

# **Databases**

**(Bases de Datos, BDA)**

## **Exercise book 1: Introduction to Relational Databases.**

The logo for the Department of Systems Informatics and Computing (DSIC) is written in a stylized, cursive font. The letters 'D', 'S', and 'I' are connected, and the 'C' is a simple curve. The 'S' has a small circle above it, and the 'I' has a small circle above it.

*Departamento de Sistemas Informáticos y Computación.*

2008/2009

**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 a University School. 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 degrees 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<sup>1</sup>:

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 *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.

<sup>1</sup> 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 screenshots copied.

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:

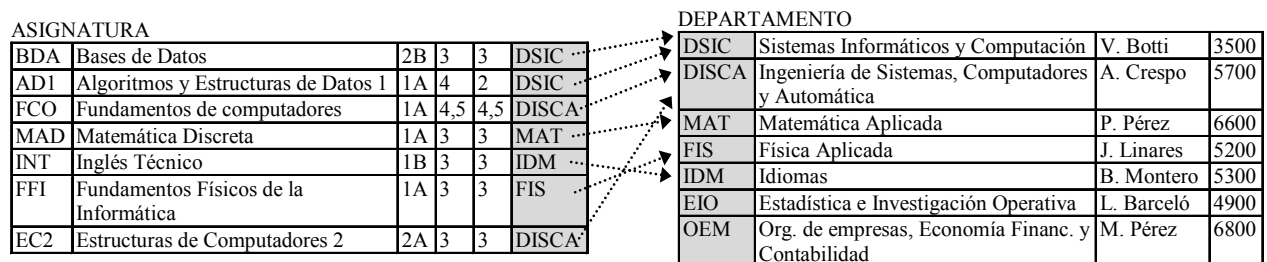


Figure 1: Graphical representation of the function of the attribute *dep\_asg* in the relation *ASIGNATURA*

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:

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.



Figure 2: Options when opening MsAccess.

When Access is initiated, we can see something similar to what is shown<sup>2</sup> in Figure 2, which allows to open (**Abrir**) an existing database or to create a new database. We will choose database<sup>3</sup> **pract12**, from where we go to a start-up menú once the database is open (Figure 3).

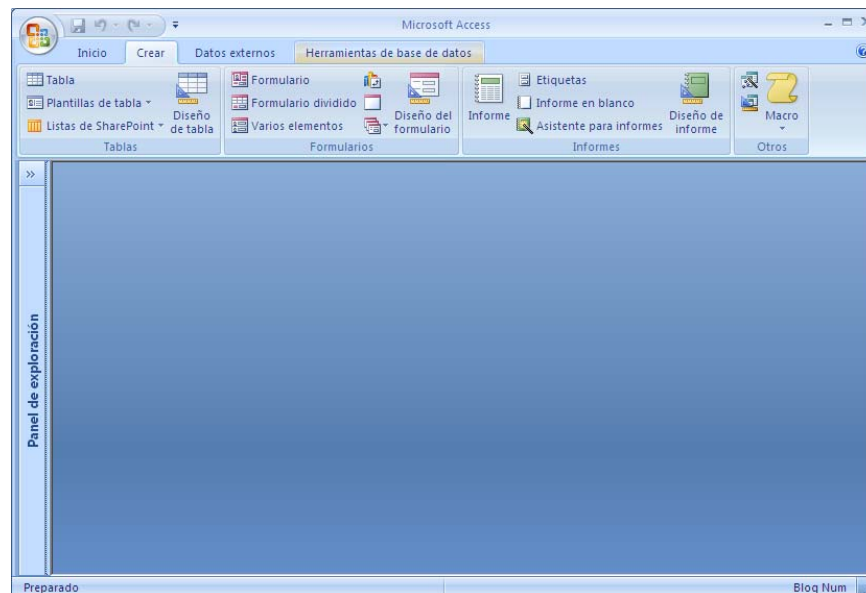


Figure 3: Start menu after opening the database..

<sup>2</sup> The figures are from Access XP in Spanish. Note that these figures might be slightly different to other versions and languages of Microsoft Office different from 2007. There is an alternative version of this document, for Access XP.

<sup>3</sup> The database includes some other extra tables, which will be explained later.

On the second folder (“*Crear*”) we find 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.
- **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.
- **Otros (others):** this option allows query definition. 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<sup>4</sup>. Every query can be established through a graphical interface or directly writing down the query in a specific language which is provided by the system (*Structured Query Language, SQL*).

As indicated above, the objective of this practical session will be to use the tools TABLES and the queries in OTROS.

## 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 we can find the screen which corresponds to the “*Panel de Exploración*”, where all the objects (pressing the arrow) can be chosen. If the “*Panel de Exploración*” is closed (Figure 3) you can click to open it. Once the option “*Tablas*” is chosen, the list of tables that are available in the database **pract12** (Figure 5) are shown.

---

<sup>4</sup> 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.

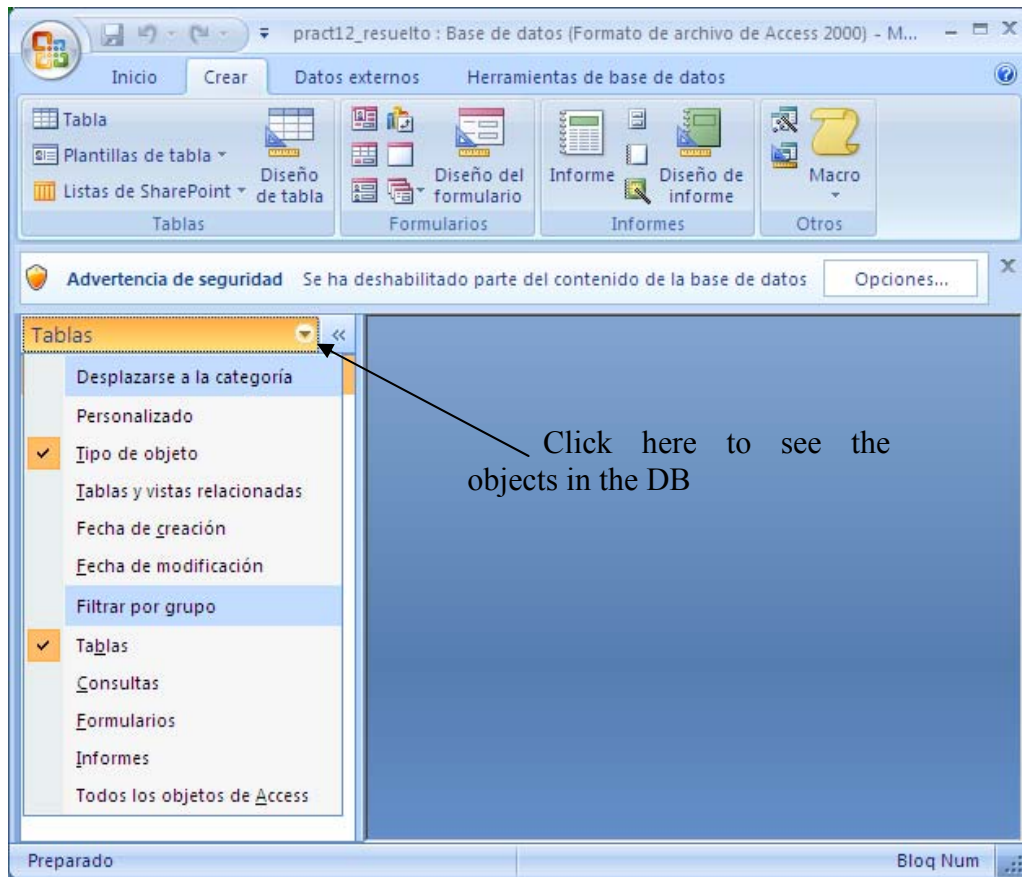


Figure 4: Options in the Tablas tool.

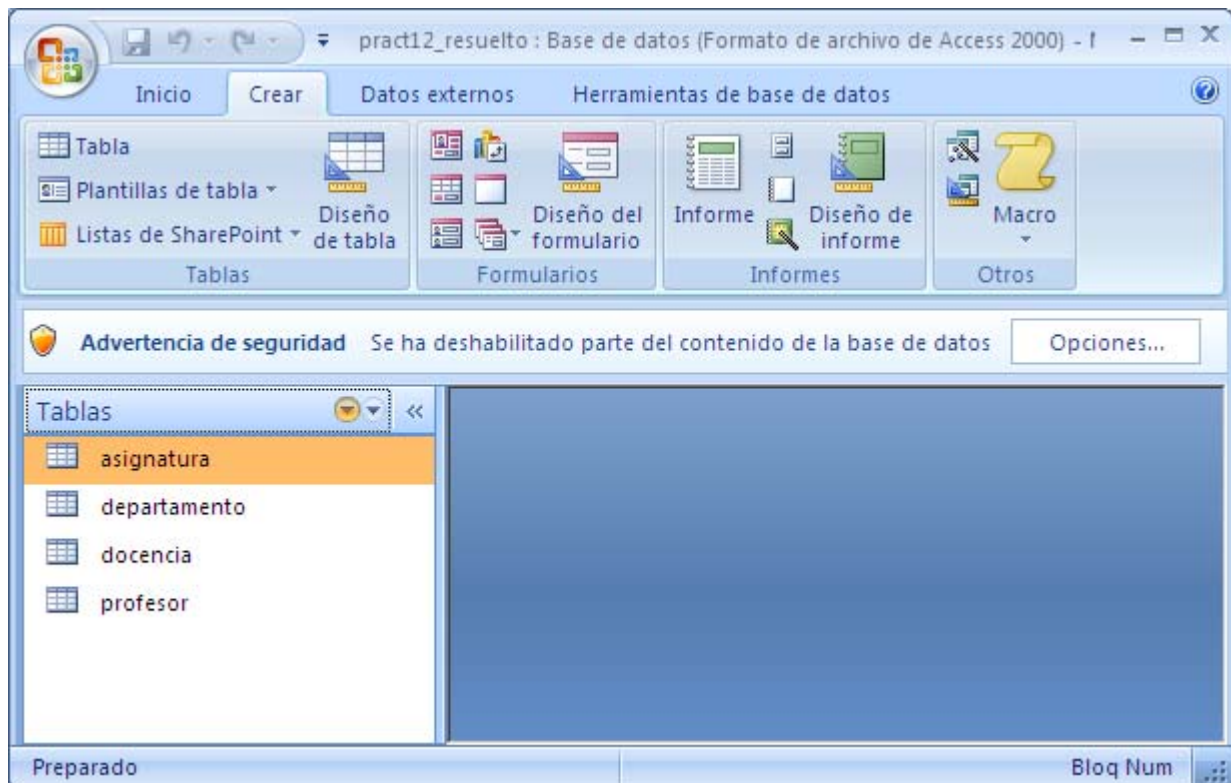


Figure 5: Tables in the database.

Choosing one of the tables in Access shows all the tuples (rows) which are stored in the table. Figure 6 shows table *Departamento*.

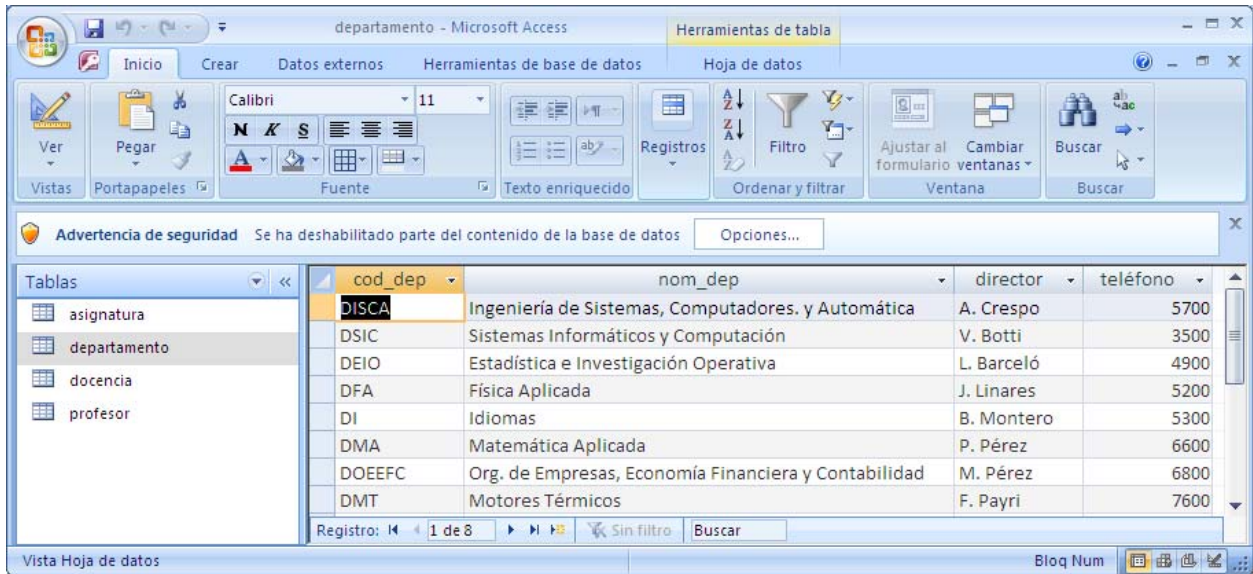


Figure 6: Table "departamento" in the view mode "Hoja de datos" (datasheet).

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

In order to examine the definition of a table, we need to choose the option "Vista Diseño" in the menu "Ver". 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 7 shows the screen which is displayed when we choose this option over the relation *DEPARTAMENTO*.

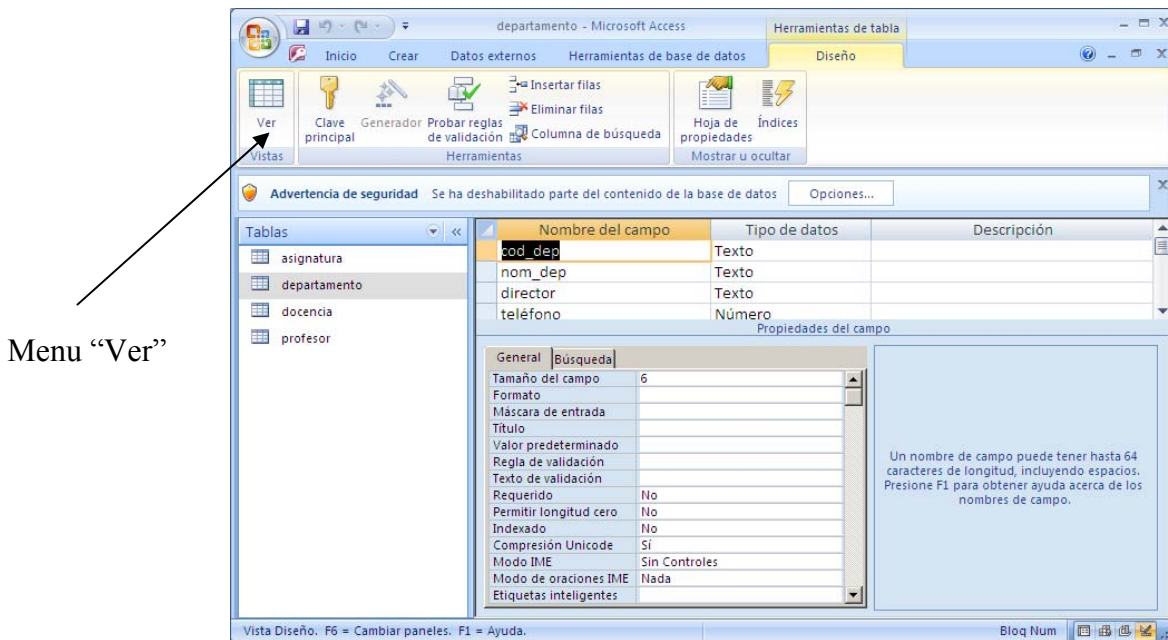


Figure 7: Table "departamento" in the view mode "Diseño" (design)

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.

### 2.1.1. 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 6. 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 8.

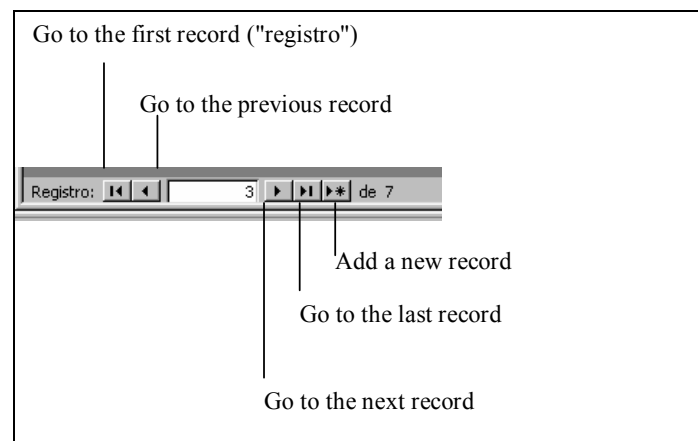


Figure 8: Record selectors to move to a different record.

## 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 6 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 9.

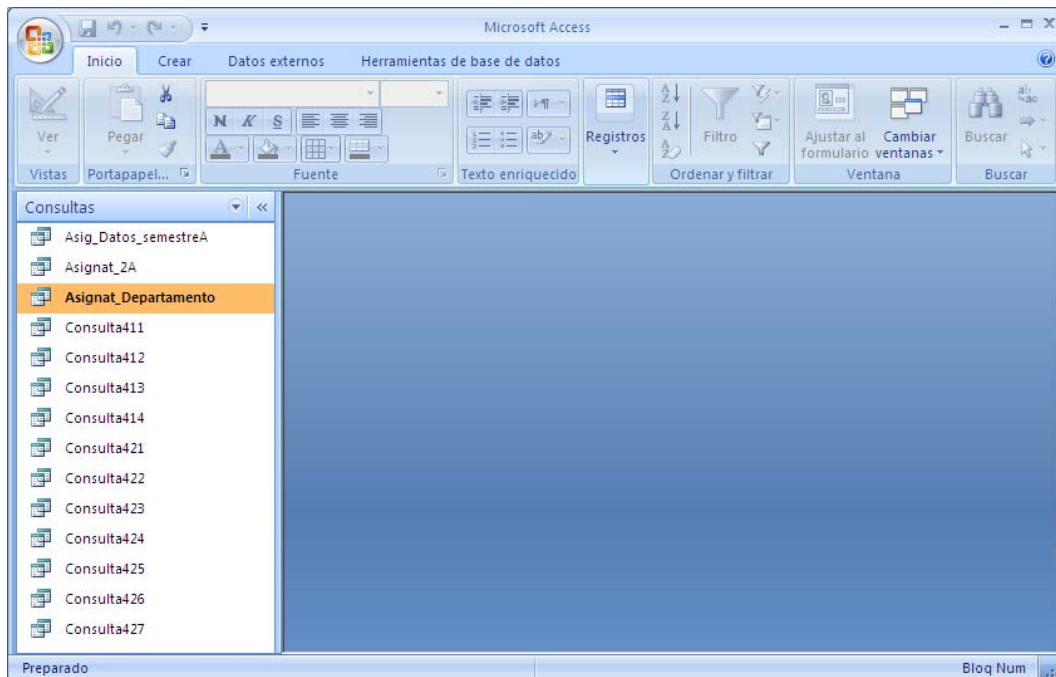


Figure 9: Window of the tool "Consultas" (queries).

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

The window *Consulta* in the mode "Hoja de datos" (Figure 10) is similar to the window *Tabla* in the same mode.

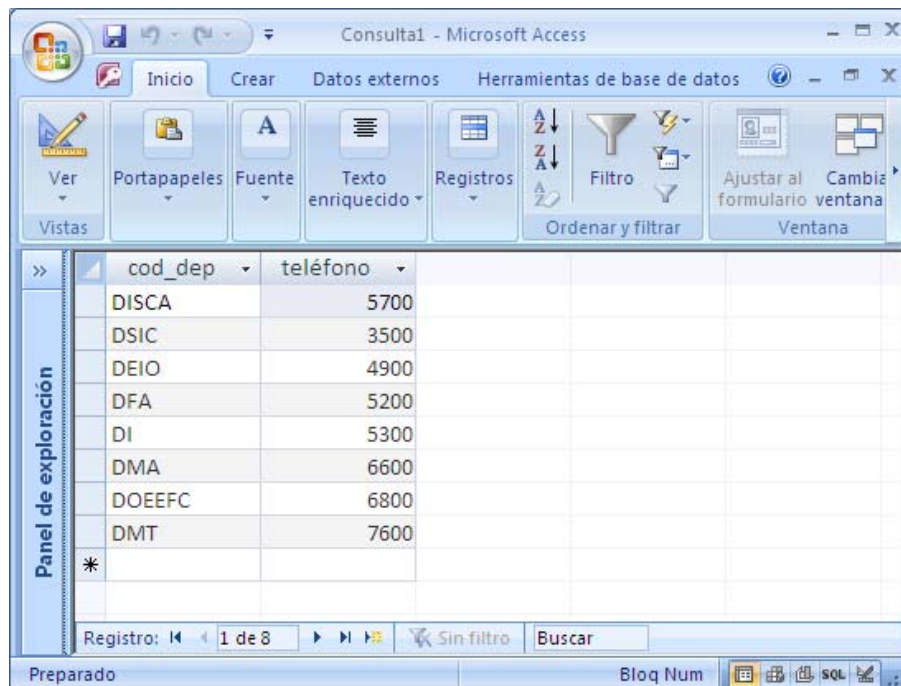


Figure 10: Window "Consulta" in the view mode "Hoja de datos" (datasheet).

When we want to change the definition of a query we must select the query in the view mode "Diseño" from the menu "Ver" (Figure 11). On the top of the window we can see a list of attributes for the relations used in the selected query.

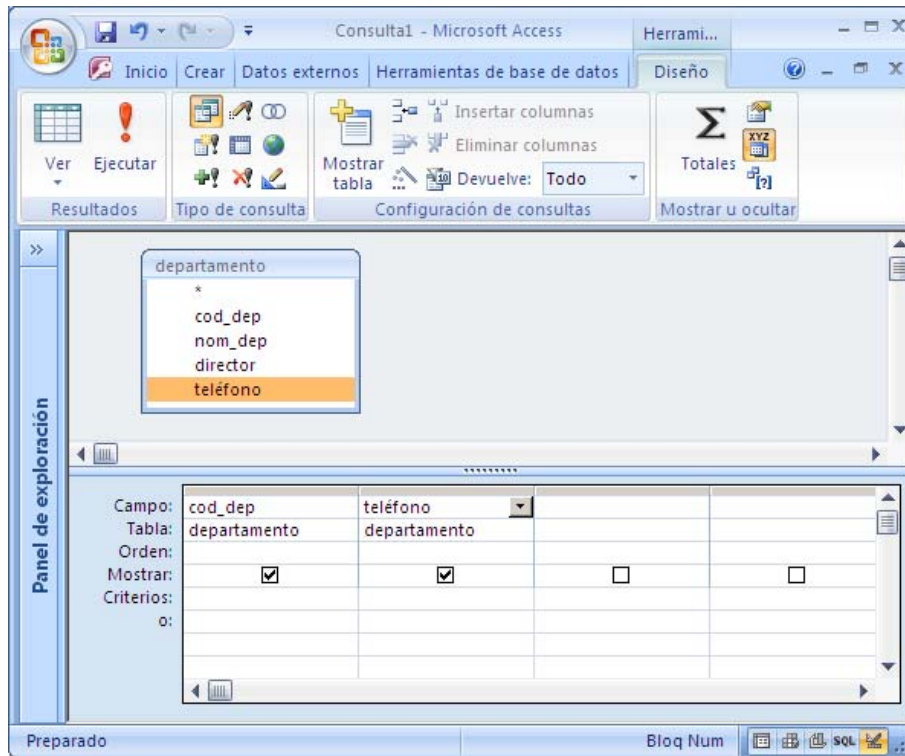
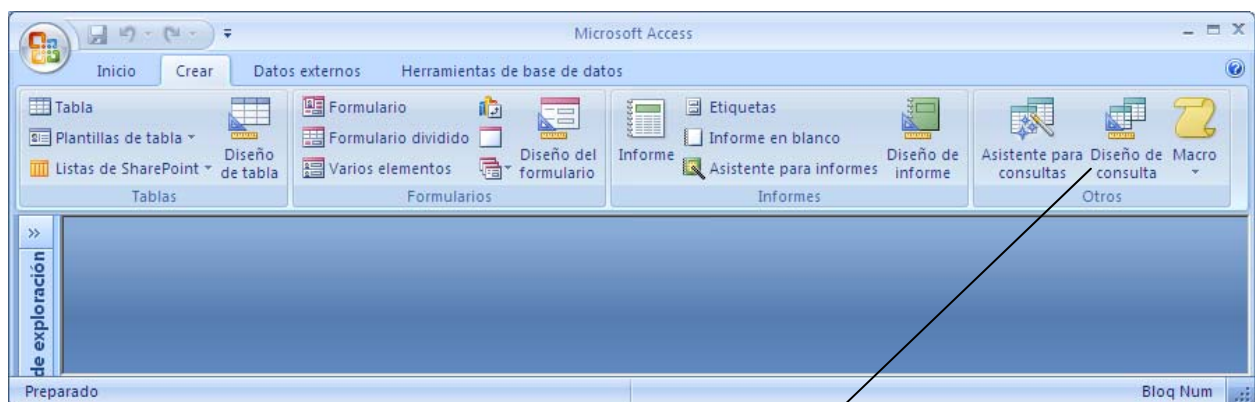


Figure 11: Window "Consulta" in the view mode Diseño (design).

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.

### 2.2.1. 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* from the menu *Crear* (Figure 12). After this selection, we can see the dialog window "Mostrar Tabla" in Figure 13, which allows the user to aggregate relations that the user wants to include in the query.



Creating a new query

Figure 12: Creating a new query.

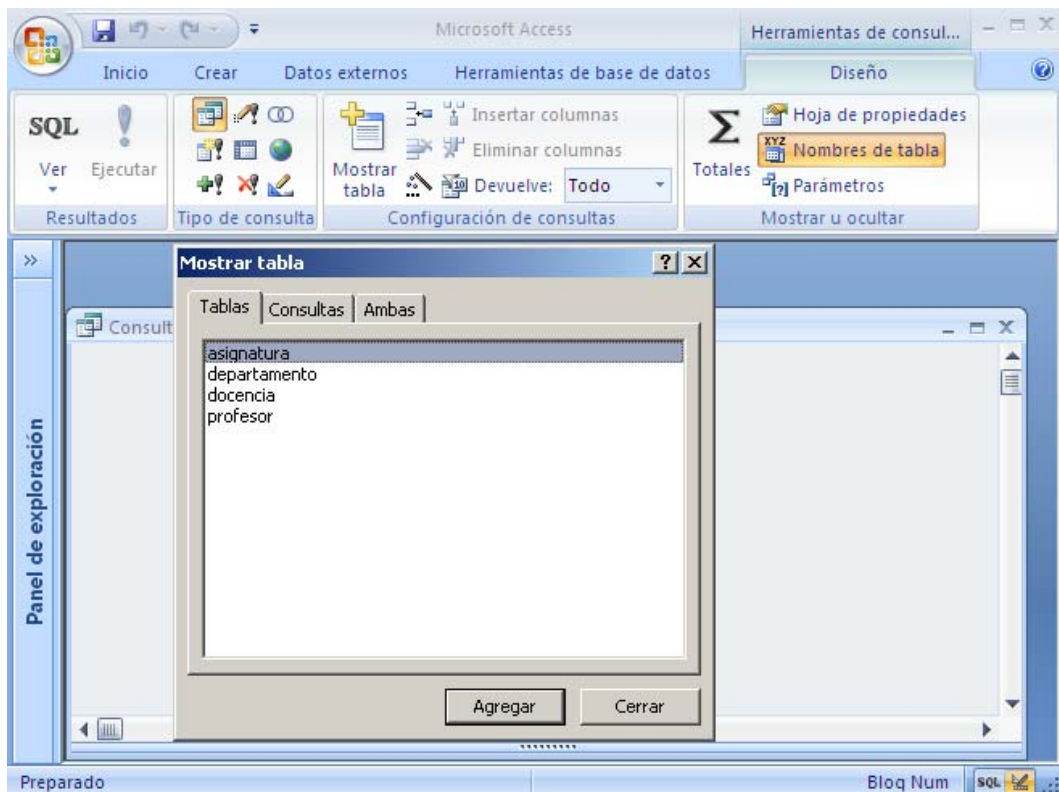


Figure 13: Selection of relations for the creation of a new query.

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 14. 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.

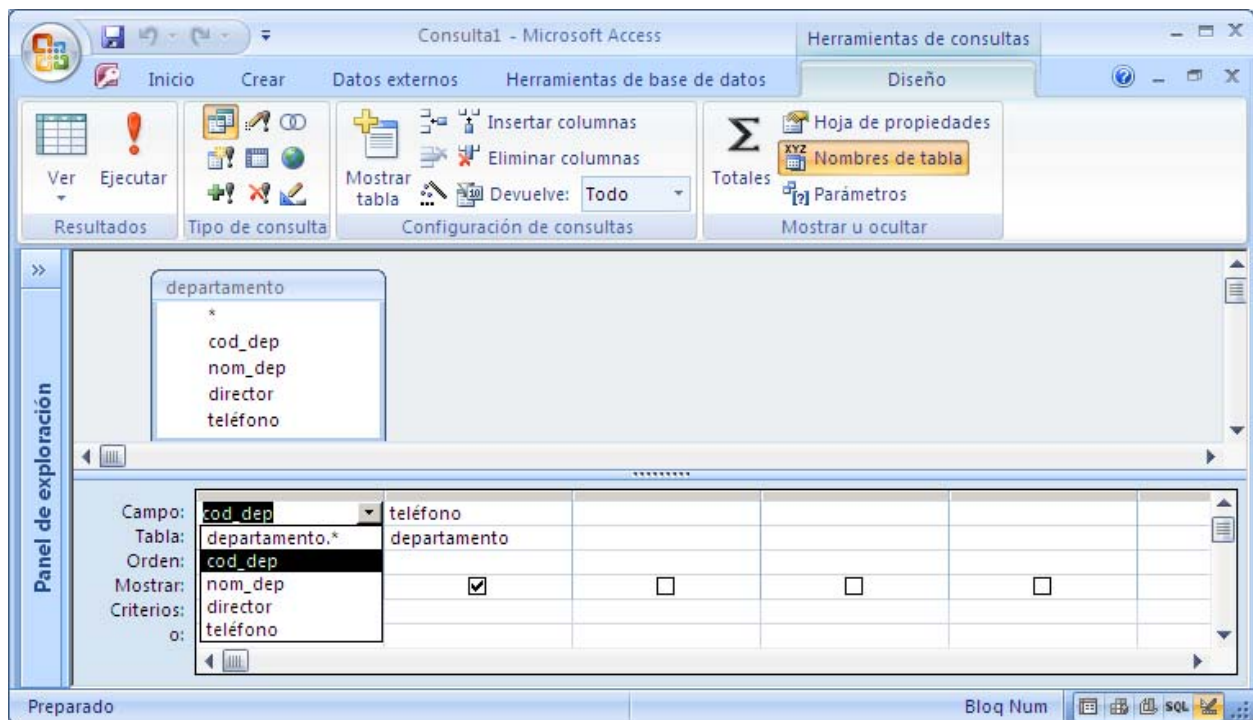


Figure 14: Field selection.

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 15 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.

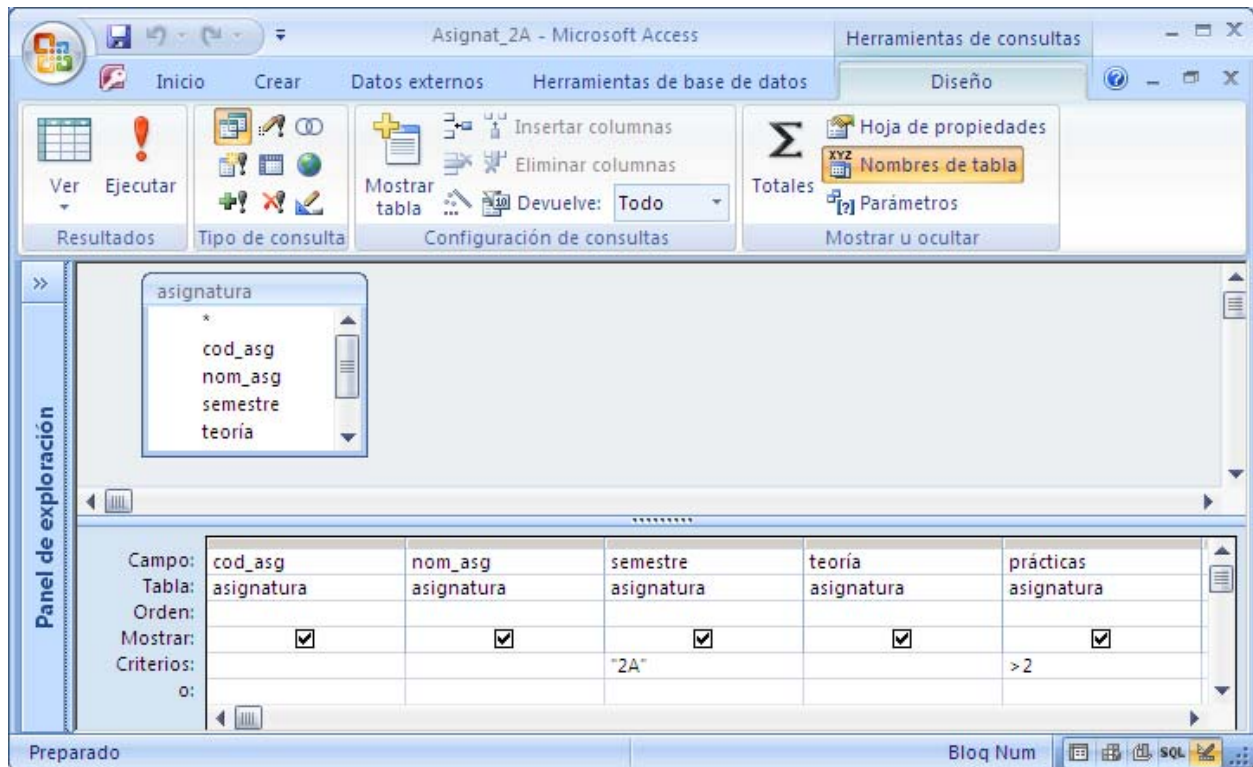


Figure 15: Example of the construction of a condition.

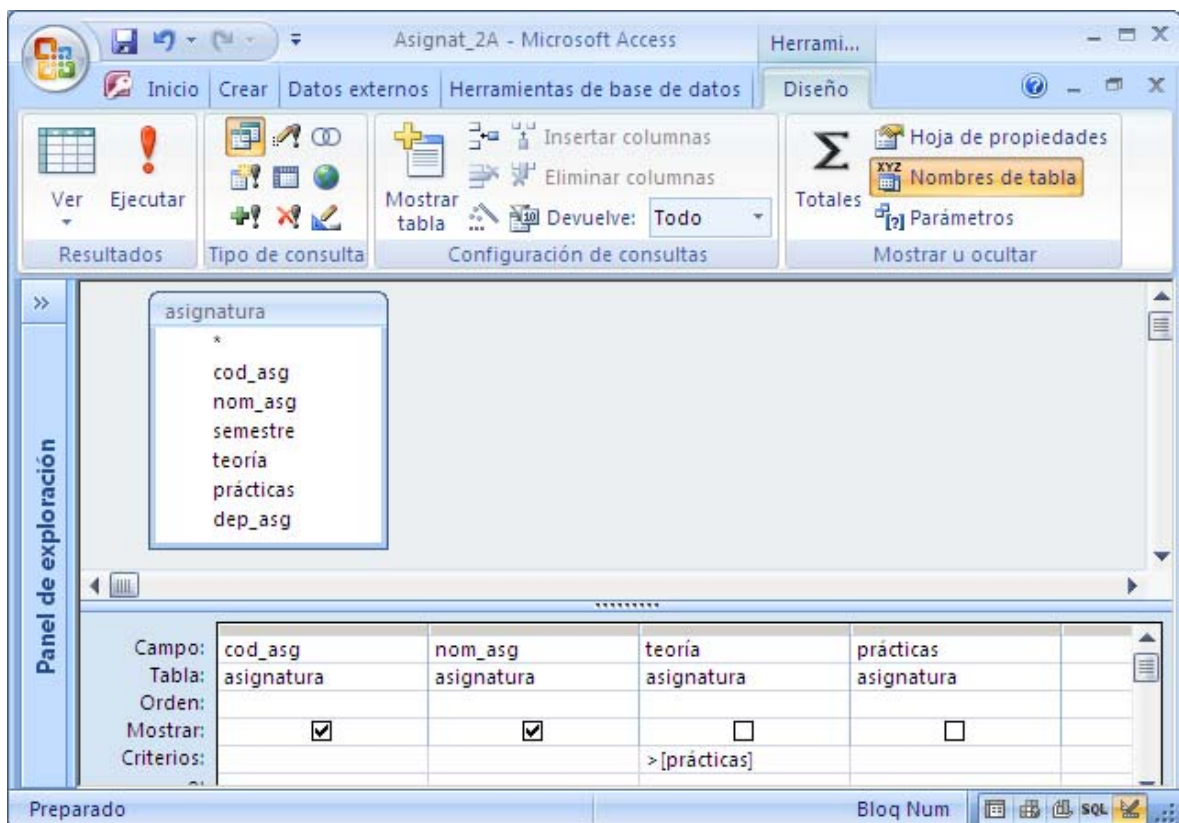


Figure 16: Example for the construction of a condition with references to other attributes.

We can see constants in the comparison, as seen in Figure 15, but we can see references to other attributes; in Figure 16 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.

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 *Criteria* and disjunction when these are introduced in different rows *Criteria* and *O*.
3. Special predicates:
  - *ENTRE* (between): to specify a range of values. The clause "*ENTRE 10 Y 20*" is the same as " $\geq 10$  Y  $\leq 20$ " and it means " $\geq 10$  AND  $\leq 20$ ".
  - *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 17 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”.

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 18, 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).

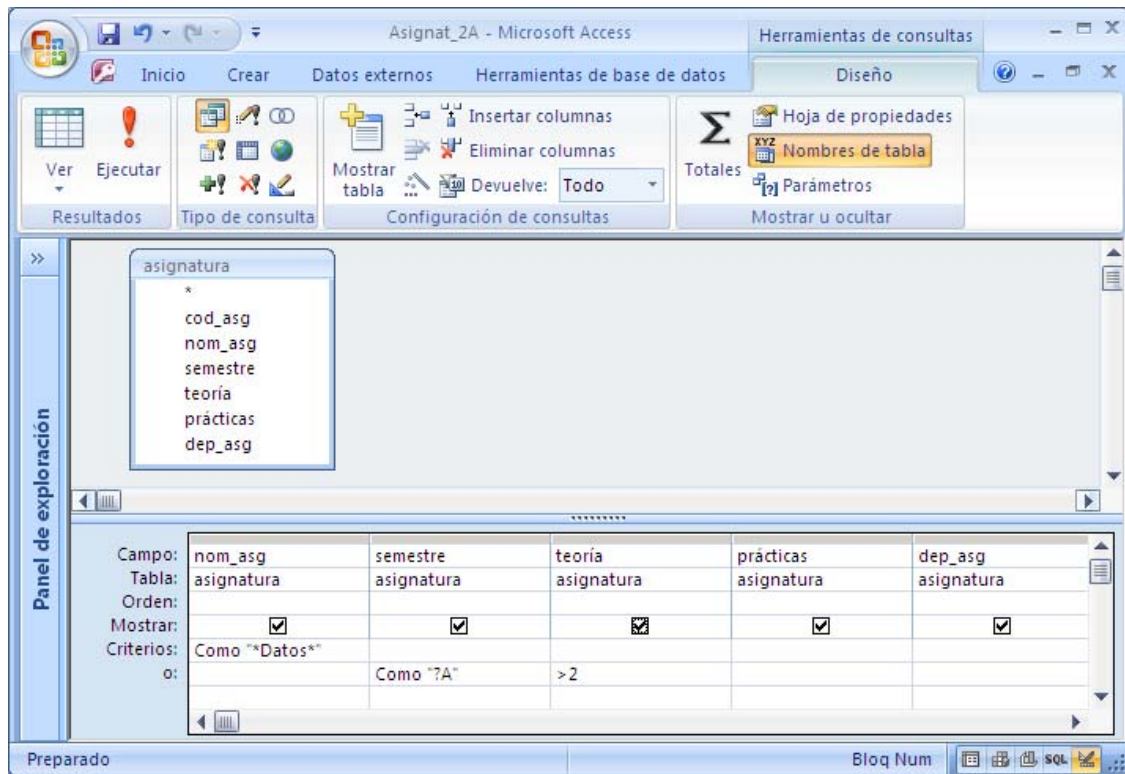


Figure 17: Example of a query design using patterns

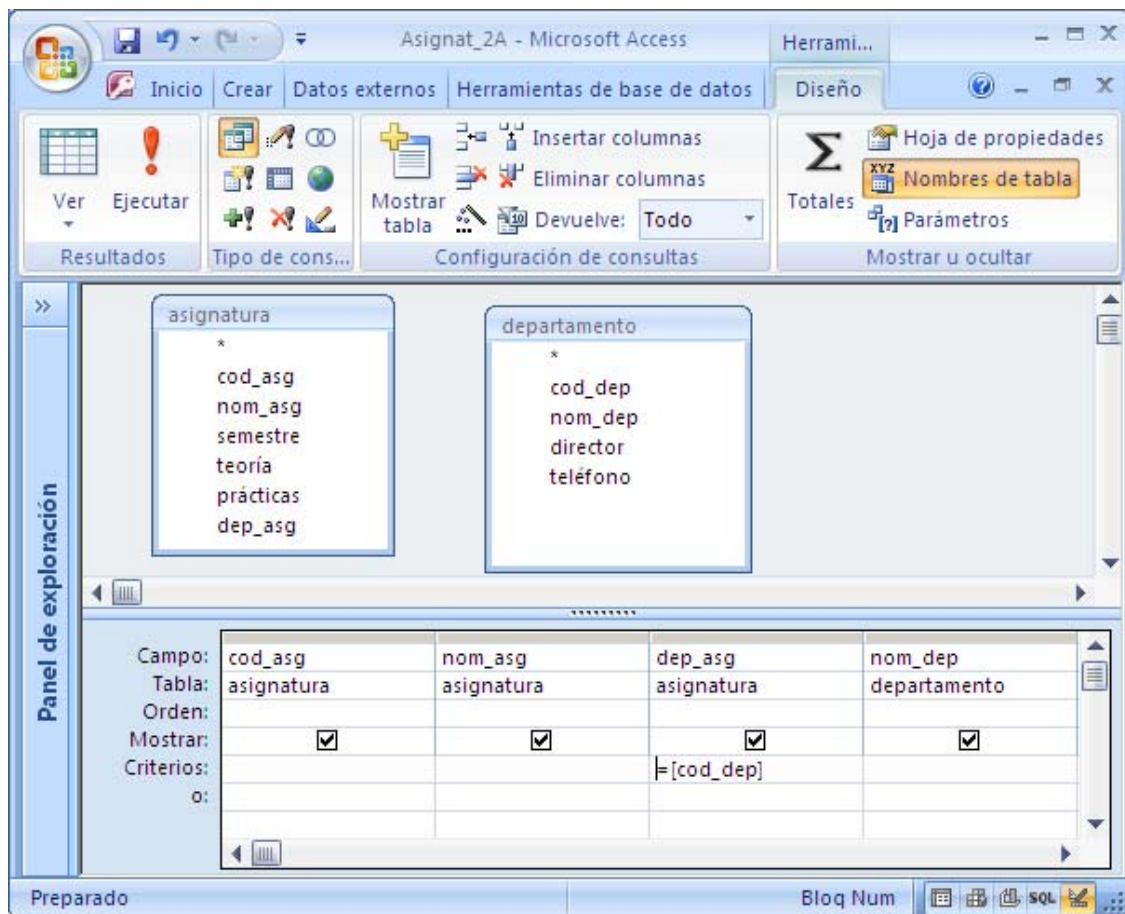


Figure 18: Example of a query design using several tables..

### 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.

The designed query can be seen in SQL clicking on the icon at the right-top corner, as shown in Figure 19.

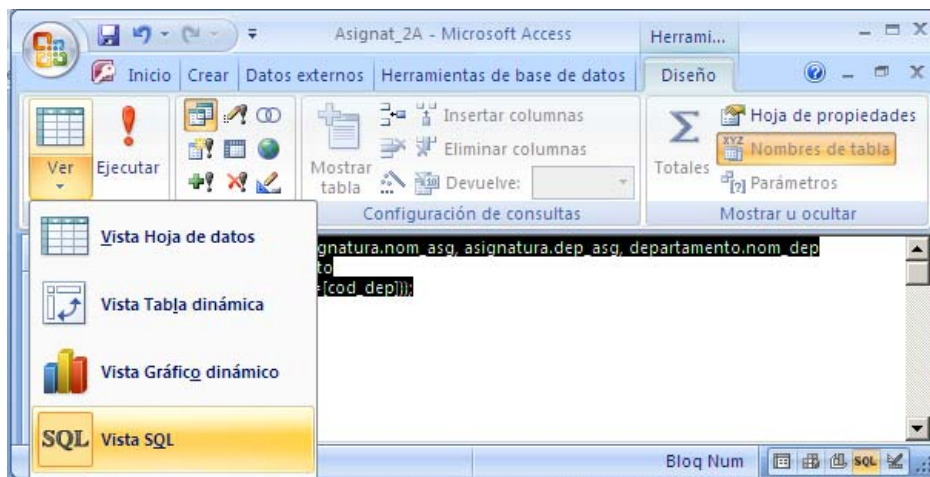


Figure 19: Where to click to visualise the SQL command associated to a query.

## 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 charge 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*.

## 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

5. 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)
6. 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)
7. 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? (9 rows)
8. 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)
9. Modify the previous query such that we only show the lecturers of the department DSIC. (13 rows)
10. 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)
11. 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).