Databases

(Bases de Datos, BDA)

Exercise book 1: Introduction to Relational Databases.

Departamento de Sistemas Informáticos y Computación.

2007/2008

Objectives:

- Introduce, in an intuitive way, the concept of relation.
- Introduce, in an intuitive way, the concept of relational database.
- Introduce, in an intuitive way, which function is played by the identifying attributes and the reference attributes.
- Design simple queries over one relation.
- Design simple queries over several relations.
- Design complex queries over two or more relations.
- Get used to the query interfaces in a relation database management system. We will use the MsAccess DMBS.

1. Relational Databases

As can be derived from the aforementioned objectives, the general objective in this session is to "introduce, in an intuitive form, the basic concepts found in relational databases". The presentation which is planned here is completely practical, i.e., through a very simple example we will progressively introduce the different concepts, in such a way that the practical learning facilitates the comprehension of the formal definition seen in the theoretical sessions.

The example used is the management of the teaching assignments in the Escuela Técnica Superior de Informática Aplicada (EI). The description of this information system is as follows:

The University is organised in departments which are responsible for the teaching of courses leading to several university degrese are formed by the lecturers and courses that fall into one discipline (foreign languages, mathematics, physics,...).

The data that must be stored are the data relative to the courses offered at the EI, the lecturers who teach at this school and the departments of the University involved in the teaching of these courses.

The data of the department which must be known are:

** code.*

** name.*

** secretariat's telephone.*

** name of the head of the department.*

On the other hand, the data of each course which must be known are:

** code of the course.*

** name of the course.*

** term in which the course is imparted.*

** N of Credits of the theoretical sessions.*

** N of Credits of the practical sessions.*

** department that is responsible for imparting the course.*

For each lecturer that teaches at the EI we need to know:

∗ *code.*

∗ *name.*

∗ *telephone number (internal).*

∗ *department to which he/she belongs.*

In a relational database, the data are organised in data structures called *relations*. From an informal viewpoint, a relation can be seen as a table, i.e., a set of data (scalar values) which are organised into rows and columns. Next, we present a relation *DEPARTAMENTO* for the entity 'department' represented as a table[1]:

DEPARTAMENTO

cod_dep	**nom_dep**	**director**	**teléfono**
DSIC	Sistemas Informáticos y Computación	V. Botti	3500
DISCA	Ingeniería de Sistemas, Computadores y Automática	A. Crespo	5700
MAT	Matemática Aplicada	P. Pérez	6600
FIS	Física Aplicada	J. Linares	5200
IDM	Idiomas	B. Montero	5300
EIO	Estadística e Investigación Operativa	L. Barceló	4900
OEM	Org. de Empresas, Economía Financ. y Contabilidad	M. Pérez	6800

where "cod_dep" stands for "code de departamento" (department's code), "nom_dep" stands for "nombre de departamento" (department's name), "director" stands for the head of the department and "teléfono" stands for telephone.

From the previous table, we can think of several issues. Each row (in database terminology, the term *tuples* is also used) represents a department and contains its descriptive information, i.e., its code, name, head and telephone. Each descriptor is stored in a column of each department's row (in database terminology, columns are also called *attributes*). As can be observed, the attributes in the relation have an associated data type. The code is a string with a length of 5 characters, the name is also a string with a length of 40 characters, the head is also a string with a length of 30 characters and the telephone is an integer number. The names of the attributes are needed to refer to the stored value in each corresponding tuple to that attribute. In this way, the relation *DEPARTAMENTO* can be expressed as follows:

DEPARTAMENTO(cod_dep: string(5), nom_dep: string(40), director: string(30), teléfono: integer)

As can be seen, the previous expression defines which the **structure** of the relation *DEPARTAMENTO* is. This expression indicates that each tuple of the relation contains four values, one of them of type string(5) (the code), another of type string(40) (the name), another of type string(30) (the head) and another of type integer (the telephone). In order to refer to these values we will use the names corresponding to the attributes, *cod_dep, nom_dep, director* and

[1] The names of the relations and its attributes are left in Spanish since the database we're working with is in Spanish. The version of MsAccess we will use in the labs is also in Spanish, and so are the screens copied as illustrations.

teléfono, respectively. It is also interesting to highlight that **the attribute *cod_dep* is an *identifier* attribute of this relation** since it allows us to identify (distinguish) its tuples in an unequivocal way. Obviously, in order to use an existing attribute in reality as an identifier in a table, it is necessary for that attribute to be unique in the real world, i.e., at the University the codes of the departments are unique for each department. An immediate consequence of this is that it should not exist the case in which we can find two tuples in the relation *Departamento* with the same value for this attribute.

The relation *ASIGNATURA* which is presented as follows includes the relevant data of the courses ("asignaturas"):

ASIGNATURA

cod_asg	**nom_asg**	**semestre**	**teoría**	**prácticas**
BDA	Bases de Datos	2B	3	3
AD1	Algoritmos y Estructuras de Datos 1	1A	4	2
FCO	Fundamentos de Computadores	1A	4,5	4,5
MAD	Matemática Discreta	1A	3	3
INT	Inglés Técnico	1B	3	3
FFI	Fundamentos Físicos de la Informática	1A	3	3
EC2	Estructuras de Computadores 2	2A	3	3

If we observe the tuples of the relation *ASIGNATURA* in detail, we can see that each one represents a course and contains its descriptive data: the code of the course (cod_asg), the name of the course (nom_asg), the term (semestre), the n of theory credits (theory) and the n of practical credits (prácticas). This information is represented and stored in a very similar way as for the relation *DEPARTAMENTO*.

Coming back to the initial description of the information system, we observe that the only data which is left out in *ASIGNATURA* is the department which is responsible for imparting the course.. **Since there is only one department which is responsible for a course, it is reasonable to add an additional attribute to the tuples of the relation *ASIGNATURA* which represents this information**. On the other hand, since the departments are unequivocally identified by their code, the most adequate modification of the relation *ASIGNATURA* will consist of adding a new attribute to each tuple which stores the code of the department in charge of the teaching of the course. Once performed this modification, the relation *ASIGNATURA* remains as follows:

ASIGNATURA

cod_asg	**nom_asg**	**semestre**	**teoría**	**prácticas**	**dep_asg**
BDA	Bases de Datos	2B	3	3	DSIC
AD1	Algoritmos y Estructuras de Datos 1	1A	4	2	DSIC
FCO	Fundamentos de Computadores	1A	4,5	4,5	DISCA
MAD	Matemática Discreta	1A	3	3	MAT

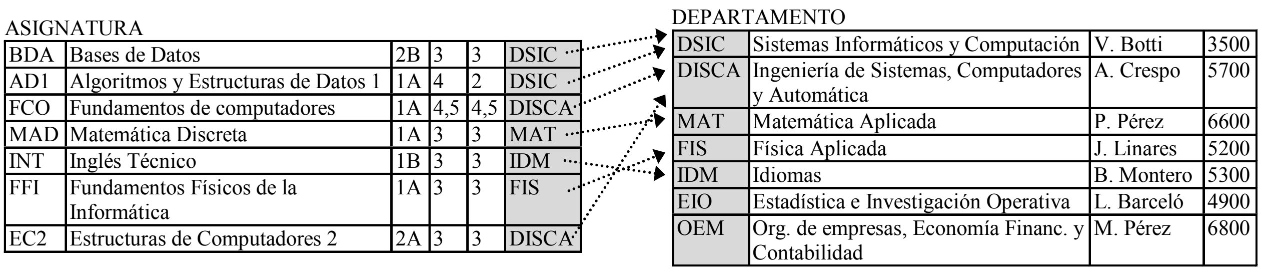
INT	Inglés Técnico	1B	3	3	IDM
FFI	Fundamentos Físicos de la Informática	1A	3	3	FIS
EC2	Estructuras de Computadores 2	2A	3	3	DISCA

So modified, the definition of the relation *ASIGNATURA* is as follows:

ASIGNATURA(cod_asg: string(3), nom_asg: string(40), semestre: string(2),

teoría: real, prácticas: real, dep_asg: string(5))

A detailed analysis of the attributes of this relation shows that this last attribute, *dep_asg*, presents some particular characteristics. Clearly, this attribute establishes a link between the courses and the departments and provides information about which department is responsible for each course. This can be seen graphically as follows:



ASIGNATURA

BDA	Bases de Datos	2B	3	3	DSIC
AD1	Algoritmos y Estructuras de Datos 1	1A	4	2	DSIC
FCO	Fundamentos de computadores	1A	4,5	4,5	DISCA
MAD	Matemática Discreta	1A	3	3	MAT
INT	Inglés Técnico	1B	3	3	IDM
FFI	Fundamentos Físicos de la Informática	1A	3	3	FIS
EC2	Estructuras de Computadores 2	2A	3	3	DISCA

DEPARTAMENTO

DSIC	Sistemas Informáticos y Computación	V. Botti	3500
DISCA	Ingeniería de Sistemas, Computadores y Automática	A. Crespo	5700
MAT	Matemática Aplicada	P. Pérez	6600
FIS	Física Aplicada	J. Linares	5200
IDM	Idiomas	B. Montero	5300
EIO	Estadística e Investigación Operativa	L. Barceló	4900
OEM	Org. de empresas, Economía Financ. y Contabilidad	M. Pérez	6800

Figure 1: Graphical representation of the function of the attribute dep_asg *in the relation ASIGNATURA*

In this way, in each tuple of the relation *ASIGNATURA* the value of the attribute *dep_asg* refers to the department which is responsible for its teaching. These special attributes are usually called *referring attributes.* An important remark is that the attribute *dep_asg* establishes the connection by the value it contains. In this way, the arrows which appear in the diagram above must be understood only as a graphical way of expressing this connection. They are not pointers or other kind of physical connections.

The referring attributes, such as *dep_asg* in *ASIGNATURA*, which allow us to link two relations must always follow the rule that the stored value must be equal to some value that appears in the referred attribute in the referred table, in this case, the attribute *cod_dep* in *DEPARTAMENTO.*

The descriptive information for the lecturer ("profesor") can be represented in a similar way as we have seen for the departments or the courses. In this way, the relation *PROFESOR* would be as follows:

PROFESOR

cod_pro	**nom_pro**	**extensión**
JCC	Juan C. Casamayor Ródenas	7796
RFC	Robert Fuster I Capilla	6789
JBD	José V. Benlloch Dualde	5760
MAF	María Alberola Fernández	3560
CPG	Cristina Pérez Guillot	7439

JTM	José M. Torralba Martínez	4590
IGP	Ignacio Gil Pechuán	3423
DGT	Daniel Gil Tomás	5679
MCG	Matilde Celma Giménez	7756

Taking a look to this relation we can see that the information which corresponds to the code of the lecturers, their name and telephone extension at the University. In the same way as *cod_dep* in *DEPARTAMENTO* and *cod_asg* in *ASIGNATURA* the code of the lecturers ("profesor") identifies in an unequivocal way the tuples of the relation *PROFESOR*.. Clearly, the information is not complete. The information of the department the lecturer belongs to is still missing.. This information is equivalent to the case of the department that is responsible for the teaching of one course. **Hence, we can represent this in the same way as before: we add to each tuple in the relation *PROFESOR* a new referring attribute which corresponds to the code of the department she/he belongs to.** Once this modification is performed, the relation *PROFESOR* remains as follows:

PROFESOR

cod_pro	**nom_pro**	**extensión**	**dep_pro**
JCC	Juan C. Casamayor Ródenas	7796	DSIC
RFC	Robert Fuster I Capilla	6789	MAT
JBD	José V. Benlloch Dualde	5760	DISCA
MAF	María Alberola Fernández	3560	DSIC
CPG	Cristina Pérez Guillot	7439	IDM
JTM	José M. Torralba Martínez	4590	OEM
IGP	Ignacio Gil Pechuán	3423	OEM
DGT	Daniel Gil Tomás	5679	DSICA
MCG	Matilde Celma Giménez	7756	DSIC

The schema of the relation *PROFESOR* is defined as follows:

PROFESOR(cod_pro: string(3), nomprof: string(40), extensión: integer, dep_pro: string(5))

The attribute *dep_pro* in the relation *PROFESOR* connects this relation and the relation *DEPARTAMENTO*. Thus, the stored value in this attribute in each of the tuples should be equal to the value in the attribute *cod_dep* in some tuple of the relation *DEPARTAMENTO*.

Essentially, what we have seen so far corresponds to the fundamental concepts of the relational databases, although presented in a very intuitive and informal way. Next we review these concepts:

- The relations are structures that can be intuitively seen as tables.
- Las relations contain tuples which correspond to the rows in the tables.
- A tuple in a relation contains a set of values which can be referred by the names of the attributes. These attributes correspond to the columns of the tables.

- There are two kinds of special attributes:
  - ⇒ **attributes which identify the tuples in a relation**: *cod_dep* in *DEPARTAMENTO*, *cod_pro* in *PROFESOR* and *cod_asg* in *ASIGNATURA*.
  - ⇒ **referring attributes which associate two relations**: *dep_asg* in *ASIGNATURA* which associates it with *DEPARTAMENTO* or *dep_pro* in *PROFESOR* which associates it with *DEPARTAMENTO*.

2. The database management system Microsoft Access

The software package *Microsoft Office* consists of a set of programs which include *Access*, a relational database management system.

When Access is initiated, we can see something similar to what is shown in Figure 2.

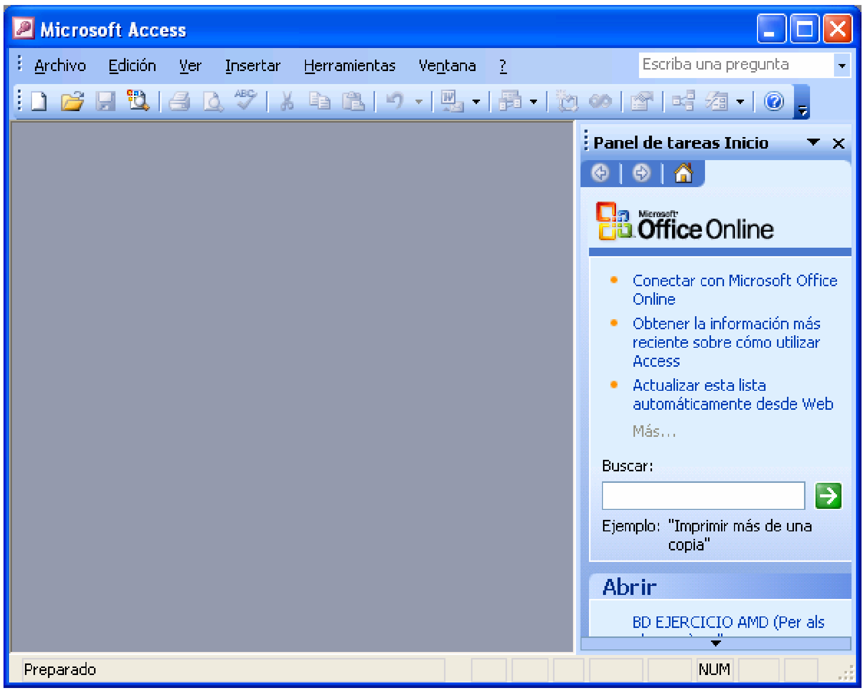

Microsoft Access
Archivo
Edición
Ver
Insertar
Herramientas
Ventana
?
Escriba una pregunta
Panel de tareas Inicio
Microsoft Office Online
Conectar con Microsoft Office Online
Obtener la información más reciente sobre cómo utilizar Access
Actualizar esta lista automáticamente desde Web
Más...
Buscar:
Ejemplo: "Imprimir más de una copia"
Abrir
BD EJERCICIO AMD (Per als
Preparado
NUM

Figure 2: Options when opening MsAccess.

If we select the option "**Archivo -> Abrir**" (File -> Open, we see the dialog box "Abrir" shown in the following Figure 3.

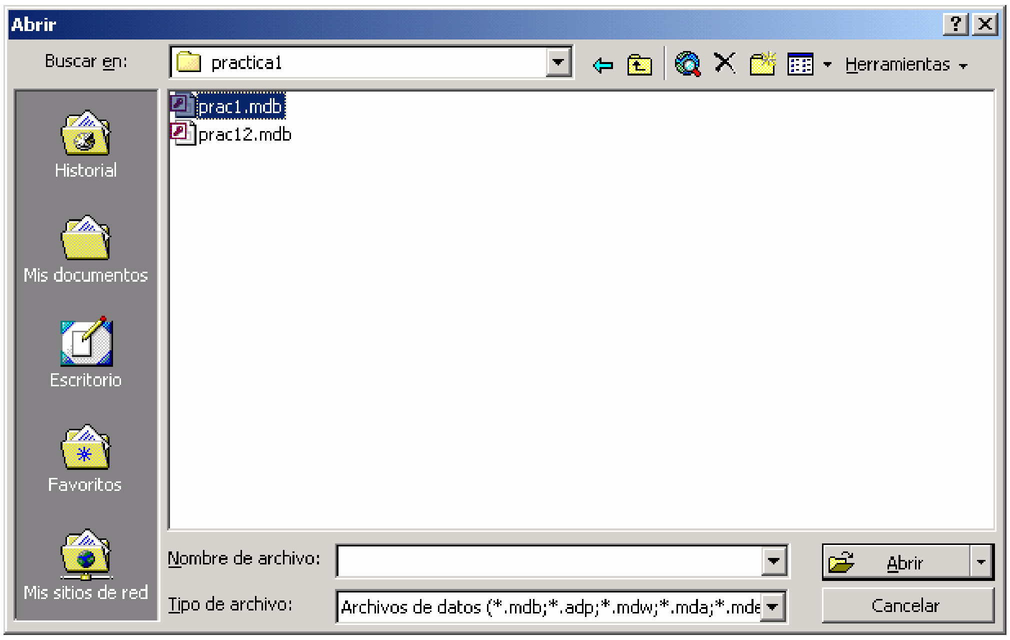


Figure 3: Opening a database.

We will select the database **pract1.mdb**, and we will go to the start menu with the open database (Figure 4).

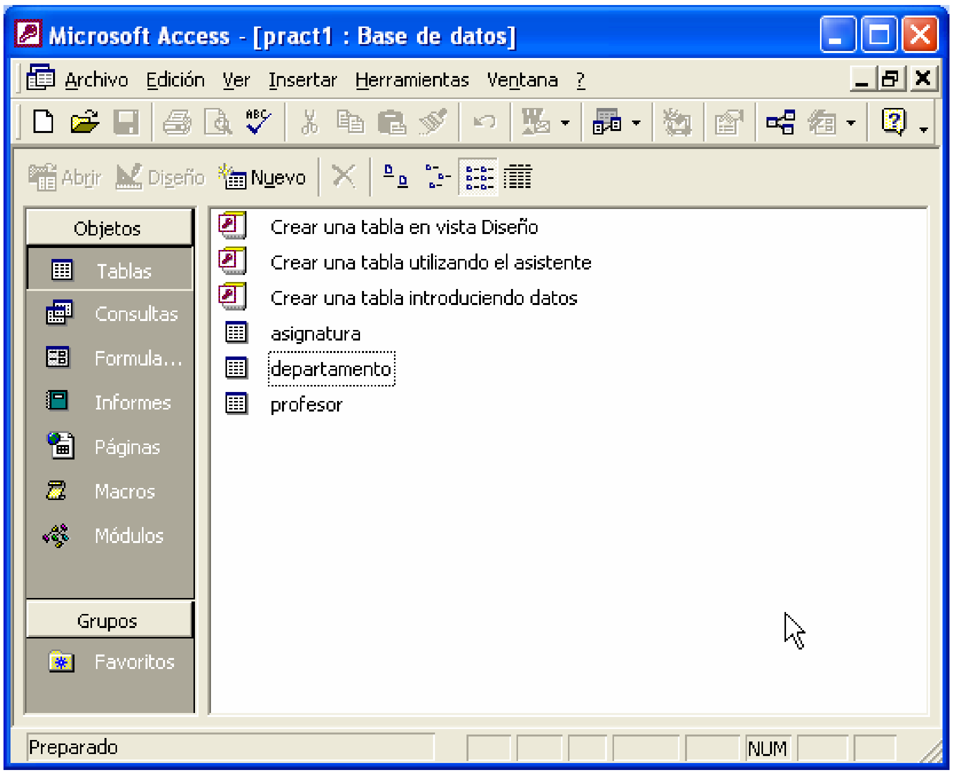

Microsoft Access - [pract1 : Base de datos]
Archivo Edición Ver Insertar Herramientas Ventana ?
Abrir Diseño Nuevo
Objetos
Tablas
Consultas
Formula...
Informes
Páginas
Macros
Módulos
Grupos
Favoritos
Crear una tabla en vista Diseño
Crear una tabla utilizando el asistente
Crear una tabla introduciendo datos
asignatura
departamento
profesor
Preparado
NUM

Figure 4: Start menu after opening the database.

On the left-hand window we can see the main tools we can use:

- **Tablas (tables)**: here we can define and use the several tables that are used to store the data corresponding to the different relations in the database. Each table has a number of columns (fields), which are equivalent to the attributes of the relation, and rows (records), which are equivalent to the tuples of the relation. Additionally, we can establish the attributes which identify in an unequivocal way, the tuples of each relation, relations between tables, etc.

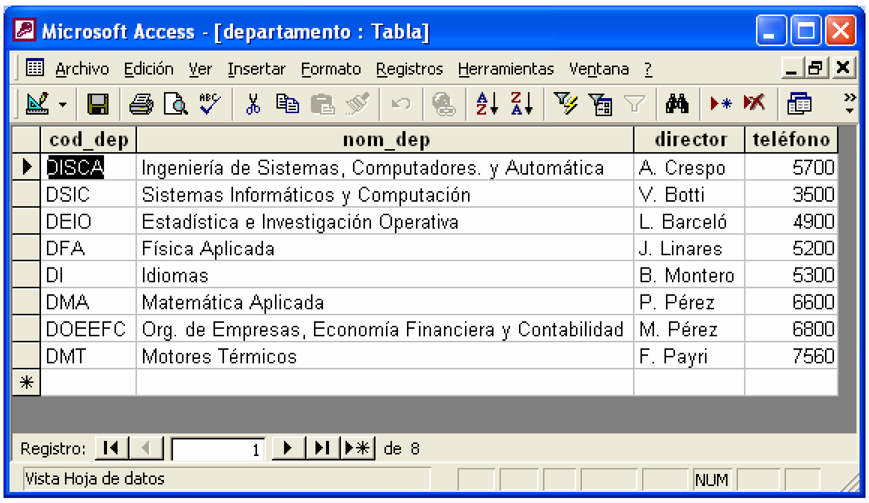
- **Queries (queries)**: provide a personal view for the data stored in one or more tables. We can generate queries over the stored data to select, update, insert or delete data[2]. Every query can be established through a graphical interface o directly writing down the query in a specific language which is provided by the system (*Structured Query Language, SQL*).
- **Formularios (forms)**: can be used to introduce and view data from the tables in a systematic way.
- **Informes (reports)**: allow to present data in a customized and effective way. There can be reports from data in one table or several related tables, or over one or several named queries (known as views). This tool allows the user to give the appropriate format to the document that is to be printed as a report.
- **Macros (macros)**: this tool is useful to create programs in a specific language provided by Access, which allows some tasks to be automated.
- **Módulos (modules)**: this tool is useful to define personal procedures in VBA (Visual Basic for Applications). These procedures can capture errors (this is not possible with macros).

As indicated above, the objective of this practical session will be to use the first two tools, i.e., "TABLAS" and "QUERIES".

2.1. The tool TABLAS

In almost every commercial system, relations are called tables, so this must be taken into account when working with Access. In Figure 4 above we see the screen that corresponds to the start menu for the tool "TABLAS" and shows the list of tables that are available in the database **pract1**. On the left upper corner we can see the three possible options:

- **Abrir (open)**: this option shows the selected table from the list of tables and allows the user to update its data. When we select this option, we can see a window in the view mode "hoja de datos" (data sheet) as shown in Figure 5.

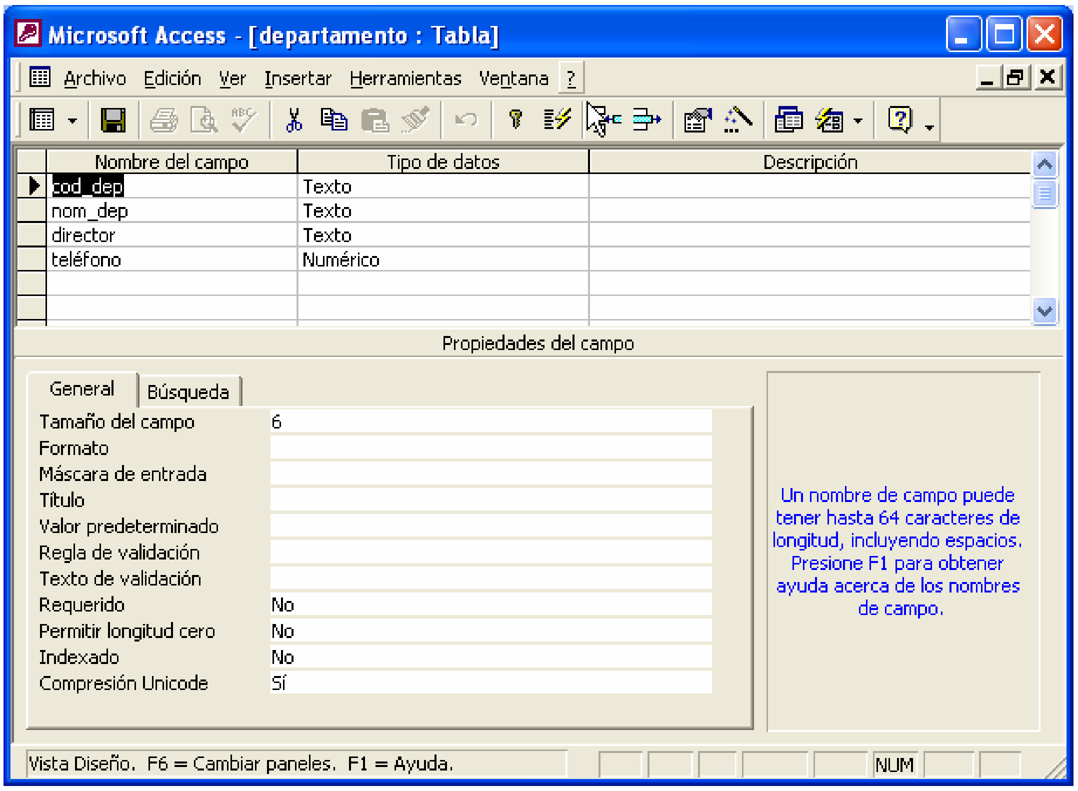

Microsoft Access - [departamento : Tabla]
Archivo Edición Ver Insertar Formato Registros Herramientas Ventana ?
Registro: 1 de 8
Vista Hoja de datos
NUM

cod_dep	nom_dep	director	teléfono
DISCA	Ingeniería de Sistemas, Computadores. y Automática	A. Crespo	5700
DSIC	Sistemas Informáticos y Computación	V. Botti	3500
DEIO	Estadística e Investigación Operativa	L. Barceló	4900
DFA	Física Aplicada	J. Linares	5200
DI	Idiomas	B. Montero	5300
DMA	Matemática Aplicada	P. Pérez	6600
DOEEFC	Org. de Empresas, Economía Financiera y Contabilidad	M. Pérez	6800
DMT	Motores Térmicos	F. Payri	7560

Figure 5: table "departamento" in the view mode *"Hoja de datos" (datasheet).*

[2] Note that in standard SQL and other DBMS, the term "query" is restricted to select/retrieve information from a database, never to operations which modify the information (insert, update, delete) such as in Access.

- **Diseño (design)**: displays and modifies the definition of the selected table from the list of tables. When we select this option, the following window "Vista Diseño" is shown (Figure 6).


Microsoft Access - [departamento : Tabla]
Archivo
Edición
Ver
Insertar
Herramientas
Ventana
?
Propiedades del campo
General
Búsqueda
Vista Diseño. F6 = Cambiar paneles. F1 = Ayuda.
NUM

Nombre del campo	Tipo de datos	Descripción
cod_dep	Texto	
nom_dep	Texto	
director	Texto	
teléfono	Numérico	

Tamaño del campo	6
Formato	
Máscara de entrada	
Título	
Valor predeterminado	
Regla de validación	
Texto de validación	
Requerido	No
Permitir longitud cero	No
Indexado	No
Compresión Unicode	Sí

Un nombre de campo puede tener hasta 64 caracteres de longitud, incluyendo espacios. Presione F1 para obtener ayuda acerca de los nombres de campo.

Figure 6: table "departamento" *in the view mode "Diseño" (design)*

.

- **Nuevo (new)**: allows the user to define a new table.

In what follows, we will present the options *Diseño* and *Abrir* in detail. The option *Nuevo*, which creates new tables, will not be considered in this session.

2.1.1. Examining the definition of a relation: "Diseño"

The option allows the user to examine and modify the definition of a relation in a database. In this way, we can check the structure of a relation. Figure 6 shows the screen which is displayed when we choose this option over the relation *DEPARTAMENTO*.

On the upper side we can observe a table which describes the structure of the chosen relation. Each row in the table corresponds to the definition of an attribute in the relation and each column includes the specific information over that attribute. These columns are the following:

- "Nombre del campo": the name of the attribute.
- "Tipo de datos": the type or domain which is associated with the attribute. Access provides a set of standard types such as: texto (text), numérico (numerical), booleano (boolean) (sí/no (yes/no)), fecha/hora (date/time), moneda (currency), etc.
- "Descripción": description of the attribute.

When selecting a row, on the bottom of the window there is a list of properties which are associated with the attribute described in the row.

The bottom right frame shows a contextual help at every moment which gives hints for the interpretation of each of the objects appearing in the window.

2.1.2. Updating and querying a relation: "Abrir"

With the option *Abrir* we can query the tuples of a relation. Additionally, we can also modify the data in a relation, i.e., insert and delete tuples and update values in their attributes.

When selecting this option, we can see the data that each relation contains in a tabular way. I.e., each row represents a tuple or record in the relation; columns correspond to the values of the attributes, as can be seen in Figure 5. With the arrow keys we can go from one record to another.

Also, we can use the cursor bar on the bottom left part of the window to move on through the data sheet. The bar shows the active record and the keys have the meaning explained in Figure 7.

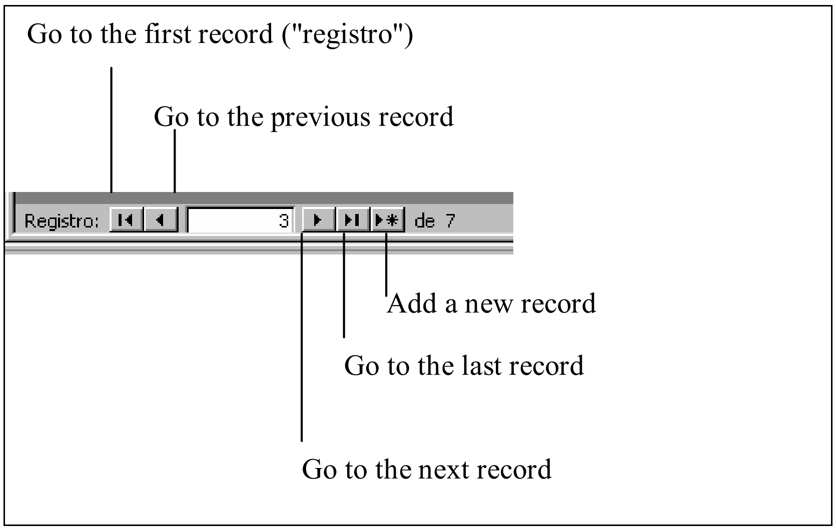


Figure 7: Record selectors to move to a different record.

2.2. The tool QUERIES

In what follows we will study the tool *Queries* which allows us to query the information which is stored in the database. The concept of query goes beyond a simple visualisation of the tuple in a relation, as we saw in Figure 5 with the option *Abrir* of the tool *tables*. A query consists of the retrieval of the database data that comply with some conditions specified by the user. In a relational database, the result of a query is also a relation, and can also be shown in a tabular form.

For example, to show a subset of attributes in a relation, such as the department and the telephone; on the other hand, if we want some information which is contained in more than one relation, such as the name of the department and the name of the courses imparted by the department, etc. In each case, we will need a specific query.

Choosing the tool *Queries* in the general window of the "Base de Datos", Access shows the existing queries, as can be seen in Figure 8.

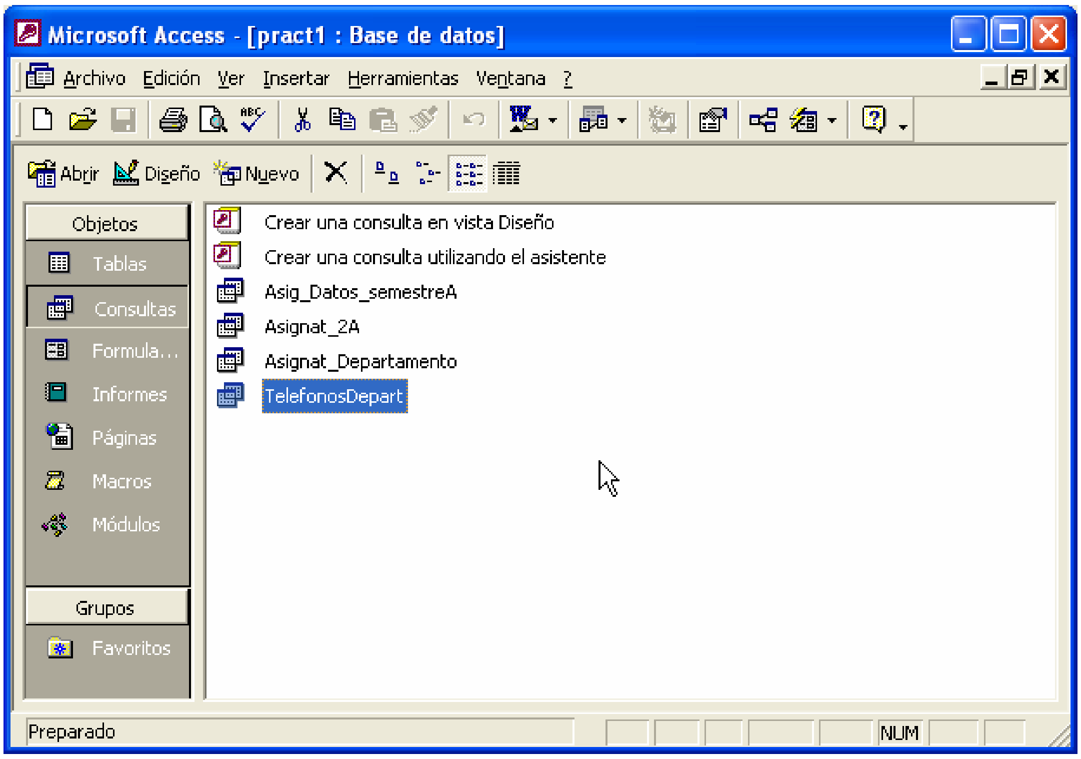
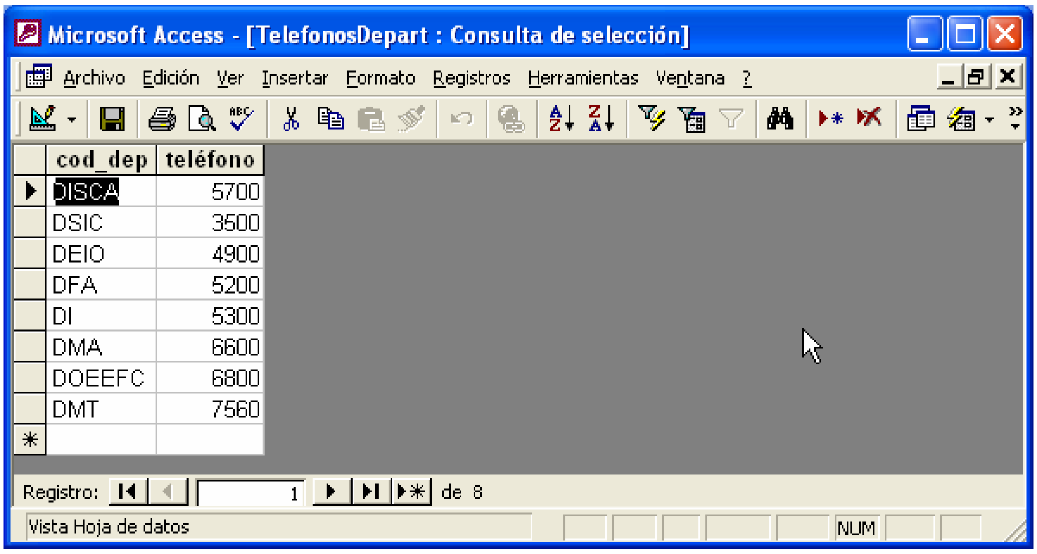

Microsoft Access - [pract1 : Base de datos]
Archivo Edición Ver Insertar Herramientas Ventana ?
Abrir Diseño Nuevo
Objetos
Tablas
Consultas
Formula...
Informes
Páginas
Macros
Módulos
Grupos
Favoritos
Crear una consulta en vista Diseño
Crear una consulta utilizando el asistente
Asig_Datos_semestreA
Asignat_2A
Asignat_Departamento
TelefonosDepart
Preparado
NUM

Figure 8: Window of the tool "Queries" (queries).

On the top left corner there are some buttons which correspond to the following operations:

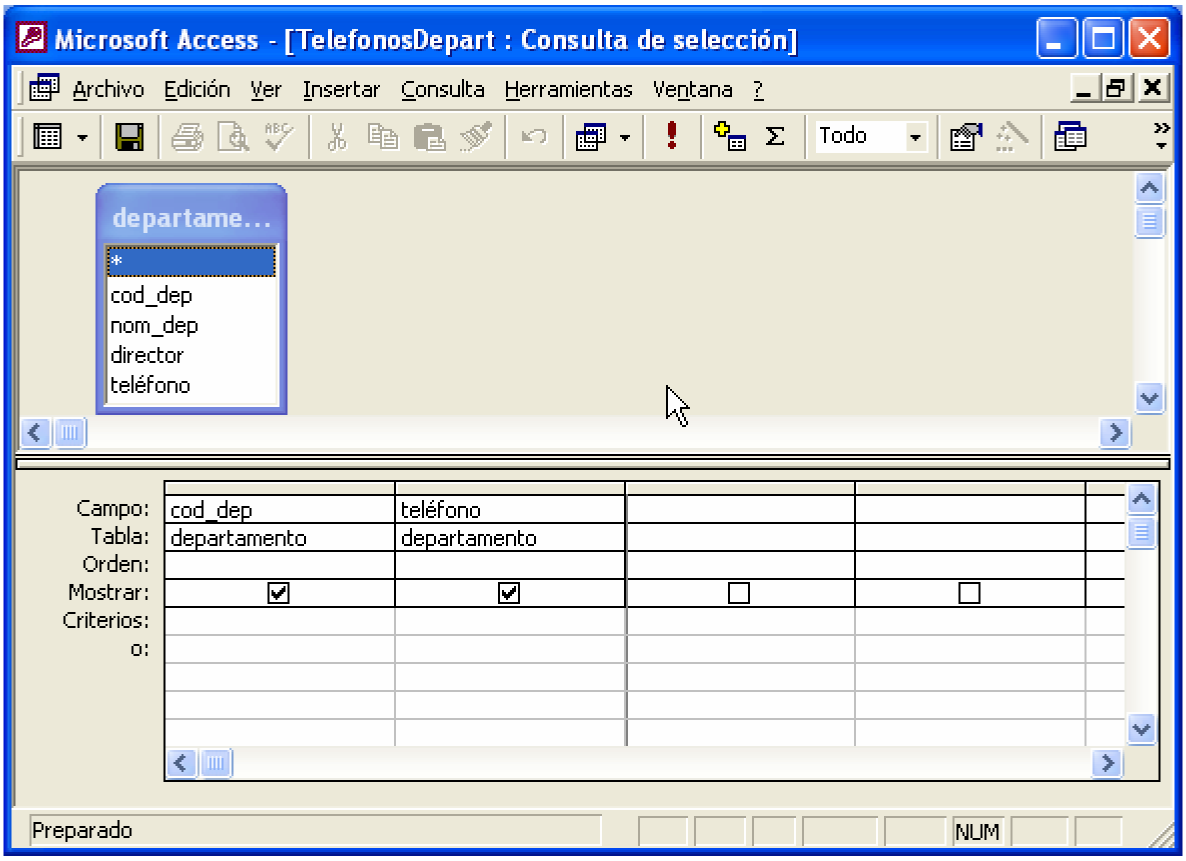
- **Abrir (open)**: visualises the data which are retrieved by the selected query. When we choose this option, the following window "Consulta" is open in the view mode "Hoja de datos" (Figure 9).


Microsoft Access - [TelefonosDepart : Consulta de selección]
Archivo Edición Ver Insertar Formato Registros Herramientas Ventana ?
Registro: 1 de 8
Vista Hoja de datos
NUM

cod_dep	teléfono
DISCA	5700
DSIC	3500
DEIO	4900
DFA	5200
DI	5300
DMA	6600
DOEEFC	6800
DMT	7560

Figure 9: Window "Consulta" in the view mode "Hoja de datos" (datasheet).

- **Diseño (design)**: visualises and allows the user to modify the definition of a selected query. When we choose this option, the window Consulta is open in the view mode "Diseño" (Figure 10).


Microsoft Access - [TelefonosDepart : Consulta de selección]
Archivo Edición Ver Insertar Consulta Herramientas Ventana ?
Todo
departame...
Preparado
NUM

departame...
- *
- cod_dep
- nom_dep
- director
- teléfono

Campo:	cod_dep	teléfono		
Tabla:	departamento	departamento		
Orden:				
Mostrar:	☑	☑	☐	☐
Criterios:				
o:				

Figure 10: Window "Consulta" in the view mode Diseño (design).

- **Nuevo (new)**: allows the user to define a new query.

2.2.1. Operations over existing queries: *Abrir* and *Diseño*.

The window Consulta in the mode " Hoja de datos" (Figure 9) is similar to the window Tabla in the same mode.

When we want to change the definition of a query we must select the query in the view mode "Diseño" (Figure 10). On the top of the window we can see a list of attributes for the relations used in the selected query.

On the bottom, we see a design grid in which we see the attributes which are used in the query, the relations where the attributes come from, the sorting criterion, the attributes that will be visualised, and whatever selection criterion which is established for the attributes.

2.2.2. Creating queries

The creation of a new query can be performed through the tool *Queries* by selecting the option *Nuevo* in the mode *Vista Diseño*. After this selection, we can see the dialog window "Mostrar Tabla" in Figure 11, which allows the user to aggregate relations that the user wants to include in the query.

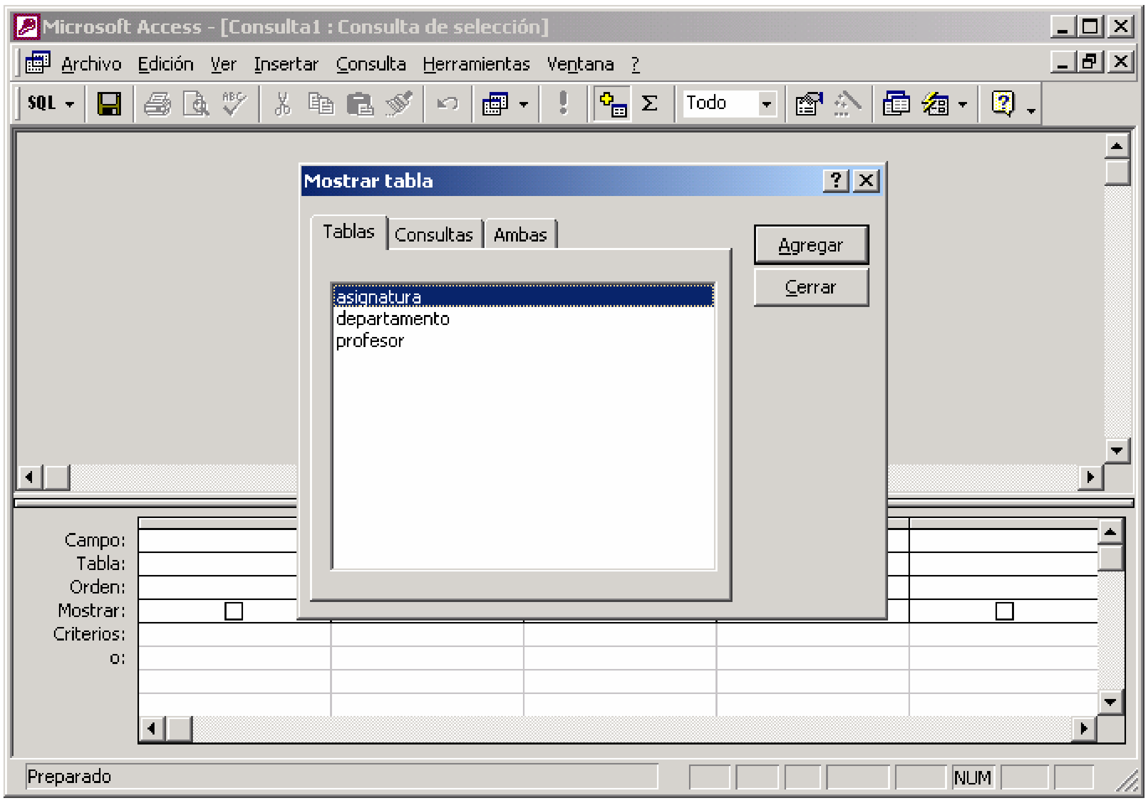


Figure 11: Selection of relations for the creation of a new query.

During the query design process, when we use the term *campo* (field) we refer to a place where it is possible to express conditions and that is associated with some attribute in the relation, with some calculated field based on several attributes or with a field of totals which uses one of the functions which are provided by the system. The sequence of steps to design a new query is as follows:

- Select the fields.
- Construct a condition which is associated with a query.
- Execute a query.
- Navigate through the queried information.

Below we present each of these steps in detail.

a) Select the fields

The first step in the construction of a query is to select the fields that we want to visualise or about which we want to express conditions. As we mentioned above, at the bottom of the window Diseño there is a design grid, where most of the query is defined. Each column in the grid represents a field over which the query works. The selection of the field is performed by placing on the row *Campo* of the grid and selecting the attribute from the pop-up list as shown in Figure 12. Another possibility consists of dragging the field from the upper part of the window to the row *Campo* in the grid. For the addition of a field you can drag the asterisk (*) to the design grid.

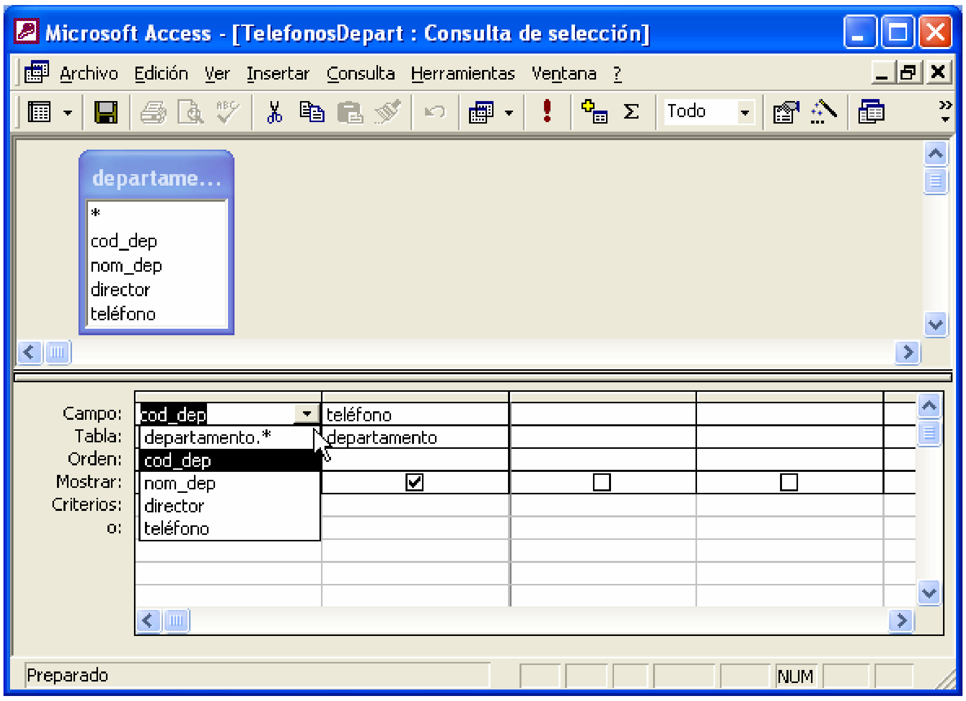


Figure 12: Field selection.

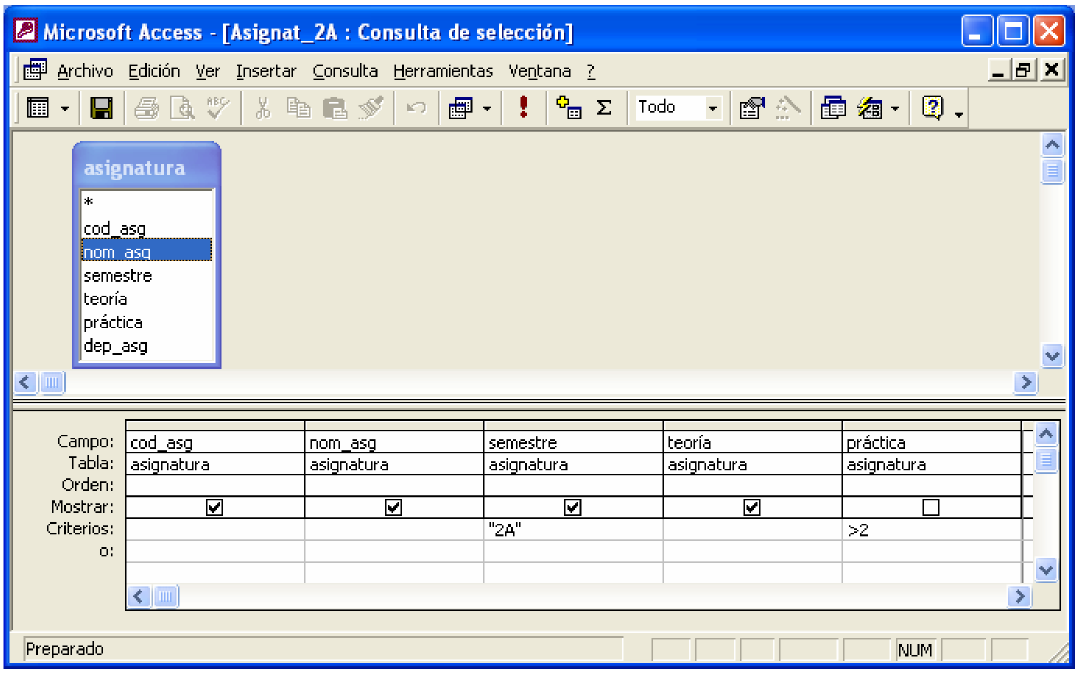
The fields which compose the output of a query inherit the properties which are defined for that field in the table. We can also define some other properties over the fields, such as the description, the format, the input mask or the title; this is done by placing the cursor over the column of the field on the design grid or by pressing the button "Propiedades" or the menu "Ver".

b) Build the condition which is associated with the query.

A condition in a query is a logical formula which is composed by a set of comparisons with the logical connectives: disjunction ("or") and conjunction ("and"). The condition which is associated with a query is constructed by using the rows "*Criterios*" and "*O*" of the grid which is at the bottom of the screen. Thus, in order to look for the rows that have a simple value for one attribute, we just write down the name of the value in the row *Criterios* at the column which corresponds to the field that we want to check. The comparison operator by default is "=" but we can use other operators: = ,<>, <, >, <=, >=.

In the same place where we introduce the comparisons we can indicate if the conditions are combined with the conjunction (if the comparisons are in the same criterion row) or a disjunction (if the comparisons are in different rows). Additionally, it is possible to explicitly introduce connectives (through preserved words "OR" and "AND" or, in Spanish, "O" and "Y") to combine comparisons.

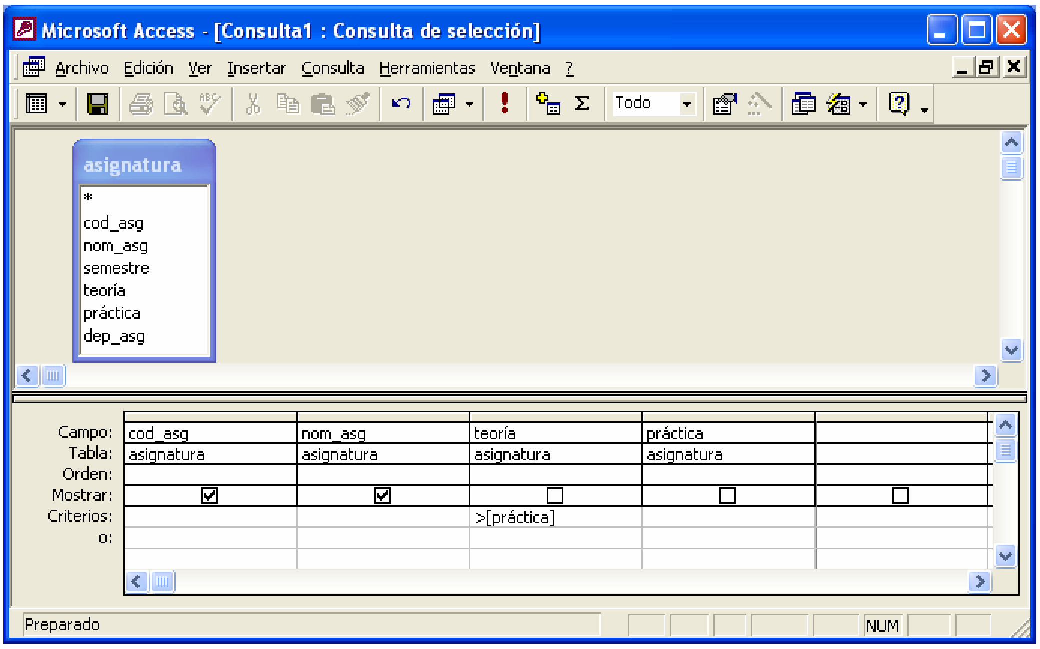
Figure 13 illustrates a condition over the query: "Obtain the code, name, term and n of credits of the courses taught in term 2A with a number of credits in the laboratory ("prácticas") greater than 2". As can be observed, in the column which corresponds to the attribute *semestre*, we can just see the value "2A" (an abbreviated form of the comparison "=2A") and on the column which corresponds to the attribute *prácticas,* we see the comparison ">2". In this case, the combination of the two comparisons is performed in an implicit way depending on their location in the grid; it is a conjunction since both comparisons have been introduced in the same row.


Microsoft Access - [Asignat_2A : Consulta de selección]
Archivo Edición Ver Insertar Consulta Herramientas Ventana ?
Todo
asignatura
*
cod_asg
nom_asg
semestre
teoría
práctica
dep_asg
Preparado
NUM

Campo:	cod_asg	nom_asg	semestre	teoría	práctica
Tabla:	asignatura	asignatura	asignatura	asignatura	asignatura
Orden:					
Mostrar:	☑	☑	☑	☑	☐
Criterios:			"2A"		>2
o:					

Figure 13: Example of the construction of a condition.

We can see constants in the comparison, as seen in Figure 13, but we can see references to other attributes; in Figure 14 we see the query "Obtain the code and name of the courses with more credits for lectures ("teoría") than for laboratory ("prácticas"). In this query we can see that in order to indicate that we are using an attribute in the comparison (and not a value), we must use squared brackets. Additionally, the reference to an attribute in a comparison must be preceded by the name of the relation to which it belongs (e.g. [asignatura].[prácticas]). In this way, the ambiguity which appears when two relations have two attributes with the same name is solved.


Microsoft Access - [Consulta1 : Consulta de selección]
Archivo Edición Ver Insertar Consulta Herramientas Ventana ?
Todo
asignatura
*
cod_asg
nom_asg
semestre
teoría
práctica
dep_asg
Preparado
NUM

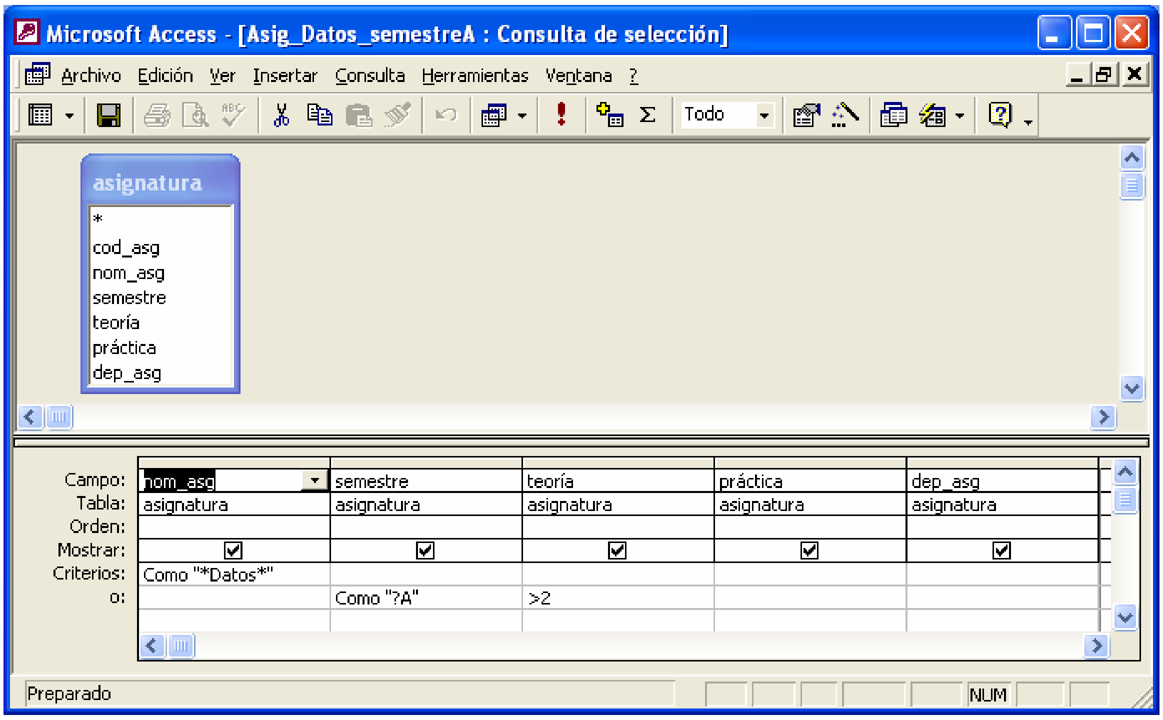
Campo:	cod_asg	nom_asg	teoría	práctica	
Tabla:	asignatura	asignatura	asignatura	asignatura	
Orden:					
Mostrar:	☑	☑	☐	☐	☐
Criterios:			>[práctica]		
o:					

Figure 14: Example for the construction of a condition with references to other attributes.

Next we present the most common ways of constructing a condition:

1. Conditions over the same field: we can use several comparisons and combine them with logical connectives ("AND" and "OR").
2. Conditions over several fields: in this case it is not possible to express the logical connectives to express the condition. These are expressed, in an implicit way, by the position they have in the grid. Conjunction will be used when the comparisons are in the same row *Criterios* and disjunction when these are introduced in different rows *Criterios* and *O*.
3. Special predicates:
   - *ENTRE* (*between*): to specify a range of values. The clause "*ENTRE 10 Y 20*" is the same as "*>=10 Y <=20*" and it means "*>=10 AND <=20*".
   - *EN* (in): to specify a list of values, where one of them can match the field value. The clause "*EN (“1A”; “2A”; “3A”)* " is the same as "*“1A” O “2 A” O “3 A”* ", which means "*“1A” OR “2 A” OR “3 A”* ".
   - *COMO* (like): to use patterns. We can use special characters to express patterns. The character *?* is useful to express the pattern which corresponds to any string of one character. The character * is useful to express a pattern which corresponds to expressing a string of whatever length (including the empty string). Note that both patterns ("?" and "*") only work if we use the comparison symbol "COMO". They do not work with the comparison symbol "=".
   - Access also provides some functions to treat date and time, as well as other arithmetic operators.

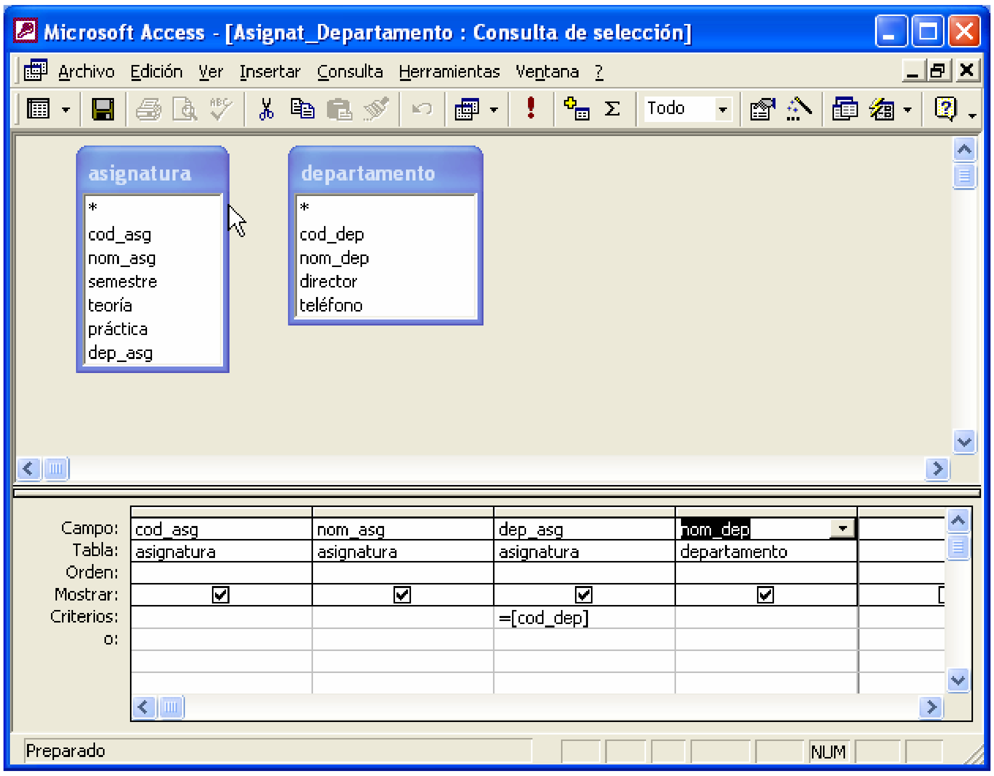
With these basics we are able to express conditions of different complexity. Figure 15 shows the following query: “Obtain the name, term, n of credits and code of the department with courses which either contain the word ‘Datos’ or that their term is the first (i.e. ending with ‘A’) and their n of lecture credits is greater than 2”.



Campo:	nom_asg	semestre	teoría	práctica	dep_asg
Tabla:	asignatura	asignatura	asignatura	asignatura	asignatura
Orden:					
Mostrar:	✓	✓	✓	✓	✓
Criterios:	Como "*Datos*"				
o:		Como "?A"	>2		

Figure 15: Example of a query design using patterns.

In case the query requires several relations, we normally must include some comparisons which ensure that the rows of the involved relations are related in an appropriate way. Thus, in Figure 16, we show the query “Obtain the code and name of all the courses, including the code and name of their department”. As can be seen, the criteria for the attribute *dep_asg* in the relation ASIGNATURA include the comparison "=.[cod_dep]" which ensures that each course is related with only one department (its corresponding department).


Microsoft Access - [Asignat_Departamento : Consulta de selección]
Archivo Edición Ver Insertar Consulta Herramientas Ventana ?
Todo
Preparado
NUM

asignatura
*
cod_asg
nom_asg
semestre
teoría
práctica
dep_asg

departamento
*
cod_dep
nom_dep
director
teléfono

Campo:	cod_asg	nom_asg	dep_asg	nom_dep
Tabla:	asignatura	asignatura	asignatura	departamento
Orden:				
Mostrar:	☑	☑	☑	☑
Criterios:			=[cod_dep]	
o:				

Figure 16: Example of a query design using several tables.

c) Executing a query

The execution of a selected query is performed by simply pressing on the operation "*Abrir*", or if we are in the mode "Diseño", we can press the button "!" of the toolbar.

d) Navigating with the retrieved data

When executing a query, the system shows the tuples of the relation which are consistent with the specified condition. The system presents this information in the form of a table.

If the answer to the execution of a query requires the results to be sorted, then we can use the row *Orden* in the grid to use an ascending or descending ("Ascendente" or "Descendente") order from a pop-up list of attributes. We can also sort through several fields. Access applies the sorting criteria from left to right, according to their appearance in the design grid.

3. A complete database example

In the first section of this document, we presented a simple database in which we included information about the departments with teaching at the school EI, the lecturers who are inscribed in each department and the courses of which they are in charge. We saw that the most convenient representation using a relational database was to define three relations *DEPARTAMENTO*, *PROFESOR* and *ASIGNATURA*. The schema of these relations was as follows:

DEPARTAMENTO(cod_dep: string(5), nom_dep: string(40), director: string(30), teléfono: integer)

ASIGNATURA(cod_asg: string(3), nom_asg: string(40), semestre: string(2), teoría: real, prácticas: real, dep_asg: string(5))

PROFESOR(cod_pro: string(3), nomprof: string(40), extensión: integer, dep_pro: string(5))

We now want to add the information relative to the lecturers who teach at the school EI. The description of this information is as follows:

From all the lecturers who teach at the school, we also want to know the courses taught by each lecturer and the number of lecture and theory groups she/he is in charged of.

Obviously, if we know which courses are given by each lecturer, we will also know (and we want to know this as well), which lecturers give each course.

Hence, we need to find a way of representing the information about the courses that are imparted by each lecturer (or symmetrically, the lecturers who teach each course). It might seem feasible to represent this information in a similar way as the department to which a lecturer belongs. I.e., including in the relation "PROFESOR" the codes of the courses she/he teaches. The problem is that, a priori, we do not know how many courses a lecturer can teach: one, two or many... Consequently, it is impossible to know how many values (and hence attributes) per course must be added to the relation *PROFESOR*. The same rationale can be applied if we try to represent the information in a symmetric way in the relation *ASIGNATURA*. In other words, it is impossible to know a priori how many lecturers can be ascribed to a course, so it is impossible to know how many attributes we should add to the relation *ASIGNATURA*.

The solution to this problem, however, is fairly simple. We can define a new relation in which each tuple includes the information which corresponds to the fact that a lecturer teaches a course, including the lectures and lab session she/he is in charge of. This relation, which is known as *DOCENCIA*, can be seen as follows:

DOCENCIA

asg	**pro**	**gteo**	**gpra**
BDA	JCC	0	4
MAD	RFC	1	2
FCO	DGT	2	2
AD1	MAF	1	1
INT	CPG	1	0
EC2	JBD	2	0
BDA	MCG	1	3
AD1	JCC	0	1
FCO	JBD	2	2
AD1	MCG	1	1

Thus, we can see that each tuple in the relation *DOCENCIA* represents that a lecturer teaches a course, including the number of lecture and lab groups she/he is in charge of. The schema of the relation *DOCENCIA* is as follows:

DOCENCIA(asg: string(3), pro: string(3), gteo: integer, gpra: integer)

The attributes *asg* and *pro*, both together, are useful to identify in an unequivocal way the tuples in the relation *DOCENCIA.* Additionally, these attributes are useful to associate the relation *DOCENCIA* with the relations *ASIGNATURA* and *PROFESOR*, respectively. Hence, in each tuple in *DOCENCIA* the value of the attribute *asg* should be equal to the value in the attribute *cod_asg* in some tuple in the relation *ASIGNATURA*. The same constraint applies to the attribute *pro* regarding the relation *PROFESOR*. The new database can be accessed through the name of **pract12.mdb.**

4. Exercises

We propose some exercises to be solved using Access. We show between brackets the number of rows that the proposed queries should return. This is an indication of whether the query has been solved right or wrongly. Nonetheless, it is important to highlight that this is just an indication: a query can be wrong and the number of rows might match. However, if the number of rows does not match, the query is clearly wrong.

4.1. Queries over a single table

1. Get the code, the name and the number of lecture and lab credits of the courses with more than 2 credits in lectures or labs ("teoría" or "prácticas"). (26 rows)
2. Get the code, the term and the name of the courses which are offered in the first term ("A") such that the department responsible for their teaching has code DSIC. (7 rows).
3. Get the code and the name of the courses such that their name starts with "E" or with "M". (7 rows).
4. Get the code, the name and the telephone of the departments whose telephone number is greater than 5000. (6 rows).

4.2. Queries over several tables

1. Get the code and the name of the courses, as well as the code and the name of the department to which they are ascribed. Show the result sorted by term. (27 rows)
2. Get the code and the name of each course with code and the name of the department to which they are ascribed only if the course is offered in the first or second year, and the name of the department contains the word "*Aplicada*". (4 rows)
3. Get, for every department including the character "C" in any position of its code, the code, the name of the department and the name of the courses which are offered in the second year. Why are some departments missing in the result?
4. Get, for each lecturer, her/his code, her/his name and the courses she/he teaches, indicating for each of them their code, their name and the lecture and lab groups she/he is in charge of. Why are some lecturers missing in the results? Get the result alphabetically sorted by the name of the lecturer. (29 rows)

5. Modify the previous query such that we only show the lecturers of the department DSIC. (13 rows)
6. Get, for each department in the database, the following information: code and name of the department, lecturers ascribed to the department, indicating the code and the name of the lecturer as well as the courses she/he teaches, showing the code and the name of the courses. Get the information sorted by the code of the department. (29 rows)
7. Get the code and name of those lecturers who teach a course such that the department which is responsible for the teaching of that course is not the same as the lecturer's department. Include in the query the code of their respective department codes (lecturer's and course's). (3 rows)