

Introducción a las Bases de Datos

Por Nacho Cabanes, ncabanes@pobox.com,

Versión 0.03, Julio 2001

Introducción.....	3
Conceptos básicos.....	3
¿Por qué este texto?	4
Condiciones de uso.	5
Nociones básicas de diseño.....	5
Etapas básicas del diseño.	5
Un primer ejemplo.	6
Introducción al modelo Entidad-Relación.....	8
Diagrama EER de nuestro ejemplo.....	9
Convirtiendo el diseño a tablas.....	11
Método general.....	11
Convirtiendo a tablas nuestro ejemplo.	12
Tipos de datos existentes.....	14
Problemas que hemos evitado con nuestro diseño.	15
Problemas que aún tiene nuestro diseño.	16
Creando la base de datos con StarBase (StarOffice).	17
Creando nuestra BD con Access.....	24
Creando nuestra BD con WinSQL.....	31
Creando nuestra BD con dBase III+.....	33
Un segundo ejemplo paso a paso.	33
Enunciado del segundo ejemplo.....	33
Diseño del segundo ejemplo.....	33
Implementación del segundo ejemplo.	34
Visualizando datos: consultas básicas y relaciones.	34
Listados e informes.....	34
Otras notaciones en Entidad-Relación.....	34
Versiones de este texto.....	34
Índice Alfabético.....	35

Introducción.

Conceptos básicos.

En el sentido más amplio, se podría considerar que una base de datos es simplemente un conjunto de información.

En un ordenador, esta información normalmente será básicamente una serie de “fichas”.

Por ejemplo, una base de datos muy sencilla podría ser simplemente una agenda de direcciones en la que anotemos datos de nuestros amigos. Tendríamos una ficha para cada uno de estos amigos. En cada ficha, a su vez, existirá una serie de apartados, como el nombre, la dirección, el teléfono, etc.

Esto nos permite introducir dos primeros conceptos que utilizaremos: cada una de estas “fichas” recibe el nombre de “**registro**”, y cada uno de los “apartados” que tienen las fichas se llama “**campo**”.

El conjunto de las fichas (registros) forma un “bloque” de información, que llamaremos “**tabla**”, y que se suele representar escribiendo cada ficha (registro) en una fila y cada apartado (campo) en una columna, así:

<i>Nombre</i>	<i>Direcc</i>	<i>Ciudad</i>	<i>Tlf</i>
José	C/ Rana, 1	Alicante	111-11-11
Jesús	C/ Sapo, 2	Madrid	222-22-22
Juan	C/ Boa, 3	Barcelona	333-33-33

Pero en la práctica, una “**base de datos**” real suele estar formada por más de una tabla. Por ejemplo, la base de datos que utiliza una empresa “normal” para su gestión deberá almacenar datos sobre clientes, proveedores, artículos, facturas, etc. Cada uno de estos “bloques” de datos será una tabla, y estas tablas estarán relacionadas entre sí (por ejemplo: un artículo será suministrado por un cierto proveedor, y ese artículo aparecerá en ciertas facturas, cada una de las cuales corresponderá a un cierto cliente).

Todo este conjunto de información que forman las tablas y las relaciones entre ellas (y alguna cosa más que ya veremos) será nuestra “base de datos”. En concreto, se tratará de lo que se conoce como una “**base de datos relacional**”.

En este texto veremos las nociones básicas de cómo se crea y se maneja una base de datos: cómo diseñar las tablas y las relaciones, cómo introducir los datos y como buscar información en la base de datos. Intentaré aplicarlo a alguna de las bases de datos más habituales, como Access para Windows (para quien tenga un ordenador “moderno”) o dBase III+ (para los que tengan un ordenador “antiguo”). También daré unas nociones básicas de lenguaje SQL, un lenguaje de consulta a bases de datos que la mayoría de los sistemas actuales permiten entregar.

¿Por qué este texto?

Este texto **no es para informáticos**. Me explico: voy a profundizar mucho menos de lo que se podría exigir a un informático “de carrera”, incluso cuando apenas acaba de comenzar sus estudios. Si a algún estudiante de informática le resulta de utilidad, me sentiré muy halagado, pero que nadie espere aprobar asignaturas de universidad sólo con este texto. No trataré apenas conceptos básicos, ni los requisitos que debe cumplir un SGBD (sistema de gestión de bases de datos) real mínimamente grande, ni la normalización, ni otros modelos distintos del Entidad-Relación, ni siquiera veremos buena parte de las posibilidades de este modelo.

Este es un texto **para aficionados** que quieran hacer sus pequeños proyectos. Pero insisto en lo de “pequeños proyectos”: un proyecto de una cierta envergadura debería dejarse en manos de un informático “de carrera”, porque un simple error de diseño nos puede hacer perder mucho tiempo, y si la base de datos la hemos creado para una empresa (por pequeña que sea), ese tiempo perdido es dinero perdido.

Pero es que, más de una vez, he tenido yo mismo que parchear bases de datos creadas por aficionados (y lo de “aficionados” no lo digo en tono peyorativo sino con respeto y admiración), y he perdido mucho tiempo en intentar aprovechar los datos existentes y adaptarlos para que fueran eficientes. Si se hubieran diseñado apenas UN POCO mejor, habrían sido MUCHO más versátiles, más fáciles de corregir y ampliar, ocupado menos espacio, etc.

Por eso, comentaré alguno de los errores más habituales, de modo que quien cree sus bases de datos, aunque sea como hobby, cree algo que realmente resulte útil y manejable.

Condiciones de uso.

Este texto se puede imprimir y se puede distribuir libremente, tanto en su forma electrónica como impresa, SIEMPRE Y CUANDO NO SE MODIFIQUE y se conserve el nombre del autor.

Si este texto se desea utilizar como base para cualquier otro texto, artículo o similar, también se puede hacer libremente SIEMPRE Y CUANDO SE MENCIONE AL AUTOR.

El autor no podrá ser en ningún caso considerado responsable de ningún problema debido directa o indirectamente a la información contenida en este texto. El usuario acepta todas las responsabilidades. *Es decir, que si por experimentar algo de lo que aparece en el texto, tu disco duro salta en pedazos (cosa que veo muy difícil), sería responsabilidad exclusivamente tuya.*

EL USO DE ESTE TEXTO SUPONE LA ACEPTACIÓN TOTAL DE ESTAS CONDICIONES.

Si descubres cualquier error en el texto, rogaría que me lo comunicases por correo electrónico a la dirección:

ncabanes@pobox.com

Este texto es “**GraciasWare**”. Esto quiere decir que si te gusta y/o te resulta útil, basta con que me mandes un correo electrónico a mi dirección ncabanes@pobox.com. Si veo que tengo apoyo moral, procuraré ir mejorando este texto y lanzando nuevas versiones, cada vez más ampliadas.

Nociones básicas de diseño.

Etapas básicas del diseño.

El **primer paso** antes de crear una base de datos es pararse a pensar. Ni más ni menos. Si en la programación es muy peligroso eso de empezar a teclear según aparece una idea (a pesar de que hay bastante gente que lo hace, y alguno lo hace incluso bien), en la creación de bases de datos es muy raro que salga bien.

El **segundo paso** recomendable es ir anotando las ideas según surgen. Cuando creemos que ya está todo, deberíamos volver a leer todas las notas que habíamos tomado, porque eso nos ayudará a tener una visión de conjunto y a notar si falta algo que no hayamos previsto inicialmente.

El **tercer paso** será empezar a dibujar garabatos que representen esa información. Para ello veremos por encima un modelo llamado “Entidad-Relación”. El dibujo nos ayudará a tener una nueva versión de conjunto, mucho más fácil de seguir y más completa que las anotaciones. Aquí se verán todavía mejor las carencias y las incongruencias que puedan existir.

El **cuarto paso** será convertir este dibujo en las tablas. Este paso puede ser casi totalmente mecánico. Por ejemplo, la conversión del modelo Entidad-Relación (el que veremos) a una base de datos relacional (las que normalmente manejaremos) es casi inmediato.

Se acabó el diseño. Podríamos añadir un **quinto paso** que sería la introducción de los datos y la creación de una serie de estructuras auxiliares, como formularios, consultas o informes, que ya veremos.

Un primer ejemplo.

Vamos ver un primer ejemplo, que nos ayudará a llevar a la práctica todo esto y a introducir el modelo “Entidad-Relación”.

Supondremos que nos proponen el siguiente problema:

“Se desea informatizar un centro de estudios de pequeño tamaño. Interesa controlar exclusivamente los asuntos académicos: qué alumnos tenemos, qué cursos/asignaturas han realizado, qué profesores tenemos en plantilla, quién ha impartido cada uno de los cursos, etc”.

Esto serían las indicaciones que nos daría el cliente (o que nosotros pensaríamos, si lo realizamos para nosotros mismos).

Ahora tendríamos que pensar si vemos que falta algo (y preguntar al cliente, si procede, cosas como si desea guardar la dirección y demás datos postales de los alumnos y de los profesores, o si quiere saber la nota que cada alumno obtuvo en cada curso) o incluso si sobra algo (porque resulte demasiado difícil de llevar a cabo -¿difícil, para nosotros?... lo dudo-).

Pasamos a desglosar en bloques de información. De momento todavía no hablaremos de tablas, sino de “**entidades**” (un nombre más ambiguo pero más adecuado) y de “**relaciones**” entre estas entidades.

En nuestro caso, las “cosas” (entidades) que tenemos son básicamente éstas:

- Alumnos.

- Cursos.
- Profesores.

Y las relaciones que hay entre estas entidades son:

- Los profesores IMPARTEN cursos.
- Los alumnos ASISTEN a cursos.
- (Indirectamente, los alumnos y los profesores también están relacionados: un alumno ha asistido a un curso que ha impartido un cierto profesor; esta relación ya queda reflejada a partir de las otras dos, así que no es necesario detallarla).

Aun comentaremos algo más sobre las relaciones. Una característica importante de las relaciones es su **“cardinalidad”**: por ejemplo, en la relación de que “los alumnos asisten a los cursos”, es importante si a cada curso sólo puede asistir un alumno o varios, y si un alumno puede asistir a un solo curso o a varios.

Tendremos cuatro posibilidades:

- Que cada alumno asista uno y solo uno de los cursos (se expresa como 1:1 -uno a uno-)
- Que cada alumno pueda asistir a muchos cursos, pero en cada curso sólo puede haber un alumno (1:M -uno a muchos-)
- Que cada alumno pueda asistir a un único curso, pero pueda haber en un curso varios alumnos en un curso (M:1 -muchos a uno-).
- Que cada alumno pueda asistir a muchos cursos, y en cada curso pueda haber varios alumnos (M:M -muchos a muchos-)

En nuestro caso, la relación “asistir” es una relación de “muchos a muchos” (M:M).

Podríamos preguntarnos la cardinalidad de la otra relación (“los profesores imparten cursos”). En este caso, cada profesor puede impartir varios cursos, y supondremos que cada curso es impartido por un único profesor (estoy dando por supuesto que se considera distinto un curso de “Bases de Datos” impartido en una fecha y otro de la misma temática pero impartido en fecha distinta). Se trataría de una relación “de uno a muchos” 1:M.

Una observación: en las relaciones es importante el sentido en el que se leen. Por ejemplo, la relación “los profesores imparten cursos” es una relación 1:M (uno a muchos), mientras que la

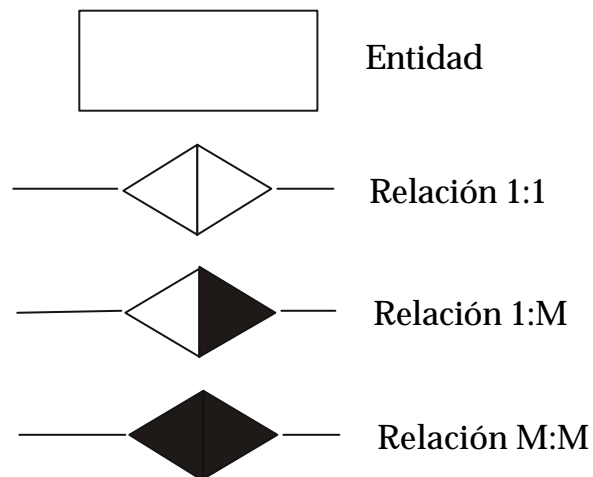
relación “los cursos son impartidos por profesores” es una relación M:1 (muchos a uno).

Estas relaciones que hemos comentado son “relaciones binarias” (entre dos entidades). Por el carácter introductorio de este texto, no entraremos en las relaciones que engloban más entidades (como las ternarias), ni en cierto tipo de restricciones (como las de existencia o valor no nulo), ni en generalizaciones, asociaciones ni agregaciones.

Introducción al modelo Entidad-Relación.

Este es un modelo que nos permitirá “dibujar” las entidades y las relaciones que existen entre ellas. Nosotros usaremos el modelo “Entidad-Relación Extendido” (**EER**, de aquí en adelante). Existen varias notaciones ligeramente distintas. Voy a utilizar la que considero más sencilla.

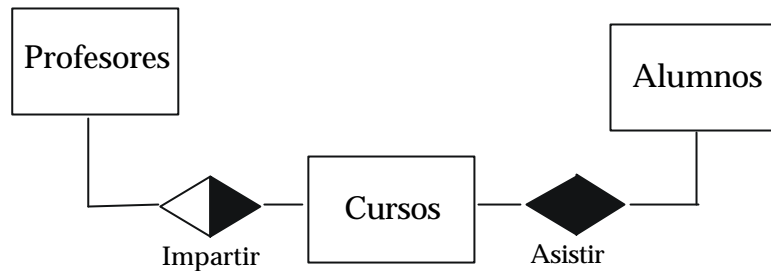
En esta notación se representan las entidades como un rectángulo y las relaciones binarias como un rombo partido por la mitad. Si la relación es 1:M, una de las mitades (la que corresponde al “muchos”) deberá estar sombreada, y si es M:M, todo el rombo estará sombreado.



(Nota: he visto las relaciones expresadas también de otras formas, por otros autores; de momento, emplearé esta notación y más adelante comentaré otras notaciones que es posible utilizar o encontrar en otros textos).

Diagrama EER de nuestro ejemplo.

Vamos a ver cómo quedaría el diagrama Entidad-Relación de nuestro ejemplo:



Así de sencillo: tenemos 3 entidades (profesores, cursos, alumnos) y dos relaciones (impartir, entre profesores y cursos, 1:M, y asistir, entre alumnos y cursos, M:M).

Realmente, ya a este nivel se suele indicar los “apartados” que hay en cada entidad (lo que serán los “campos” de nuestras tablas), que ahora llamaremos “**atributos**”. Estos atributos se representan como pequeñas elipses que salen de las entidades.

Vamos a pensar primero qué atributos nos podría interesar para nuestras entidades:

Alumnos:

- DNI (Documento Nacional de Identidad)
- Nombre
- Dirección
- Ciudad
- Teléfono
- Fecha de nacimiento
- Fecha de alta en el centro
- Fotografía

Profesores:

- DNI.
- Nombre
- Dirección
- Ciudad
- Teléfono

- Conocimientos
- Sueldo
- Cuenta bancaria

Cursos:

- Nombre del curso
- Fecha de comienzo
- Duración (horas)
- Importe (pesetas)
- Número máximo de alumnos

Es sólo un ejemplo. Insisto en que de momento no estamos pensando en tablas, sino simplemente en qué información queremos almacenar. Según el sistema de bases de datos que empleemos realmente, puede ocurrir que sea incómodo (o incluso imposible) trabajar con algunos de estos datos que hemos previsto (por ejemplo, la “fotografía” del alumno). Pero eso ya nos lo plantearemos después.

Lo que sí vamos a pensar ya es cual de esos datos nos permitirá distinguir una ficha de otra. Esto se hace porque podemos tener dos alumnos con el mismo nombre, pero claramente son personas distintas, y debemos saber qué cursos ha realizado cada uno de ellos sin posibilidad de confusión, para no dar a uno el diploma que corresponda a otro, ni cobrarle un dinero de otro.

En el caso de los alumnos, no son datos únicos: el nombre (puede repetirse, incluso con apellidos), la dirección (dos hermanos o dos amigos pueden vivir en la misma casa), el teléfono (ocurre lo mismo), la fecha de nacimiento (también podemos encontrar dos alumnos que hayan nacido el mismo día), etc. Lo que realmente distinguirá a un alumno de otro es su número de DNI (Documento Nacional de Identidad) o pasaporte, que sí es único.

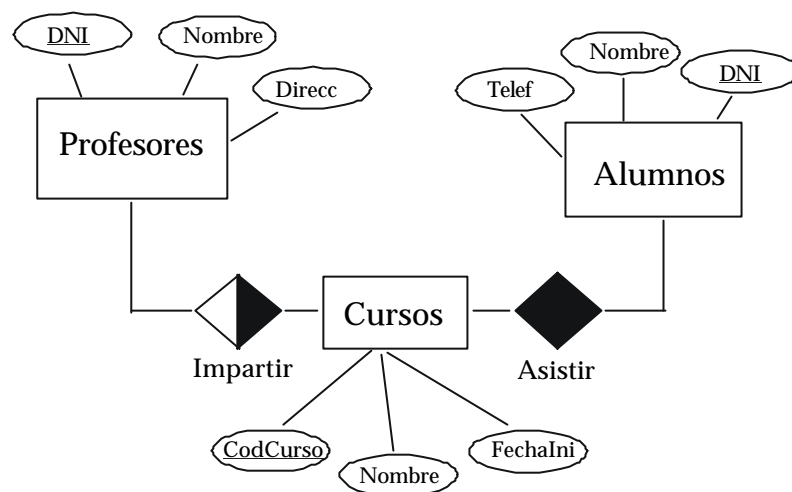
Pues bien, este dato que puede distinguir una persona de otra (o en general una ficha -registro- de otra) es lo que llamaremos la “**clave**”.

Puede ocurrir que no exista nada que nos sirva claramente como clave, como es el caso de los cursos: no es único el nombre (podemos impartir más de un curso con el mismo contenido), ni la fecha de comienzo (varios cursos pueden comenzar el mismo día), ni la duración, ni el importe, ni el número máximo de alumnos. En estos casos se suele añadir algo arbitrario, un código, que nos permita distinguir un curso de otro (en general una ficha -registro-

de otra). En nuestro caso, incluiríamos un nuevo atributo, llamado “Código de curso”.

Un último comentario antes de ver cómo quedaría nuestro diagrama EER con sus atributos. Puede ocurrir que nuestra entidad tenga varios atributos únicos, todos los cuales puedan servir como clave. Entonces escogemos una de ellas como “**clave principal**”, y el resto serán “**claves alternativas**”, que no llegaremos a usar como claves. En el diagrama, el atributo que vaya a utilizarse como clave principal aparecerá subrayado.

Ahora ya sí. Nuestro diagrama quedaría así (no incluyo todos los atributos que habíamos pensado, sólo algunos como ejemplo, que es con los que trabajaremos a partir de ahora):



Convirtiendo el diseño a tablas.

Método general.

Posiblemente la mayor ventaja del modelo EER es que la conversión a tablas es casi inmediata:

- En principio, cada entidad corresponderá a una tabla, y los atributos de la entidad darán lugar a los campos de la tabla.
- Las relaciones “de uno a muchos.” (1:M) se reflejarán poniendo la clave del “uno” en la tabla de los “muchos”. Me explico: si un profesor imparte muchos cursos (y cada

curso es impartido por un único profesor), lo que haremos es poner el código del profesor (su DNI) en la ficha de cada curso que imparte.

- Las relaciones “de muchos a muchos” (M:M) aparecerán como una nueva tabla, en la que cada registro estará formado por las claves de las tablas que se relacionan. En nuestro caso, la relación “de muchos a muchos” entre alumnos y cursos se convertirá en una nueva tabla, en la que cada registro contiene dos campos: el código del curso al que se asiste y el DNI del alumno que ha asistido.
- Las relaciones “de uno a uno” muchas veces son debidas a un fallo de diseño, y corresponden a datos que deberían estar en una misma tabla. Por ejemplo, si suponemos que cada persona tendrá una única dirección postal, y cada dirección corresponde a una única persona, entonces el dato de la dirección postal deberá ser uno más de la tabla que almacena todos los datos de esa persona. Esta simplificación que hemos hecho no es del todo cierta: a veces las relaciones 1:1 deberán reflejarse como una nueva tabla, dependiendo de si existe restricción de existencia o no, pero no entraremos en tanto detalle.

Convirtiendo a tablas nuestro ejemplo.

Así, en nuestro caso, obtendríamos las siguientes tablas (con sus campos, limitándonos a los atributos que hemos incluido en el último diagrama):

Alumnos:

- DNI (clave)
- Nombre
- Tlf

Profesores:

- DNI (clave)
- Nombre
- Direcc

Cursos:

- Código (clave)
- Nombre del curso

- Fecha de comienzo
- DNI del profesor

Asistir:

- Código del curso
- DNI del alumno

Ya sólo falta una cosa. Hay que decidir los **tipos de datos** de los campos y también los tamaños de los campos. Esto es porque al ordenador habrá que darle toda la información muy cuadrículada, para que podamos guardar la información que nos interesa y a la vez que no desperdiciemos demasiado espacio.

Los tipos de datos existentes pueden variar de un sistema de bases de datos a otro, así que vamos a limitarnos (por ahora) a hacer una primera aproximación, acercándonos al caso de nuestros campos:

- El nombre de un alumno, de un profesor o de un curso, estará formado básicamente por letras. Todos los sistemas de bases de datos tendrán un tipo de datos adecuado para almacenar series de letras (que podrán incluir alguna cifra numérica o algún otro símbolo). Será un tipo llamado “Texto”, “Alfanumérico”, “Carácter” o algo similar. En cuanto al tamaño, nos puede bastar con unas 40 letras (no debemos quedarnos cortos, pero si nos excedemos, estaremos desperdiciando espacio).
- La dirección tendrá también letras, números y algún otro símbolo, de modo que también será tipo Texto, y unas 50 letras de tamaño puede estar bien.
- El DNI del alumno o del profesor contendrá cifras numéricas, pero posiblemente también alguna letra, de modo que nos interesará que también este dato sea de tipo “Texto”, y entre 10 y 15 letras de tamaño (dependiendo de si vamos a escribir puntos en los millares, guión antes de la letra, etc.).
- El teléfono del alumno sólo contendrá cifras. Tendremos un tipo de dato “Numérico”, que nos puede servir en este caso y que será imprescindible en el caso de que queramos hacer operaciones aritméticas con los datos almacenados en un campo. En el caso del teléfono, no necesitamos hacer operaciones, y también es posible que nos interese escribir paréntesis, guiones o espacio, de modo que quizá sea más interesante dejarlo como tipo “Texto”, de unas 12-15 letras.

- Para la fecha de inicio de un curso, casi todos los sistemas de bases de datos nos permitirán utilizar un tipo de datos llamado “Fecha”.
- El código de un curso queda a nuestra elección: si queremos que esté formado sólo por números, sería tipo de datos “Numérico”; si queremos que pueda contener letras u otros símbolos, debería ser de tipo “Texto”. Algunos sistemas de bases de datos van más allá y permiten un tipo “Autonumérico”, que es un dato numérico que va incrementándose automáticamente (en el primer registro que introduzcamos será un 1, en el segundo un 2, y así sucesivamente), para que no tengamos ni siquiera que pensar qué código queremos para cada registro (hay gente a quien esto lo parece muy cómodo y otros que lo consideran demasiado rígido).

Tipos de datos existentes.

En general, los tipos de datos habituales, que encontraremos en casi cualquier sistema de bases de datos, son los siguientes:

- **Texto** (o alfanumérico, o carácter), cuando nuestro campo deba almacenar letras y quizás algún otro tipo de símbolos de puntuación y/o cifras numéricas. Deberemos indicar la cantidad de letras (o en general, de caracteres) para las que queremos dejar espacio (no deberíamos quedarnos cortos, para que nos quepa toda la información que nos interesa, pero tampoco hay que dejar mucho espacio de más, o estaríamos desperdiciando parte de la capacidad de nuestros sistemas de almacenamiento sin necesidad).
- **Numérico**, cuando nuestro campo vaya a guardar cantidades numéricas, especialmente si más adelante necesitaremos realizar operaciones aritméticas con estas cantidades numéricas. Tendremos que indicar también el espacio que queremos reservar, pero esto puede que se haga de forma distinta según el sistema de bases de datos que usemos. Por ejemplo, unos esperarán que les digamos el número de cifras que queremos guardar, mientras que otros emplearán nombres más cercanos a como realmente se va a guardar la información en el ordenador (cosas como “número entero largo” o “número real de doble precisión”).
- **Lógico**, cuando sólo hay dos posibilidades (verdadero o falso, sí o no).

- **Fecha**, para almacenar fechas (y, en ocasiones, también horas). Se utiliza para que las comparaciones y las ordenaciones sean correctas (por ejemplo, si escribimos las fechas 12/01/2000 y 31/10/1975 como “texto”, el ordenador consideraría que la primera es menor -anterior- a la segunda, lo cual es claramente incorrecto).
- **Memo**, es un campo de texto especial, que permite una longitud ilimitada, pero a cambio es más lento que el campo de texto normal, por lo que sólo se usa en casos muy concretos, en los que la longitud del texto a guardar sea muy variable y no importe que las búsquedas sean lentas. Es el caso de un apartado de “observaciones” sobre un alumno, o el “resumen” de una película.
- Otros menos habituales nos permitirán guardar imágenes o ficheros en general, números que se incrementen automáticamente, hipervínculos (enlaces a una cierta dirección dentro de nuestro ordenador u otro), etc.

Problemas que hemos evitado con nuestro diseño.

Cuando uno piensa directamente “en tablas”, puede sentirse tentado a crear una única tabla para los cursos (o casi), que contuviera esta información (por ejemplo):

- Nombre del curso
- Duración
- Fecha de comienzo
- Alumno1
- NotaAlumno1
- Alumno2
- NotaAlumno2
- Alumno3
- NotaAlumno3
- Alumno4
- NotaAlumno4
- Alumno5
- NotaAlumno5

Esto es tan **grave** (mucho) como **frecuente**. Hemos dejado espacio para 5 alumnos en cada curso. Veamos los problemas que esto puede provocar:

- Si en algún curso hay menos de 5 alumnos, estamos desperdiciando el espacio que habíamos reservado para los demás.

- Si necesitamos que entren más de 5 alumnos a un mismo curso, estamos desbordando lo que habíamos previsto y tenemos que empezar a hacer “parches” (¿dos fichas para el mismo curso?).
- Si queremos buscar qué cursos ha realizado un cierto alumno, tendremos que recorrer todos los registros, mirando en los campos (atributos) alumno1, alumno2, alumno3, alumno4, alumno5. Y se complica todavía más si en vez de una simple búsqueda queremos obtener un listado que incluya varios datos.

En nuestro diagrama podemos ver que nuestra base de datos permite que un curso tenga 1 alumno, 50 alumnos o incluso ninguno, y en ningún caso estaremos desperdiciando espacio. Para buscar los alumnos que han asistido a un cierto curso o incluso para crear un listado con esta información, sólo tendremos que recorrer la tabla “Asistir”.

Problemas que aún tiene nuestro diseño.

Hemos evitado algunos errores frecuentes, pero nuestra base de datos todavía no es todo lo buena que debería.

Por ejemplo, si miramos el primer estudio que hemos hecho de los atributos que nos interesaría almacenar, vemos que entre un alumno y un profesor hay muchos datos en común: ambos tienen un nombre, un DNI, una dirección, un teléfono, una ciudad...

Y de hecho, si uno de nuestros profesores fuese a su vez alumno de otro curso, tendríamos sus datos por duplicado, con todos los problemas y riesgos que esto conlleva (problemas de espacio desperdiciado y riesgos de que si cambia alguno de sus datos no nos acordemos de cambiarlo en todas las tablas en las que aparece, y esto daría lugar a inconsistencias en nuestros datos).

Esto se debe a que no hemos profundizado todo lo que deberíamos. En realidad, tanto alumnos como profesores son “personas”. Esto es lo que se conocen como una “**generalización**”, y se suele resolver creando una tabla que contendría todos los datos que son comunes a las “personas”, y conservando en las tablas de “alumnos” y de “profesores” sólo los datos que realmente son exclusivos de cada uno de ellos (además del DNI, que nos permitiría enlazar los datos que un alumno tiene por ser alumno con los que tiene por ser persona). Pero esto es hilar

demasiado fino para unos principiantes como nosotros, así que lo dejaremos como está.

También se le podría buscar alguna otra cosa mejorable, pero tampoco lo haremos... por ahora.

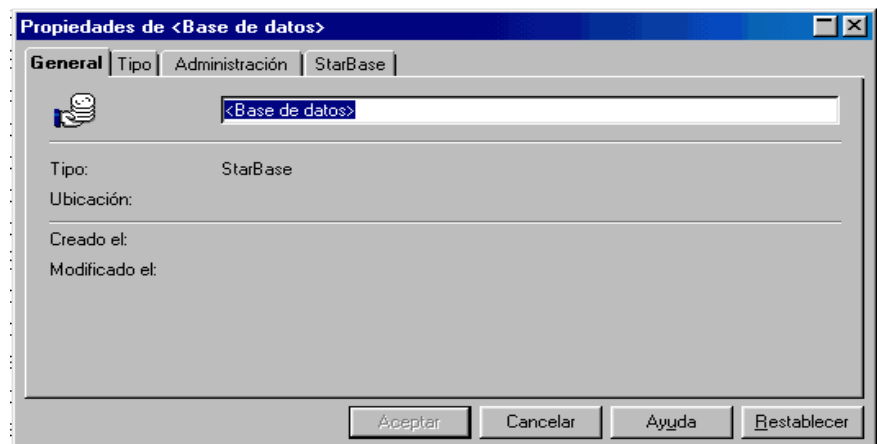
Creando la base de datos con StarBase (StarOffice).

Vamos a ver los pasos que deberíamos dar para plasmar nuestra base de datos empleando distintos sistemas de gestión de bases de datos.

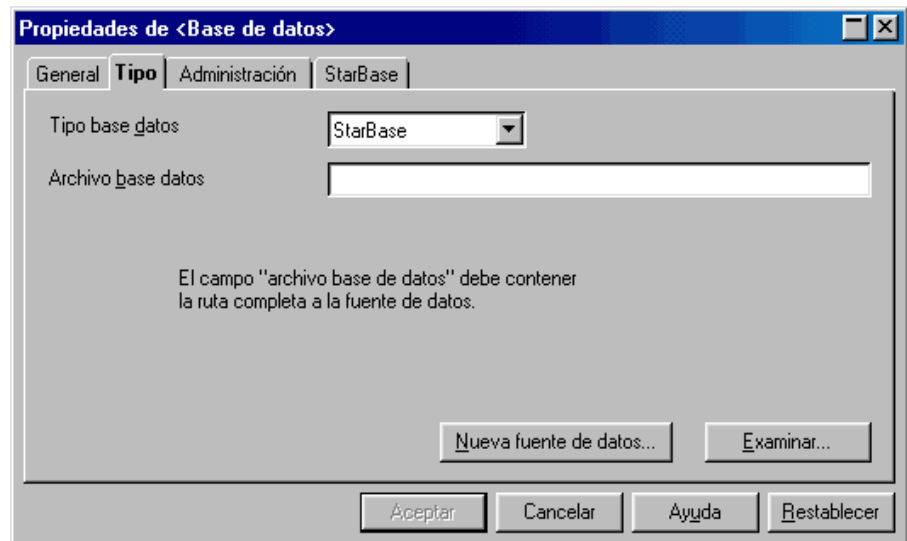
El primero que utilizaremos es **StarBase** 5.1, la base de datos que incorpora **StarOffice** 5.1, un paquete integrado de libre distribución (“gratis”) creado por StarDivision, propiedad de Sun hoy en día.

El manejo de StarBase recuerda mucho al de Access, pero es menos intuitivo. Aun así, como StarBase es “gratis” y Access no lo es, he considerado que será más fácil de conseguir y he decidido empezar por el primero.

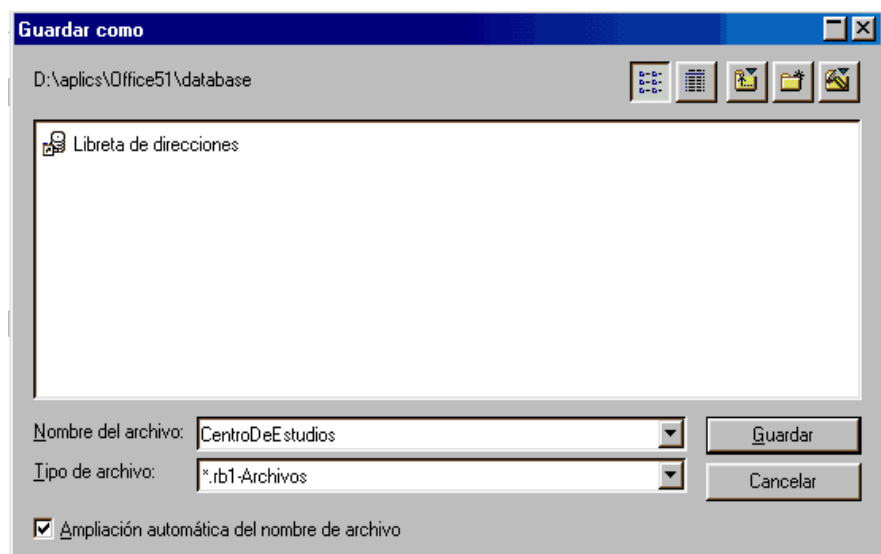
Para crear una base de datos, entramos a StarOffice y escogemos la opción “Nuevo / Base de Datos” del menú “Archivo”. Aparecerá una ventana como ésta:



Eso sí, esta ventana tiene algo desconcertante: debemos indicar el nombre que queremos que tenga el fichero que guardará nuestra base de datos, pero la casilla para indicar este nombre aparece al pulsar la pestaña “Tipo”:

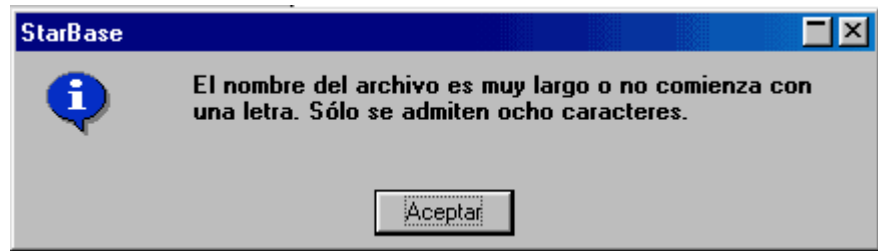


Deberemos escribir el nombre (incluyendo directorios) en la casilla que dice “Archivo base de datos”, o preferiblemente hacerlo de la forma “sencilla”, que es pulsar el botón “Nueva fuente de datos”:

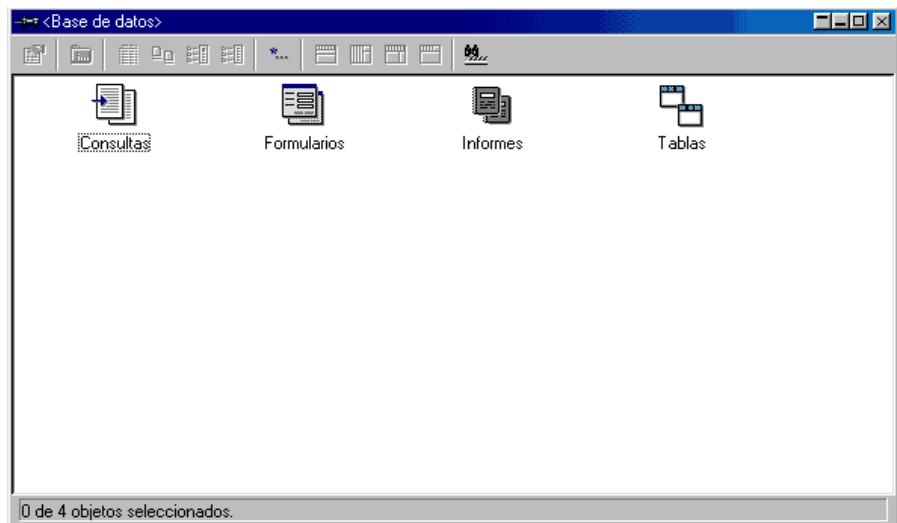


Desde esta ventana resultará más cómodo indicar el directorio (o carpeta) en que queremos que quede nuestra base de datos, así como el nombre que deseamos para ella.

Eso sí: habrá que tener en cuenta que muchos sistemas de bases de datos (y es el caso de StarBase) limitarán el número de letras que podemos utilizar para ese nombre, además de que en muchos casos no permitirán espacios ni ciertos símbolos de puntuación:



Una vez que hemos introducido un nombre correcto (por ejemplo CentrEst) nos aparecerá la pantalla principal de StarBase, desde la cual podemos crear y manipular consultas, formularios, informes y tablas.



Los tres primeros los trataremos más adelante, pero las tablas ya las conocemos, así que vamos a pasar a cómo se crearían con StarBase las tablas de nuestra base de datos (el centro de estudios).

No deberemos hacer doble clic sobre el icono de "tablas", porque así lo que conseguiríamos sería pasar a ver las tablas que ya hay disponibles en nuestra base de datos; como aún no hay ninguna tabla creada, esta ventana aparecería vacía (eso sí, si hemos llegado hasta aquí por error o por precipitación, podríamos volver atrás pulsando el botón "Anterior" de la barra de herramientas, que muestra una flecha hacia la izquierda).

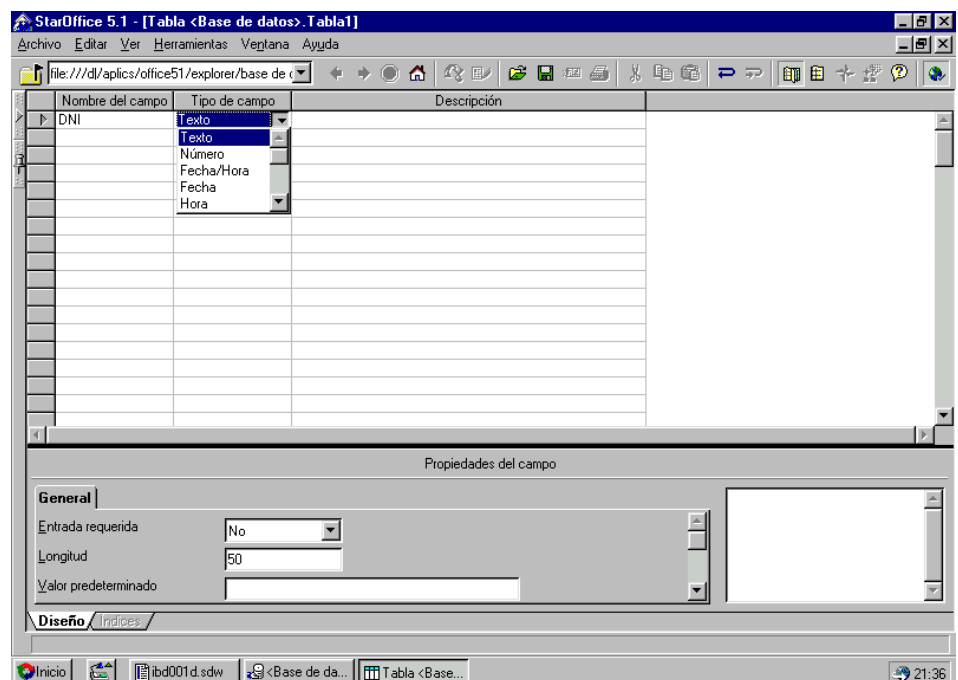


Lo que haremos será pulsar una sola vez sobre el icono de las Tablas (para seleccionar esta opción) y luego pulsar el botón derecho del ratón, lo que nos mostrará las distintas opciones que tenemos disponibles para tablas.

Entre estas opciones aparecerá la opción “Nuevo”, que nos permitirá crear una tabla y nos dará la opción de crearlo desde el “Diseño de tabla” o desde un “Asistente”. En nuestro caso, como ya hemos preparado el diseño “sobre el papel”, prescindiremos del asistente y usaremos el “Diseño de tabla”:

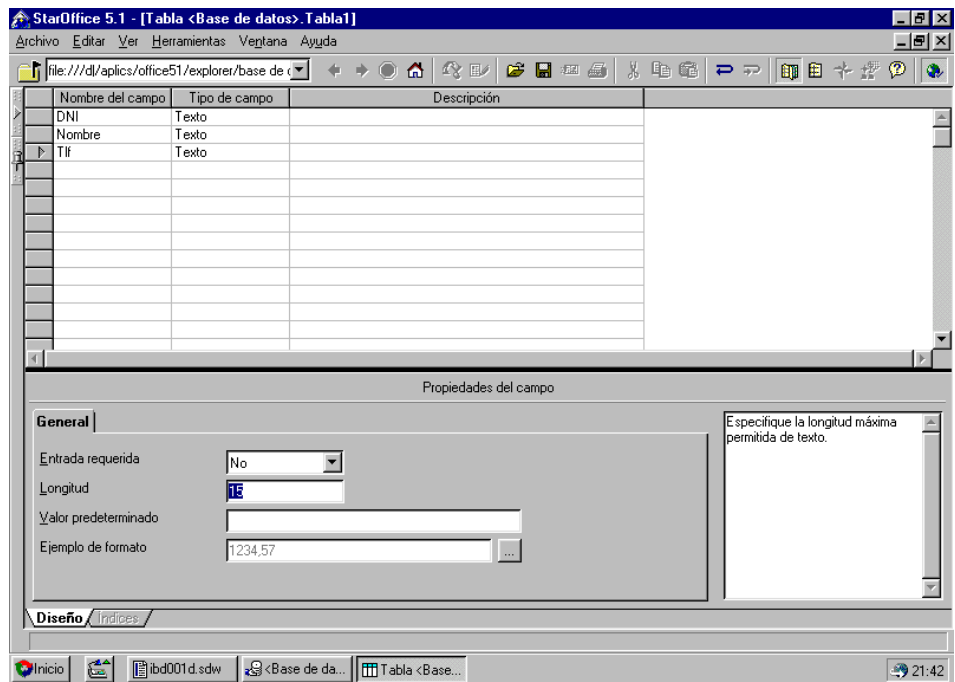


Accedemos a una nueva ventana, en la que realmente definimos nuestra tabla con StarBase. La zona de trabajo de esta ventana tiene tres columnas, en las que indicaremos el nombre del campo, el tipo de datos que va a almacenar y (si queremos) una descripción para ese campo:



Cada fila de la zona de trabajo de esta ventana representará uno de estos campos. En la parte inferior de la pantallas podemos

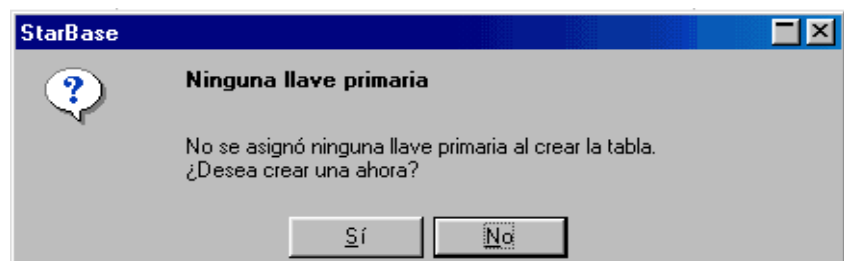
indicar el tamaño que queremos para cada campo. Al final deberíamos obtener algo parecido a:



Cuando terminemos, pulsamos el botón de “Guardar” (el que muestra un diskette) y nos preguntará el nombre que queremos dar a la tabla (en nuestro caso, “Alumnos”):



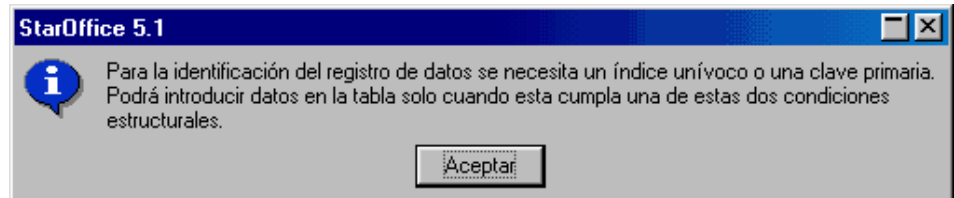
Es muy frecuente que nos aparezca una ventana de aviso como ésta:



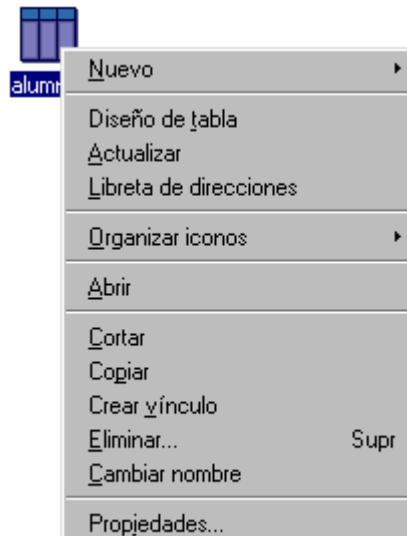
Es cierto, habíamos pensado que el DNI debía ser la clave primaria de nuestra tabla, pero no se lo hemos indicado a StarBase. No queremos que sea StarBase quien ponga la clave

primaria que le plazca, sino nosotros mismos, así que de momento le decimos que “No” queremos clave primaria, y veremos cómo indicarle que ponga como clave el campo que a nosotros nos interesa.

Aparecerá un segundo mensaje de aviso:



Si hubiéramos llegado a salir del diseño de la tabla, deberíamos volver a entrar para indicar la clave primaria. Para volver al diseño, hacemos un clic en la tabla “Alumnos”, y pulsamos el botón derecho, para luego escoger la opción “Diseño de Tabla”:



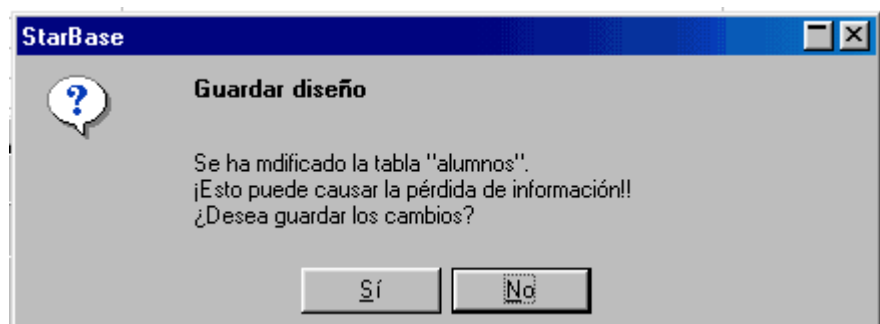
La forma correcta de indicar que un campo será la clave primaria es señalando toda la fila (haciendo clic con el ratón en la columna gris de la izquierda) y pulsando el botón derecho del ratón. Entre las opciones disponibles, aparecerá la de “Llave primaria”:

	Nombre del campo	Tipo de campo	Descripción
Cortar		Texto	
Copiar		Texto	
Pegar		Texto	
Eliminar			
Insertar filas			
Llave primaria			

Entonces StarBase mostrará una llavecita amarilla junto al nombre del campo, para indicarnos que ese campo es el que va a actuar como clave primaria:

	Nombre del campo	Tipo de campo	
?	DNI	Texto	
	Nombre	Texto	
	Tlf	Texto	

Al salir del diseño de la tabla (por ejemplo, pulsando la X de la esquina superior izquierda de la ventana), StarBase ve que hemos modificado el diseño de la tabla y nos pide confirmación para guardar los cambios (deberemos decirle que “Sí”):



Ahora ya podemos entrar a la tabla (con doble clic, o escogiendo la opción “Abrir” del menú contextual pulsando el botón derecho del ratón-). Y comenzar a introducir registros:

	DNI	Nombre	Tlf
	11111111	Juan Pérez	111-11-11
*			

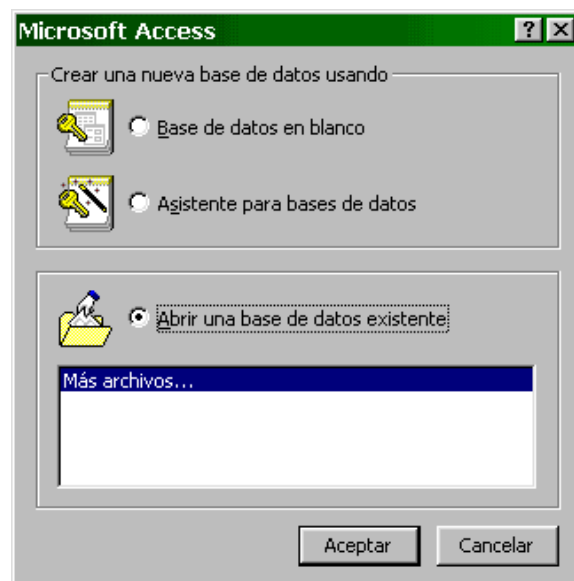
De igual forma crearíamos las otras tablas, tanto la de alumnos y cursos, como la que representa a la relación “Asistir”.

Algo más adelante veremos cómo sacar partido a estos datos. La gran utilidad de las bases de datos es la facilidad para recuperar información de entre muchísimos datos posibles. Esto lo conseguiremos básicamente con las consultas, que veremos después. También veremos entonces cómo detallar las relaciones que habíamos previsto en el diseño.

Creando nuestra BD con Access.

Emplearemos Access 97, aunque los cambios a realizar para versiones anteriores o posteriores deberían ser mínimos.

Al igual que hicimos con StarBase, vamos a dar por supuesto que ya hemos diseñado nuestra base de datos en papel. Entonces el primer paso es evidente: entrar a Access. Nos aparecerá una pantalla como ésta:



En ella se nos pregunta si queremos abrir una base de datos ya existente (no es nuestro caso) o bien crear una nueva (que sí es lo que nos interesa).

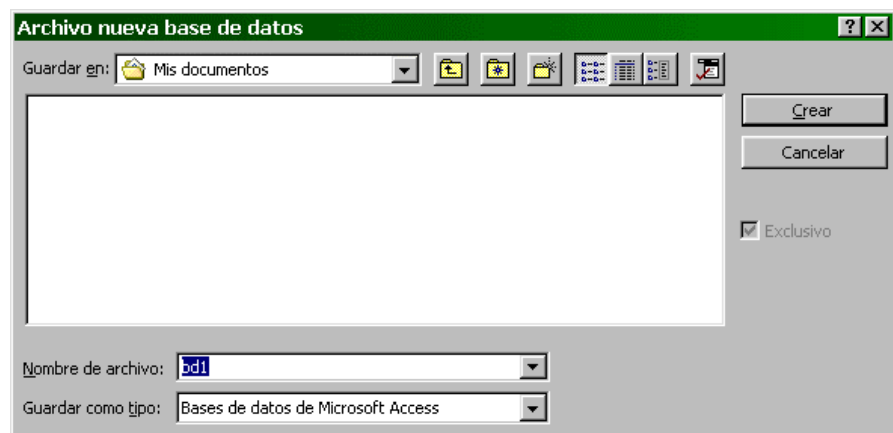
Dentro de una “nueva” base de datos, tenemos dos opciones:

- Emplear un “**asistente**”, que nos propondría varios casos típicos, con bases de datos habituales, nos mostraría la lista de las tablas que considera adecuadas y los campos que podemos incluir (estos sí se pueden modificar, pero las tablas no), nos

permitiría escoger una serie de diseños para formularios e informes, etc. Veremos estos asistentes más adelante, cuando ya sepamos algo más sobre diseño, y cuando hayamos tratado los formularios y los informes, de modo que podremos comparar lo que nosotros habríamos hecho con lo que Access propone.

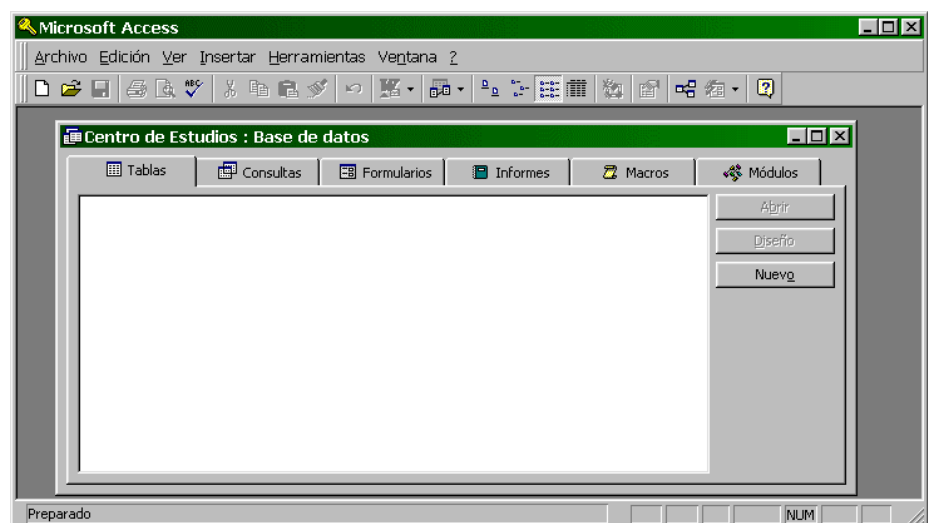
- La otra opción, la que nosotros emplearemos, es la de crear una “base de datos **en blanco**”. Esto nos permitirá partir “de cero”, para indicar a Access exactamente qué es lo que deseamos.

Entonces nos preguntará el nombre que queremos dar al fichero de base de datos:



Este nombre será más flexible que con Starbase: sí podrá contener espacios y más de ocho letras.

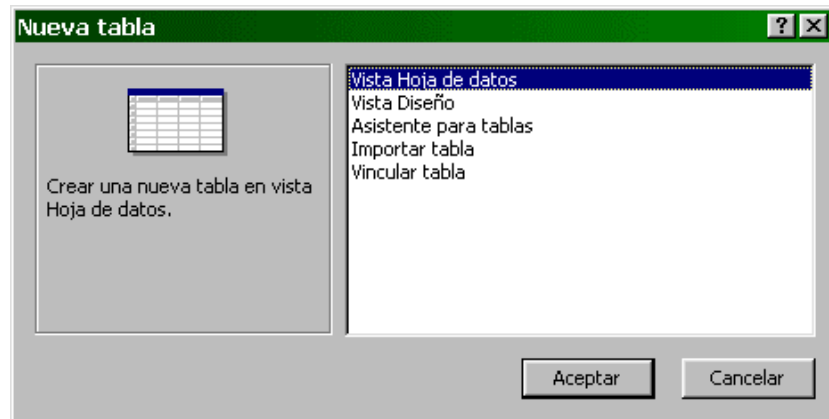
Tras teclear este nombre, veremos la pantalla principal de Access:



En la zona central de la pantalla, la zona de trabajo, vemos una serie de pestañas que recuerdan a las que ya habíamos comentado

para Starbase: Tablas, consultas, formularios, informes, macros y módulos. Estas dos últimas opciones son nuevas para nosotros, y también las veremos más adelante.

De momento, vamos a comenzar por **crear las tablas**. Para ello, comprobamos que nos encontramos en la pestaña Tabla (debería ser así), y pulsamos el botón “Nuevo” (en estos momentos, debería ser el único disponible). Aparecen más preguntas...



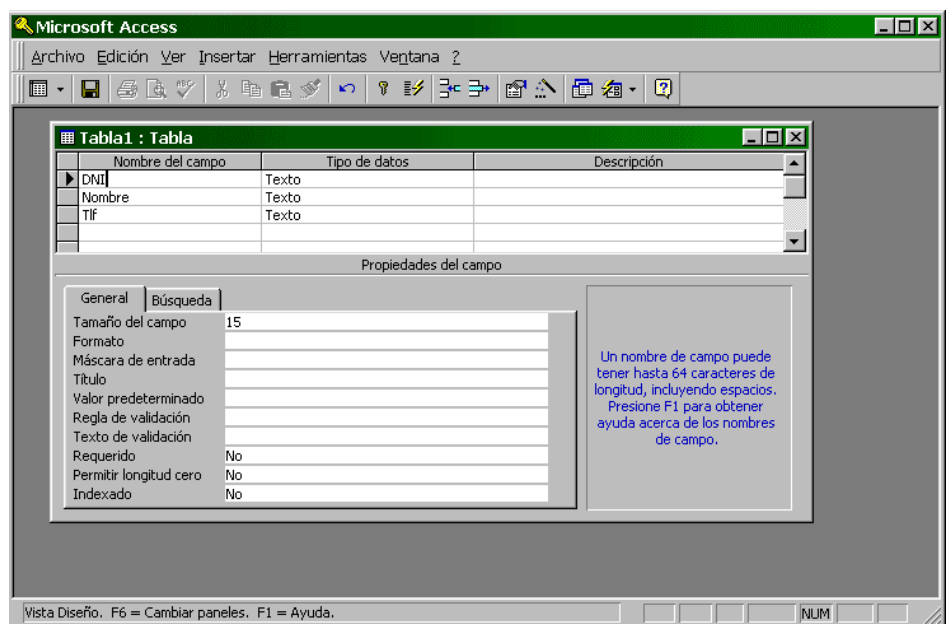
Vamos a comentar brevemente las opciones que tenemos:

- **“Vista hoja de datos”** mostraría una cuadrícula en la que podremos empezar directamente a escribir datos. No es lo que nos interesa, ya que nosotros hemos planificado previamente para que nuestra base de datos se ajuste lo más posible a lo que realmente necesitamos.
- **“Vista diseño”** es lo que buscamos nosotros. Nos centraremos en ella un poco más adelante...
- **Asistente para tablas.** Nos mostraría las tablas más frecuentes, junto con los campos que Access nos recomienda para cada una de ellas. Al igual que el asistente para bases de datos, lo veremos más adelante, cuando tengamos más conocimientos y podamos comparar la opinión de Access con la nuestra.
- **Importar tabla:** nos permitiría aprovechar los datos existentes en una tabla creada con otro sistema de bases de datos más antiguo. Access crearía una nueva tabla con campos equivalentes a los originales, y con los datos que esa tabla contenía.
- **Vincular tabla:** es similar a “importar”, con la diferencia de que no se crea una nueva tabla en Access, sino sólo un enlace a la “antigua”, de modo que los cambios que hagamos en la antigua se reflejarán en nuestra base de datos de Access (a

cambio, resultará algo más lento, y debemos llevar cuidado de no borrar la tabla antigua ni hacer ninguna modificación

En nuestro caso, deberemos seleccionar la opción “Vista diseño” y pulsar el botón “Aceptar”.

Entonces aparece una pantalla vacía en la que, al igual que ocurría con StarBase, iremos introduciendo el nombre de cada campo, el tipo de datos que contiene, y (opcionalmente) una descripción para ese campo (que aparecería en la parte inferior de la pantalla cuando el usuario estuviera introduciendo datos).



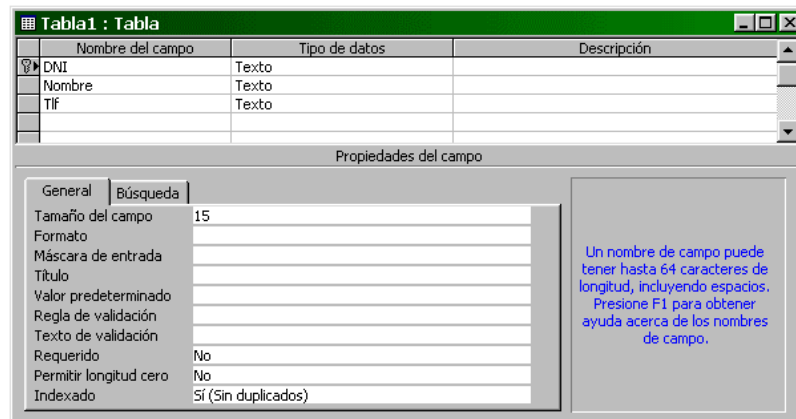
En la parte inferior de la pantalla es donde indicamos el tamaño que queremos para ese campo. Por ejemplo, como se ve en la figura anterior, nosotros queremos que el campo DNI sea texto con una longitud máxima de 15 caracteres.

Un poco más adelante veremos los tipos de datos que nos permite emplear Access. Como para esta tabla sólo necesitaremos usar campos de texto, vamos a dar antes los pocos pasos que nos faltan para completarla.

Una vez que hayamos escrito los nombres de los tres campos, sus tipos de datos correspondientes y sus tamaños, pasamos a indicar cual será el campo que actúe como **clave**. Lo conseguiremos señalando el campo correspondiente (en este caso, el DNI) y pulsando el botón que muestra la imagen de una llave.

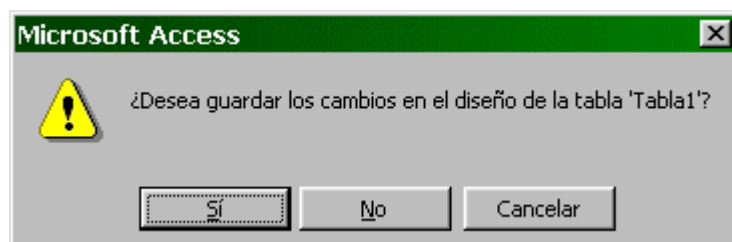


Al igual que ocurría con StarBase, Access mostrará el dibujo de la llave junto a ese campo, para indicarnos que ese será el campo que actuará como clave:

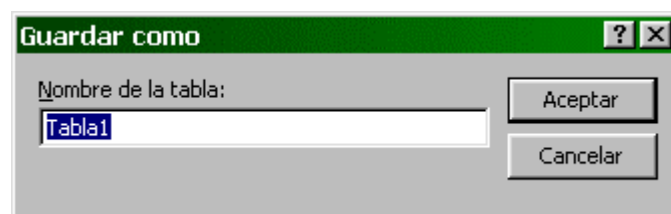


(**Nota:** si en un campo no necesitásemos clave, bastaría con no indicarla ahora, y responder “No” cuando Access nos pregunte si queremos que añada automáticamente una clave).

Para guardar la tabla, basta con pulsar la X de su esquina superior izquierda. Access se da cuenta de que vamos a cerrar sin guardar y nos pide confirmación:



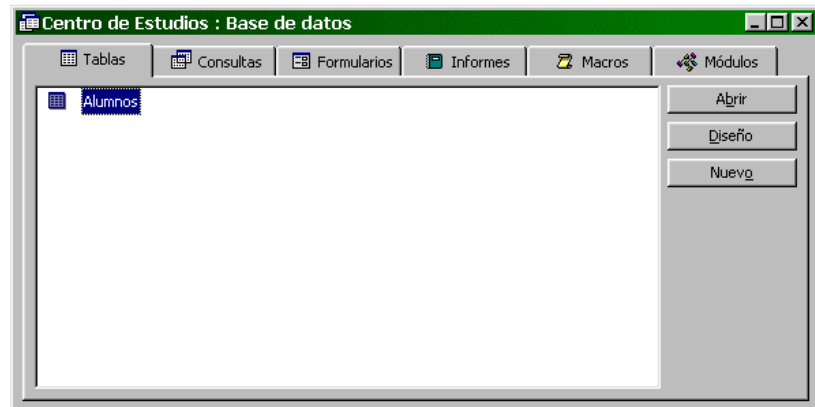
Diremos que “Sí” queremos guardar, y entonces nos preguntará el nombre que tendrá la tabla:



Nota 1: Al igual que ocurría con la base de datos, el nombre de la tabla podrá contener espacios, eñes y vocales acentuadas, y una longitud superior a las ocho letras.

Nota 2: También podríamos haber guardado sin salir, pulsando el botón que muestra el diskette, en vez de pulsar el de la X.

Finalmente, la pantalla de trabajo de Access debería tener una apariencia similar a ésta:



Antes de seguir avanzando con las consultas o con otras bases de datos distintas, vamos a echar un vistazo a los **tipos de datos** que permite Access.

Es esta tabla sólo hemos empleado campos de tipo “Texto”, pero al pinchar en la casilla “Texto” vemos que aparece una flecha hacia abajo, indicándonos que tenemos más opciones que podemos escoger. Si desplegamos esta lista, aparecen todos los tipos de datos que Access permite:

Nombre del campo	Tipo de datos
DNI	Texto
Nombre	Texto
TIF	Memo
	Númérico
	Fecha/Hora
	Moneda
	Autonumérico
	Sí/No
	Objeto OLE
	Hipervínculo
	Asistente para búsquedas...

La mayoría de ellos los hemos comentado ya, pero vamos a repasar todos brevemente:

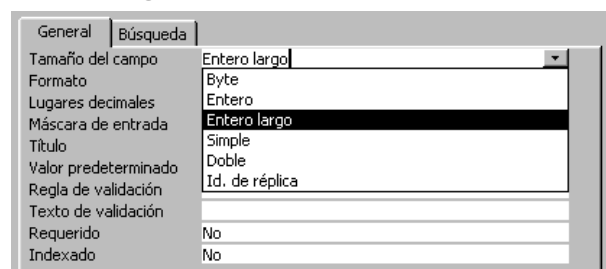
- **Texto:** letras, números y otros símbolos, con una longitud máxima definida, que no podrá exceder los 255 caracteres.
- **Memo:** similar al texto, pero de longitud indefinida (más versátil pero más lento que el campo Texto).
- **Númérico:** valores exclusivamente numéricos, con o sin decimales.
- **Fecha/Hora:** fechas y/o horas.

- **Moneda:** Un valor numérico con pocos decimales pero con muchas cifras significativas. Se usará para números “grandes” y con pocos decimales, como los que puede manejar un banco.
- **Autonumérico:** un número que aumenta automáticamente en cada nuevo registro: en el primer registro valdrá 1, en el segundo valdrá 2, y así sucesivamente. Hay gente a la que le gusta usar este tipo de campos como clave primaria (en nuestro caso, podría ser el código de alumno).
- **Si/No:** un campo que sólo puede tener un valor entre esos dos. Es más correcto que usar una S o una N (texto de una letra), porque ocupa menos espacio, no existirá problemas de mayúsculas y minúsculas, ni siquiera de “intercionalización” (por ejemplo, en un país de habla inglesa, se convertiría automáticamente en Yes y No).
- **Objeto OLE:** cualquier otro tipo de objeto “incrustado”, Nos permitirá guardar imágenes, fragmentos de video, etc. Eso sí, el tamaño del fichero resultante crecerá rápidamente. (OLE es la abreviatura de Object Linking and Embedding)
- **Hipervínculo:** un enlace al estilo de los que se emplean en Internet.
- Asistente para búsquedas: nos permitiría que en ese campo lo que apareciese fuese una lista de valores predefinidos.

Nosotros usaremos básicamente los cuatro primeros: texto, numérico, fecha/hora y memo.

El tamaño en los campos de texto no tiene dificultad, pero en los campos numéricos sí, porque Access no los mide según el número de cifras, sino según cómo se guardarán realmente los datos dentro del ordenador.

Por eso, si escogemos un tipo de datos numérico y miramos en la casilla de “tamaño” veremos que aparece la expresión “Entero largo”. Los posibles valores son:



- Byte: números enteros (sin decimales) entre 0 y 255 (ambos inclusive). El espacio que ocupará será de un byte (como su nombre indica).
- Entero: números enteros entre -32.768 y 32.767. Ocupa 2 bytes.

- Entero largo: números enteros entre $-2.147.483.648$ y $2.147.483.647$. Ocupa 4 bytes.
- Simple: números reales (con decimales) de “simple precisión”, con valores posibles entre $1,4 \cdot 10^{-45}$ y $3,4 \cdot 10^{38}$, con una precisión de 7 cifras. Ocupa 4 bytes.
- Doble: números reales de “doble precisión”, con valores posibles entre $1,79 \cdot 10^{308}$ y $4,94 \cdot 10^{-324}$, con una precisión de 15 cifras. Ocupa 8 bytes.

Creando nuestra BD con WinSQL.

El lenguaje **SQL** es un lenguaje genérico para manipular bases de datos, que está aceptado por la gran mayoría de sistemas de bases de datos actuales.

Por eso, me parece muy conveniente que veamos cómo crear nuestra base de datos en cualquier base de datos que soporte el lenguaje SQL.

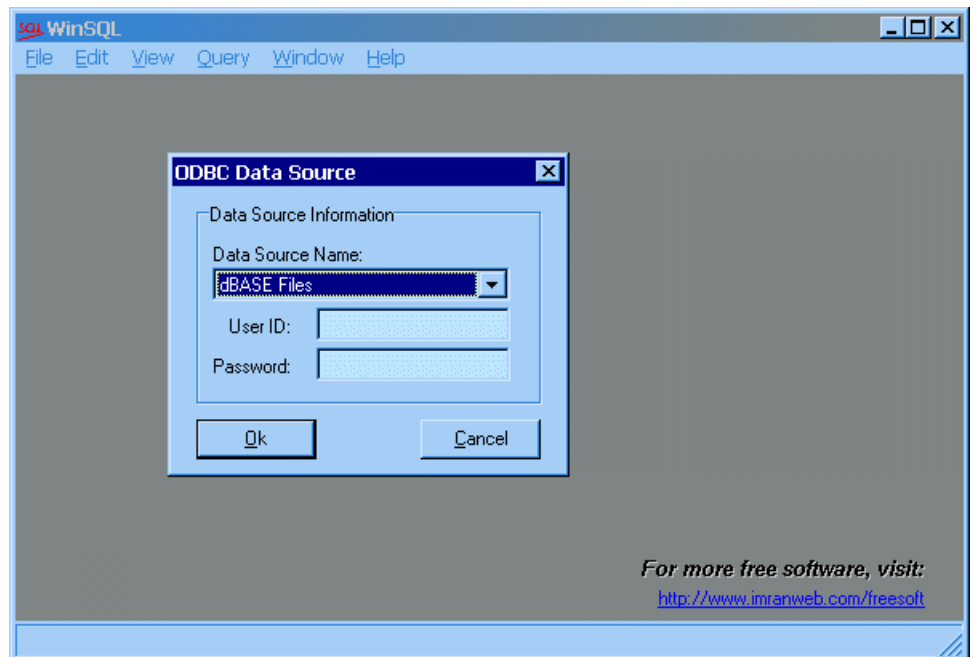
La propia Access permite emplear este lenguaje, pero como ya hemos manejado Access de la forma más sencilla (y también la más habitual), utilizaremos otra distinta.

Yo he optado por **WinSQL**, que es una base de datos para sistema operativo Windows, que se utiliza mediante lenguaje SQL, y que se puede descargar gratuitamente desde Internet:



Al entrar a WinSQL se nos pregunta qué tipo de bases de datos vamos a utilizar. Tenemos muchas posibilidades distintas, pero podemos emplear ficheros de “dBase”, por ejemplo. Esta es una base de datos que todavía no hemos manejado, pero es un auténtico estándar, y si creamos ficheros de este tipo es casi seguro

que después podremos “aprovecharlos” desde cualquier otra base de datos más moderna.



Entonces nos aparecerá una ventana vacía, en la que nosotros deberemos escribir las órdenes en lenguaje SQL.

Por ejemplo, para crear nuestra tabla “alumnos”, en la que tengamos los campos dni (10 letras), nombre (40 letras) y tlf (teléfono, 15 letras), la orden que se usaría es:

```
create table alumno
(dni char(10),
nombre char(40),
tlf char(15))
```

Para que nuestro ejemplo quede correcto, todavía falta algo. En la tabla de los “alumnos”, queríamos que el DNI actuase como clave primaria. Esto, en la mayoría de las versiones de SQL se conseguiría añadiendo la siguiente línea al final de la orden anterior (antes del último paréntesis):

```
primary key (dni)
```

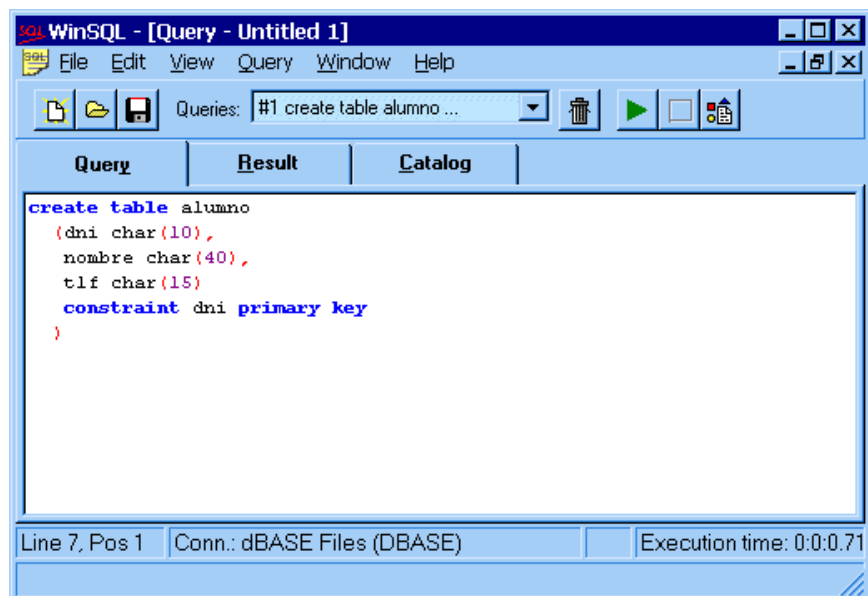
Pero resulta que WinSQL usa el “motor” de bases de datos creado por Microsoft (el mismo que emplea Access), que tiene alguna

(pequeña) diferencia de sintaxis con el SQL estándar, de manera que tendremos que hacerlo de otra forma.

La manera utilizada por WinSQL es indicando esa clave primaria como una “restricción” (“constraint”), así:

```
create table alumno
(
  dni char(10),
  nombre char(40),
  tlf char(15)
  constraint dni primary key
)
```

Nuestra ventana de trabajo de WinSQL debería estar quedando como ésta:



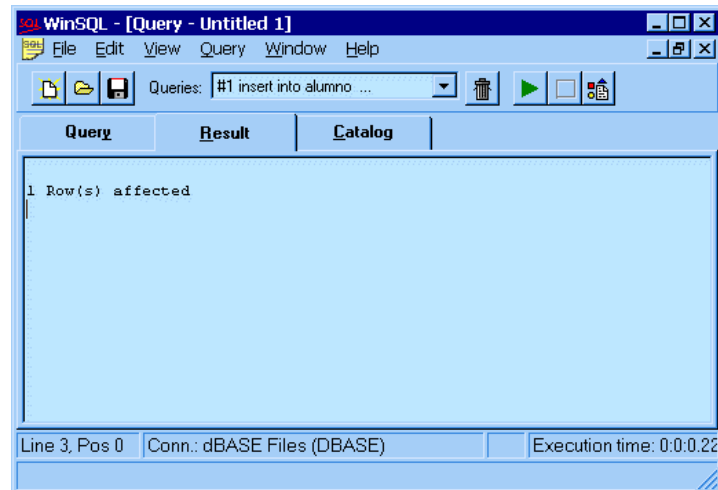
En cualquier caso, lo hagamos como lo hagamos, WinSQL responderá con un mensaje que nos indica que lo ha hecho, pero que no hay datos que mostrar en pantalla.

A la hora de **añadir datos**, se usa la orden “insert”. Su forma más sencilla consiste en indicar el nombre de la tabla, y, entre paréntesis, los valores para cada uno de los campos, en el orden en que los habíamos definido al crear la tabla:

```
insert into alumno
values ('555', 'Juan', '111-11-11')
```

Los valores los he indicado entre comillas porque son texto (cadenas de caracteres); si alguno de los campos fuera numérico, no habría que poner esas comillas.

Como hemos añadido un registro, WinSQL nos responderá diciendo que una fila de la tabla se ha visto afectado por estos cambios:



Siguiendo este mismo esquema podríamos añadir alguna ficha más. Finalmente, podríamos **visualizar los datos** obtenidos. Para ello se emplea la orden “**select**”:

```
select * from alumno
```

Esta orden mostrará todos los datos (es lo que se indica con el asterisco *) de la tabla llamada “alumno”. Obtendríamos un resultado parecido a éste:

DNI	NOMBRE	TLF
555	Juan	111-11-11
567	Jose	222-22-33

2 Row(s) affected

También podríamos pedir que se nos muestre sólo alguno de los campos. Por ejemplo, si queremos ver primero el teléfono y luego el nombre (y que no se muestre el DNI) escribiríamos

```
select tlf, nombre from alumno
```

y la respuesta sería algo como:

tlf	nombre
111-11-11	Juan
222-22-33	Jose

2 Row(s) affected

La orden “select” permite hacer muchas más cosas, pero las veremos a su debido tiempo, cuando tratemos la forma de realizar consultas a partir de los datos.

Creando nuestra BD con dBase III+.

(Apartado todavía no disponible).

Un segundo ejemplo paso a paso.

Enunciado del segundo ejemplo.

Ahora veamos un segundo problema:

“Se desea informatizar un videoclub. En el se dispone de diversas películas, que los clientes pueden alquilar. De cada película puede haber más de un ejemplar. Estos ejemplares pueden estar en el mismo formato (VHS, DVD, Beta, etc.) o en formatos distintos. Queremos poder saber detalles como las películas que tiene alquiladas un cierto cliente y desde qué fecha, o las que había alquilado con anterioridad, o las veces que se ha alquilado una cierta película, o cuando es la última vez que se alquiló, o cuantas películas se han alquilado en formato DVD”.

Diseño del segundo ejemplo.

(Por ahora, todavía queda propuesto como ejercicio).

Implementación del segundo ejemplo.

(Apartado todavía no disponible).

Visualizando datos: consultas básicas y relaciones.

(Apartado todavía no disponible).

Listados e informes.

(Apartado todavía no disponible).

Otras notaciones en Entidad-Relación.

(Apartado todavía no disponible).

Versiones de este texto

La última versión de este curso estará disponible en mi página Web, en la dirección:

<http://www.pobox.com/users/ncabanes>

Este texto es “**GraciasWare**”. Esto quiere decir que si te gusta y/o te resulta útil, basta con que me mandes un correo electrónico a mi dirección ncabanes@pobox.com. Si veo que tengo apoyo moral, procuraré ir mejorando este texto y lanzando nuevas versiones, cada vez más ampliadas.

Las **versiones existentes** hasta la actual han sido:

- 0.03, 29 Julio 2001. Añadido un subapartado sobre cómo crear las tablas desde el lenguaje SQL, empleando WinSQL. Propuesto el enunciado del segundo ejemplo, que todavía no

está resuelto. Añadida una nota sobre que existen otras notaciones para el modelo Entidad-Relación, aunque todavía no he dado más detalles sobre ellas. Ocupa 38 páginas.

- 0.02, 13 Septiembre 2000. Añadido un subapartado sobre cómo crear las tablas con Access. Revisada la ortografía, para evitar alguna errata como “direcotrio”, aritméricas” o “incuso”. Cambiada la numeración de las páginas (para que la portada no cuente como página 1). Ocupa 33 páginas.
- 0.01, 07 Agosto 2000. Primera versión, ocupa 25 páginas (incluyendo índices) y contiene los tres primeros apartados (introducción, nociones básicas de diseño, convirtiendo el diseño a tablas), hasta el subapartado de cómo crear las tablas con StarBase.

Las **próximas mejoras** serán:

- Introducción a las consultas.
- Cómo reflejar las relaciones en la base de datos.
- Cómo crear las tablas con dBase III+ y otras bases de datos “antiguas”.

Índice Alfabético.

Access, 24
Asistente para tablas, 26
atributos, 9
Autonumérico, 30
base de datos, 3
campo, 3
cardinalidad, 7
char, 32
clave, 10
clave principal, 11
claves alternativas, 11
constraint, 33
create table, 32, 33
Diseño básico, 5
EER, 8
entidades, 6
Entidad-Relación, 6
Entidad-Relación Extendido, 8
Fecha (tipo de datos), 15
Fecha/Hora, 29
from, 34
generalización, 16
GraciasWare, 5
Hipervínculo, 30
Importar tabla, 26
insert into, 33
into, 33
Lógico (tipo de datos), 14
Memo, 29
Memo (tipo de datos), 15
Moneda, 30
Numérico, 29
Numérico (tipo de datos), 14
Objeto OLE, 30
OLE, 30
primary key, 32
registro, 3
relacional (base de datos), 3
relaciones, 6
select, 34
SGBD, 4
Si/No, 30
SQL, 31
StarBase, 17
StarOffice, 17
tabla, 3
Texto, 29
Texto (tipo de datos), 14
tipos de datos, 13
values, 33
Vincular tabla, 26
Vista diseño, 26
Vista hoja de datos, 26
WinSQL, 31