soportado en una base de datos.*
- *Los procedimientos de explotación y uso.- Establecerá la normativa necesaria para la utilización de la base de datos: el modo de solicitar el acceso a la misma, su actualización, etc.*
- *Los aspectos relativos a seguridad.- Será responsable de dichos aspectos, incluidos los procedimientos de control y auditorías.*
- *El control y la interacción entre la red y la base de datos.- En el caso de bases de datos a las que se accede a través de redes o de bases de datos distribuidas.*

En definitiva, el administrador de la base de datos interviene en todas las etapas del ciclo de vida de una base de datos representando en todas ellas un papel fundamental