

## SISTEMAS GESTORES DE BASES DE DATOS (II): ADABAS

El pasado mes se inicio bajo el titulo SISTEMAS GESTORES de BASES de DATOS una serie de artículos con los que se pretende explicar la mecánica seguida por los sistemas gestores para el mantenimiento y recuperación de la información de la Base de Datos, entendienddo que este punto de vista, desde la arquitectura de la BD, debe repercutir no solo en un mejor uso de los recursos puestos a disposición del programador en sus correspondientes lenguajes, sino que puede ser aprovechado por aquellos diseñadores inquietos que deseen crear su propio gestor.

### Introducción:

De acuerdo con la clasificación expresada en el pasado artículo, el sistema gestor que se va a tratar este mes, pertenece al grupo de *Gestores Basados en Listas Invertidas*. Esta denominación es poco conocida, pero como se vera a continuación define perfectamente el sistema en el que se fundamenta la Base de datos a la que sirve.

### Estructura de la Base de datos:

La información contenida en la Base de datos se almacena físicamente en 2 grandes ficheros, denominados "DATA" y "ASSO" si bien existe otro fichero, el "WORK", que solo es utilizado por el propio gestor para la realización de su misión de gestión.

### Fichero DATA:

Este fichero contiene realmente los datos de aplicación del usuario. La información contenida en él se encuentra agrupada en archivos lógicos, pudiendo tener hasta **255 archivos**, tal y como se muestra en la figura 1.

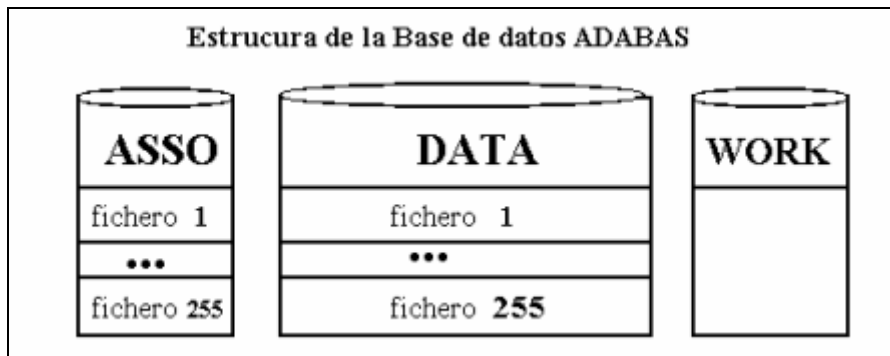


Figura 1

A su vez, cada uno de estos archivos lógicos, tiene la información agrupada en registros, pudiendo alcanzar el número de **16.777.216** registros cada fichero.

Por su parte cada registro puede tener hasta **500** campos, soportando los tipos que se describirán mas adelante.

Con el fin de agilizar las lecturas de disco, y, puesto que el sistema operativo lee y graba la información del disco por **bloques físicos**, el gestor agrupa en bloques los registros del mismo fichero, grabando y recuperando bloques completos de registros. Cada **bloque** se encuentra identificado por un número que determina su dirección relativa a principio

del fichero. Es el *RABN* o Relative Address Bloque Number.

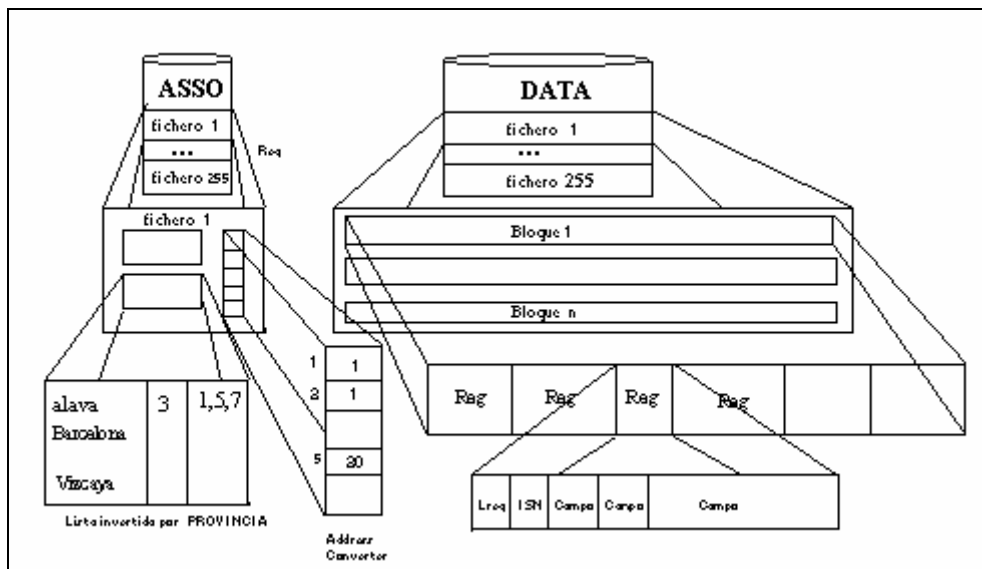


Figura 2

La figura 2 muestra de una forma esquemática la estructura de este fichero. Para poder manejar toda esta información de una forma ágil, en la definición física del registro, se definen como índices determinados campos elementales (aquí denominados *DESCRIPTORES*) o bien agrupaciones de varios campos elementales o subcampos (denominados *SUPERDESCRIPTORES*).

#### Fichero ASSO: ASSOCIATOR

Este fichero está íntimamente ligado al DATA, y su misión es la de permitir la recuperación de la información de una forma rápida, pues es el que contiene la tabla de conversión de direcciones y las listas invertidas que veremos a continuación.

Esta dividido en tantas partes como *archivos lógicos* contenga el DATA, y cada una de estas partes contiene los siguientes elementos:

- 1 único *convertidor de direcciones* o **ADDRESS CONVERTER**
- tantas *listas invertidas* (índices) como descriptores y superdescriptores se hayan definido en el archivo considerado.
- la *Tabla de gestión de almacenamiento* o **SMT** (Storage Management Table).

Los siguientes apartados amplían estos conceptos que se acaban de enunciar.

#### ISN (Internal Sequence Number)

Este es el elemento diferenciador de este modelo de SGDB. Es un **número secuencial** que el gestor asigna a cada uno de los registros en el momento de su alta en la base, y que le sirve para poder identificar al registro de datos.

Para poder asignar este número, el gestor consulta la tabla que contiene todos los ISNs asignados, el *Address Converter* o Convertidor de direcciones. Esta tabla, como ya se ha dicho, contiene en todos aquellos huecos correspondientes a ISNs utilizados, el *RABN* (Relative Address Bloc Number) del DATA en el que se encuentra ubicado el registro. Por tanto, solo tienen que buscar cual es el ultimo cuyo contenido sea distinto de cero, para saber cual es el ISN que debe asignar al nuevo registro. El bloque en el que

ubicara el registro le asigna después de examinar la *SMT* (Storage Management Table - Tabla para la gestión del Almacenamiento) que le indican el espacio libre en cada bloque.

#### **Listas Invertidas:**

Como ya se ha dicho al hablar del DATA, este gestor permite definir tantos índices como se consideren necesarios para poder recuperar la información almacenada en la Base de Datos. Cada uno de estos índices se denomina *DESCRIPTOR* si es campo elemental, o *SUPERDESCRIPTOR* si esta formado por la unión de varios campos.

Pues bien, una *lista invertida* no es mas que la forma que tienen ADABAS de implementar un índice definido sobre el fichero físico. Evidentemente, en base a lo anterior, la lista debe recoger todos los valores que estén contenidos en el campo correspondiente del registro del DATA, y, tiene asociada a cada una de las entradas, la lista de ISNs de los registros que contienen dicho valor. Además dispone de un contador que indica el número de ISNs asociados a dicho valor. Hay que tener en cuenta que los descriptores no tienen por que ser valores únicos; tal es el caso del campo descriptor PROVINCIA, de la figura 2.

Por tanto, se puede considerar que cada **DESCRIPTOR** O **SUPERDESCRIPTOR** definido genera una *LISTA INVERTIDA* con la siguiente estructura:

- Valor clave
- Número de ISNs que contienen el valor especificado,
- Lista de ISNs que tienen ese valor.

#### **Ejemplo:**

Con el fin de facilitar la comprensión de esta organización, se plantea el siguiente ejemplo:

Se desea realizar dos altas en el fichero de clientes sabiendo que cuando se definió físicamente, se estableció que Provincia fuera Descriptor, y que Teléfono también lo fuera.

Nombre	: Fulano	Mengano
Provincia	: Alava	Vizcaya
Teléfono	: 123.45.67	987.65.43

Pues bien, en el momento de dar de alta en la base a estos clientes, se asigna:

- al conjunto "fulano/ALAVA/1234567" el *ISN* 28, y se ubica el conjunto "28/fulano/ALAVA/1234567" en el RABN 12.
- al conjunto "Mengano/VIZCAYA/9876543" el *ISN* 29, y se ubica el conjunto "29/Mengano/28033/9876543" en el RABN 12.

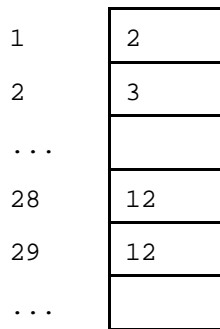
Es el propio gestor quien, en el momento de almacenar el registro en la BD, no solo asigna el ISN, sino que además actualiza todas las listas invertidas del fichero, por lo que si no existe, en la lista invertida de Teléfono el valor contenido en el registro a almacenar, cosa normal pues en principio cada cliente tendrá su teléfono, genera una entrada en dicha lista invertida con los datos correspondientes, es decir

1234567	1	28
9876543	1	29

En cambio, a la hora de actualizar la lista invertida de Provincia, se encuentra con que ya existen clientes de Alava, por lo que se limita a incrementar el contador de ISNs de clientes de ALAVA, y a incluir el ISN 28 en la lista asociada a esta entrada; y hace lo mismo con la entrada correspondiente a VIZCAYA, con lo que resulta.

Alava	4	1,5,7,28
Vizcaya	3	2,6,29

Por su parte, en el *Address Converter*, el SGDB ha incluido en la casilla **28** el valor **12**, para indicar que el registro de datos se encuentra en el bloque 12, y en la casilla **29** también ha incluido el valor **12**, pues el segundo registro también se encuentra en dicho bloque físico.



La figura 3 muestra esquemáticamente toda esta explicación.

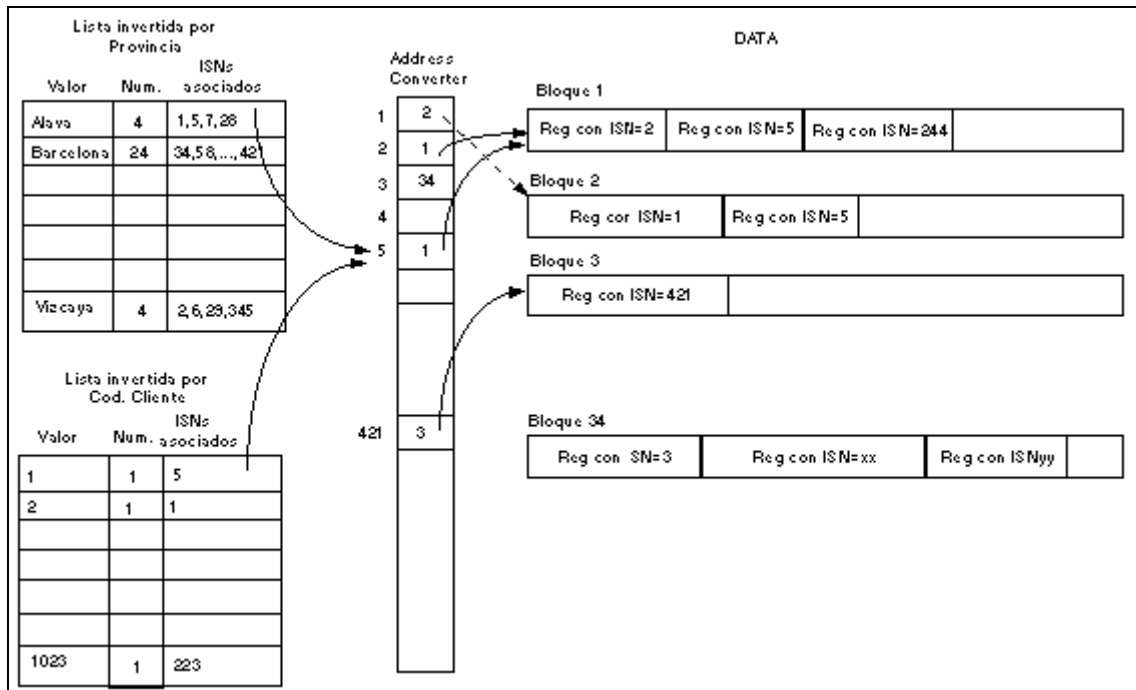


Figura 3

## Métodos de recuperación de información:

ADABAS permite recuperar la información mediante los siguientes métodos:

*READ PHYSICAL* de un fichero, en este caso ADABAS realiza la lectura secuencial del fichero de DATA, por lo cual lee el primer bloque, y devuelve el primer registro lógico del bloque físico, luego el segundo, y así hasta terminar el bloque, momento en el que lee el siguiente bloque, etc. ..., por lo que la información devuelta no sigue ningún orden.

*READ by ISN*: En este caso, ADABAS lee secuencialmente el convertidor de direcciones, es decir los ISNs del fichero, recuperando cada vez el bloque correspondiente a un ISN. Una vez cargado en memoria dicho bloque, localiza dentro del mismo el registro que tenga por ISN el ISN buscado. Para poder realizar esto, en el momento de darle de alta, coloca al principio del registro la **longitud** del mismo, de modo que si al examinar el ISN del primer registro del bloque, resulta que no es el buscado, salta los bytes correspondientes a dicho registro, y examina el ISN del segundo registro, y así sucesivamente hasta localizarlo, cosa que sin lugar a dudas consigue pues se encuentra en dicho bloque.

*GET ISN*: en este caso solo se pretende recuperar la información de un único registro del que se conoce su ISN. Para lo cual lo que hace el gestor es localizar mediante el Address Converter, el bloque en el que se encuentra el registro, leer dicho bloque y devolver al programa llamante los datos que ha solicitado de dicho registro.

*READ by DESCRIPTOR* provincia: En este caso, lo que hace ADABAS es recuperar la información de forma secuencial en base a la lista invertida que se especifique como descriptor para la recuperación. Es decir, va leyendo la lista invertida de PROVINCIA desde el principio o desde el valor que se desee. Una vez posicionado en la lista, va recuperando los registros asociados a cada valor de la lista en base a los ISNs que la acompaña, Así, para ALAVA, recupera primero el ISN 1, para lo cual se basa en el convertidor de direcciones que le indicara el bloque en el que se encuentra el registro asociado a dicho ISN. La siguiente vez que sea invocado ADABAS, devolverá el ISN 5, luego el 7, el 28, luego pasara a devolver los asociados a Barcelona, etc, hasta llegar al fin de la lista.

Los siguientes métodos no siguen una recuperación secuencial, sino que permiten extraer información con solo examinar las listas invertidas, buscando valores concretos:

*FIND with PROVINCIA=valor* : En este caso, ADABAS examina y extrae de la lista invertida el conjunto de ISNs que cumplen la condición especificada. Y dicho conjunto (o **SET de ISNs**) le guarda en el WORK (el tercer fichero de la estructura que usa para su propia gestión). Una vez hecho esto, actúa normalmente, es decir, recupera el registro asociado al primer ISN del SET, Cuando se le pida el siguiente, ya no le buscara en las listas, sino que hará uso de este SET, por lo que devolverá el registro asociado al siguiente ISN, y así sucesivamente hasta que se acaben los ISNs del SET.

*FIND with PROVINCIA=valor1 AND PROFESION=valor2*: Este caso es parecido al anterior pues solo se ha añadido una condición a la sentencia de búsqueda. Con él se pretende mostrar la potencia del método, ya que permite determinar los registros que cumplen las dos condiciones, con solo examinar los ISNs asociados a los SETs de cada condición. Es decir, cuando ADABAS recibe esta petición, busca los ISNs que cumplen la primera condición, y

les memoriza en su WORK, luego hace los mismo con la segunda condición, y por ultimo realiza la intersección entre ambos conjuntos, determinado los ISNs que cumplen las dos condiciones. Este es el set de ISNs buscados, y por lo tanto, una vez determinado solo tiene que ir recuperando los registros asociados a éstos para ir enviandoselos al programa llamante.

El número de condiciones que se pueden poner en una sentencia FIND es ilimitado, pero el programador debe ser consciente de la penalización que esto supone en el programa para la recuperación del primer registro.

*FIND with PROVINCIA=valor1 AND PROFESION=valor2 sorted by NOMBRE:*

Este es un caso mas que muestra la potencia del método. Una vez obtenido el set de ISNs que cumplen las dos condiciones, los registros deben ser devueltos en el orden que determine el descriptor usado como clasificador, en este caso **NOMBRE**. Esto es posible ya que solo debe buscar los ISNs del SET resultante de las condiciones del FIND, entre los ISNs de la lista NOMBRE. Esta sentencia solo puede usarse con descriptores.

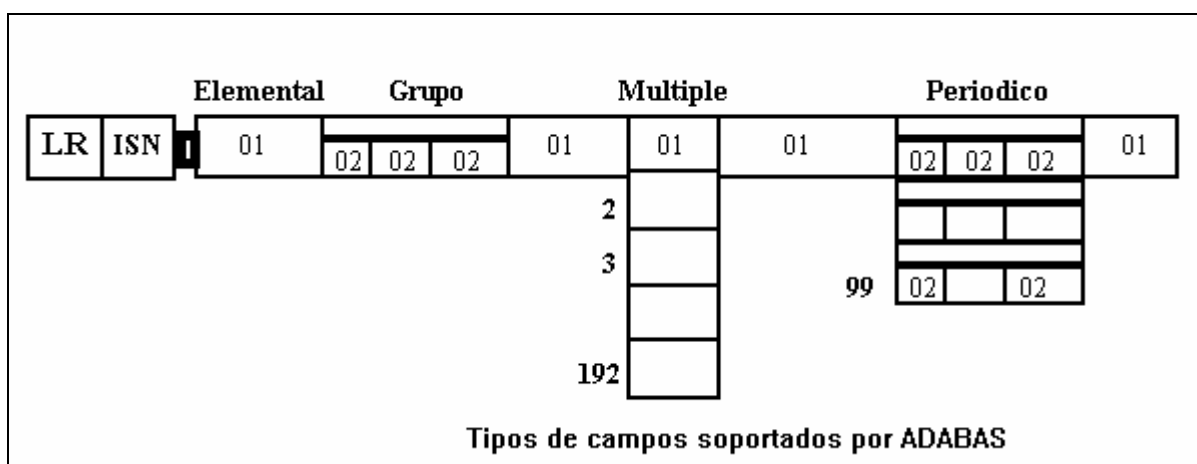
Además de estos métodos para la recuperación de los datos contenidos en los registros de la BD, ADABAS nos permite tratar las listas invertidas como si se trataran ficheros de datos mediante las sentencias siguientes:

*HISTOGRAM provincia:* En este caso, se lee la lista invertida de PROVINCIA recuperando la información en ella contenida. Es de resaltar que en este caso, solo se puede consultar los campos Valor-índice y \*number que contiene el número de ISNs asociados a cada uno de los valores, es decir, ALAVA,3 BARCELONA,27 etc...

*FIND NUMBER PROVINCIA=valor:* Esta sentencia también nos recupera información de la lista invertida, devolviendo el valor del \*number asociado al valor que se pasa como argumento. Así, si se hubiera dado ALAVA, devolvería 3.

**Tipos de campos soportados por ADABAS:**

Por ultimo es necesario describir los tipos de campos soportados por este gestor.



Campo *ELEMENTAL*: como su nombre indica es la unidad mínima de información.  
 Campo tipo *GRUPO*: Aquel que se descompone en varios campos elementales.  
 Ejemplo: El campo NOMBRE\_COMPLETO, se compone de los campos elementales NOMBRE, APELLIDO\_1, APELLIDO\_2.  
 Campo *MULTIPLE*: es un campo elemental que puede tener varias ocurrencias:

Ejemplo: Supuesto que en el registro de empleados se desee mantener la información de los proyectos en los que ha participado cada empleado, este se definiría como múltiple, pudiendo tener hasta **192** ocurrencias cada campo múltiple.

Campo *PERIODICO*: Es parecido al múltiple, con la salvedad de que en lugar de ser un campo elemental, cada ocurrencia es un campo tipo grupo. Ejemplo, si queremos conocer en el caso anterior, no solo los proyectos en los que ha participado, sino las fechas de intervención, en lugar de definir un campo múltiple se definiría un campo *PERIODICO*, donde cada ocurrencia tendría como campos elementales *NOMBRE\_PROYECTO*, *FECHA\_INICIO*, *FECHA\_FIN*.

### **La Integridad referencial**

*SOFTWARE A.G.*, la creadora de ADABAS, garantiza la integridad referencial de ADABAS mediante el uso de dos sentencias básicas para la realización del checkpoint: *END TRANSACTION* y *BACKOUT TRANSACTION*. La primera pasa a definitivas todas las modificaciones, altas y bajas realizadas por el usuario desde el ultimo checkpoint *sobre todos los ficheros actualizados*, y, mediante la segunda anula todas las modificaciones realizadas también sobre todos los ficheros desde el ultimo checkpoint realizado. Por tanto, debe ser el propio programador quien, mediante el correcto uso de estas instrucciones, pase a definitivas las actualizaciones de la B.D. generadas en la transacción.

### **DDM: Modulo de definición de datos**

Otra de las cuestiones planteadas en el pasado artículo al hablar de las ventajas de un sistema gestor, era el hecho de que al interponerse éste entre el programador y los datos, podía restringir el acceso a los mismos. Uno de los métodos usados para conseguir este objetivo es limitar la visión que el programador tenga del registro del fichero, y en consecuencia de los datos contenidos en el mismo. Esto se consigue definiendo varias "vistas" de un mismo registro. Así, por ejemplo, un vista del registro del fichero *EMPLEADO* contendría los datos básicos del mismo, tales como nombre, dirección, teléfono. Otra vista, también asociada al mismo fichero, contendría los datos económicos, tales como nombre, sueldo, primas, etc,

La DDM es precisamente la "vista" o modulo que usara el programador para acceder a los datos de un fichero. En el se encuentran solo las definiciones de todos los campos accesibles del mismo junto con los atributos de tipo, nombre, formato y longitud, así como una indicación de los descriptores y superdescriptores.

### **ADABAS y las formas Normales:**

Con solo la enumeración de los tipos de campos soportados por ADABAS, en concreto por los campos múltiples y periódicos, puede parecer que esta base no se adapta a la *primera forma normal*. Pero en este punto es necesario resaltar que no es la Base de datos quien sigue las formas normales, sino el MODELO DE DATOS diseñado para soportar los datos de una aplicación concreta quien debe normalizarse..

La misma observación cabe hacer sobre las 2ª y 3ª *forma normales*, pues como se ha descrito, los elementos no clave dependen no de una única clave, sino de todos los campos definidos como descriptores y superdescriptores, pues utilizando cualquiera de ellos se recupera la totalidad del registro (a través del ISN de la lista invertida).

Por ultimo, cabe resaltar el hecho de que ADABAS permite implementar cualquier modelo de datos diseñado para bases de datos jerárquicas, en red, o relacionales. Esto solo es posible hacerlo con ADABAS por su arquitectura y por los múltiples métodos de recuperación de la información de que dispone.

**Conclusión:**

El empuje protagonizado por ADABAS en las Grandes Instalaciones de España, avalan el posible exceso de entusiasmo mostrado por el autor de este artículo, pero es que combinando este gestor con *NATURAL*, lenguaje creado también por *SOFTWARE AG* para sacar el máximo partido a su BD, resulta una herramienta potente, sencilla de manejar, y capaz de permitir desarrollos entre 4 y 5 veces mas rápidos que con cualquier otro lenguaje.