



GOBIERNO
DE ESPAÑA

MINISTERIO
DE EDUCACIÓN

SECRETARÍA DE ESTADO
DE EDUCACIÓN Y
FORMACIÓN PROFESIONAL
DIRECCIÓN GENERAL DE
FORMACIÓN PROFESIONAL

INSTITUTO DE
TECNOLOGÍAS EDUCATIVAS

Base de datos en la enseñanza “OpenOffice”

6. Iniciación al lenguaje estructurado de consulta (SQL)



Formación en **Red**

INDICE

6 Iniciación al lenguaje estructurado de consultas (SQL).....	3
Introducción.....	3
Seleccionar registros, sentencia SELECT	4
Actividad 1	4
Filtrar registros, cláusula WHERE	7
Actividad 2	8
Operadores lógicos.....	8
Condiciones Y (AND) y O (OR) en la cláusula WHERE	8
Actividad 3	8
Actividad 4	9
Actividad 5	11
Ordenar registros, cláusula ORDER BY.....	13
Actividad 6	13
Cambiar el nombre de los campos en el resultado con ALIAS	14
Actividad 7	14
Obtener información de varias tablas.....	15
Actividad 8	16
Actividad 9	17
Actividad 10	17
Consultas de agrupación y totales con SQL	19
Criterios de filtrado en consultas de agrupación y totales, cláusula HAVING	20
Eliminar registros, comando DELETE.....	22
Sintaxis de las sentencias en el intérprete SQL	24
Ejemplo de eliminación de registros	24
Actividad 11	24
Insertar nuevos registros, sentencia INSERT.....	28
Actividad 12	28
Actualizar registros, sentencia UPDATE.....	29

6 Iniciación al lenguaje estructurado de consultas (SQL)

Introducción

El lenguaje estructurado de consultas, más comúnmente llamado SQL, lo conforman una serie de comandos, cláusulas y funciones que permiten realizar cualquier operación sobre la información almacenada en la base de datos.

SQL no es exactamente un lenguaje de programación pero lleva implícita la complejidad de estas herramientas. En realidad, los sistemas gestores de bases de datos como Microsoft Access o el mismo OpenOffice Base tienen como principal fin ocultar el lado oscuro de este lenguaje mediante el uso de utilidades gráficas. Aunque si tu intención es llegar a dominar el mundo de las bases de datos es imprescindible conocer ciertos aspectos de SQL.

OpenOffice Base tiene limitaciones que se pueden suplir mediante el uso de sentencias SQL. A lo largo de los siguientes apartados conocerás las sentencias más importantes.

Uno de los métodos más sencillos para trabajar directamente con sentencias SQL en OpenOffice Base es tener seleccionada la opción **Consultas** en la ventana principal de la base de datos y a continuación hacer clic sobre el elemento denominado **Crear consulta en vista SQL** como puedes ver en la figura 6.01. Después te encontrarás ante una fría ventana sobre la que deberás ir escribiendo las instrucciones necesarias.

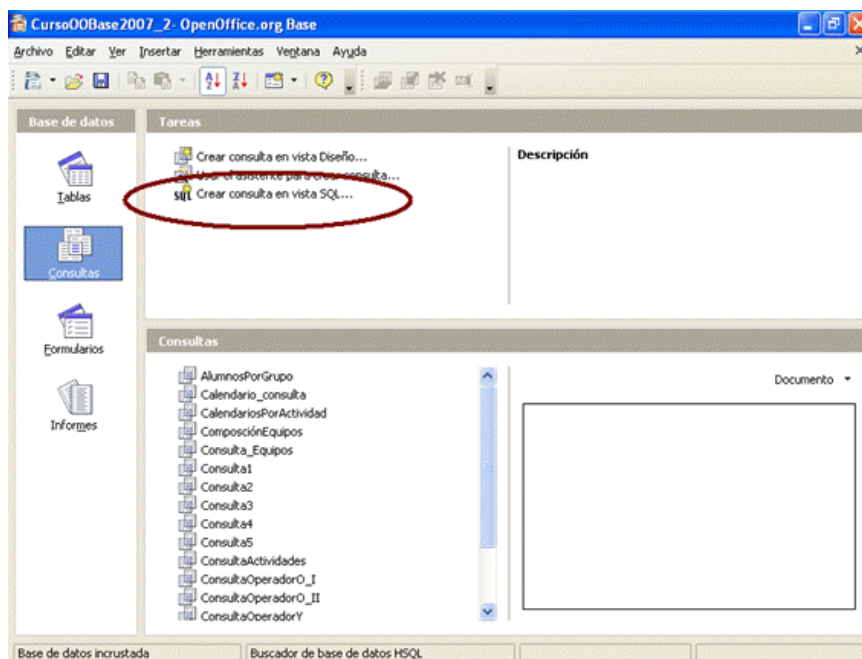


Figura 6.01

Mediante los comandos disponibles en el conjunto de elementos que agrupa SQL puedes realizar cualquier operación sobre una base de datos, desde seleccionar registros hasta crear una tabla o definir sus claves. En este caso, verás únicamente las instrucciones relacionadas con la manipulación de registros.

Seleccionar registros, sentencia SELECT

La sentencia SELECT va a permitir realizar operaciones de selección, ordenación, agrupación y filtrado de registros.

Actividad 1

1. Utilizando la base de datos **base6.odt** ,dentro del apartado de consultas de la base de datos, selecciona la opción **Crear consulta en vista SQL**. De esta forma tendrás acceso a la ventana que puedes ver en la figura 6.02.

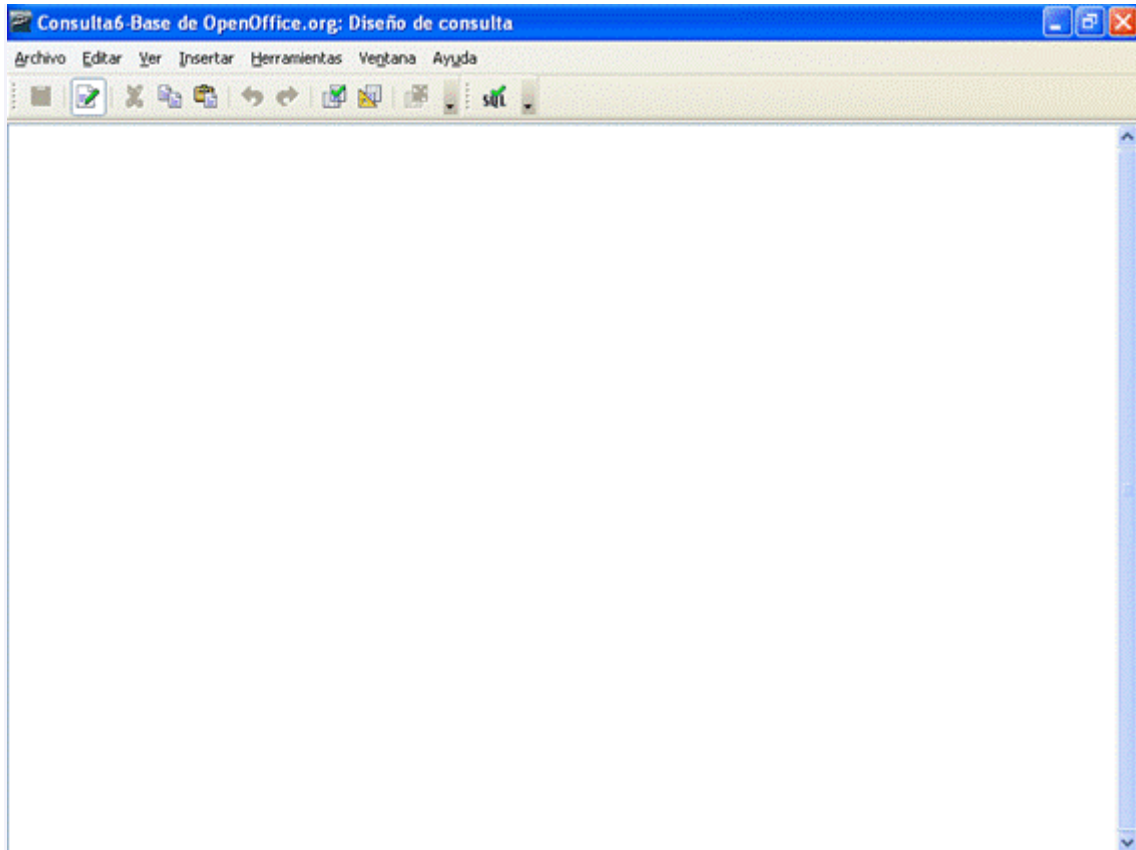


Figura 6.02

Para que te resulte más sencillo distinguir entre aquellas palabras que forman parte de la sintaxis de SQL y los parámetros que proporcionas, escribe siempre en mayúsculas las palabras reservadas que utiliza SQL.

2. Ahora escribe: **SELECT**.

1. A continuación debes indicar los campos de la tabla que deseas mostrar en la consulta. Si escribes un asterisco "*" le estás indicando al intérprete SQL que muestre todos los campos.
2. La siguiente palabra reservada de la instrucción es **"FROM"**. Escríbela a continuación, dejando un espacio en blanco entre el elemento anterior.
3. Para terminar es necesario proporcionar el nombre de la tabla desde la que obtendrás la información. En el primer ejemplo usa "Alumnos".

4. Por lo tanto, la primera sentencia SQL quedaría de la siguiente forma:
SELECT * FROM Alumnos
5. Ejecuta la consulta. Recuerda que basta con pulsar la tecla **F5** para hacerlo y comprobar los resultados como puedes ver en la figura 6.03.

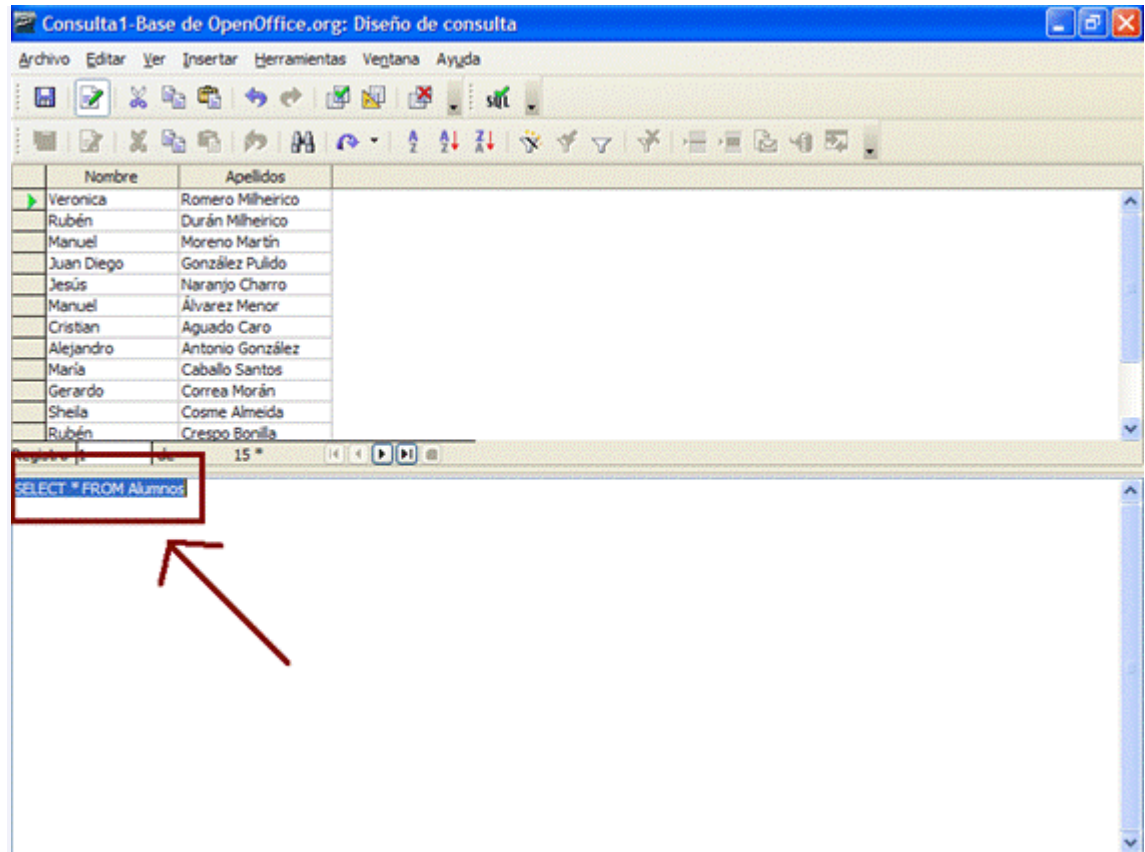


Figura 6.03

Como cualquier otro lenguaje de programación, no respetar la sintaxis de las instrucciones es uno de los problemas más habituales. Revisa cuidadosamente la sentencia completa antes de ejecutarla.

Para seleccionar sólo algunos campos de la tabla debes enumerarlos tras la palabra reservada SELECT y antes del FROM. Por ejemplo:

1. Selecciona de nuevo la opción **Crear consulta en vista SQL** para crear un nuevo diseño.
2. Ahora escribe: SELECT.
3. A continuación debes indicar los campos de la tabla que desees mostrar en la consulta. Escribe "Nombre", añade una coma, un espacio y escribe el siguiente nombre de campo: "Apellidos". Siguiendo este patrón podrás incluir tantos campos como necesites.
4. La siguiente palabra reservada de la instrucción es "FROM". Escríbela a continuación, dejando un espacio en blanco entre el elemento anterior.
5. Para terminar es necesario proporcionar el nombre de la tabla desde la que obtendrás la información: *Alumnos*.
6. Por lo tanto, después de estos pasos la sentencia SQL quedaría de la siguiente forma:
SELECT Nombre, Apellidos FROM Alumnos
7. Ejecuta la consulta para comprobar los resultados como puedes observar en la figura 6.04.

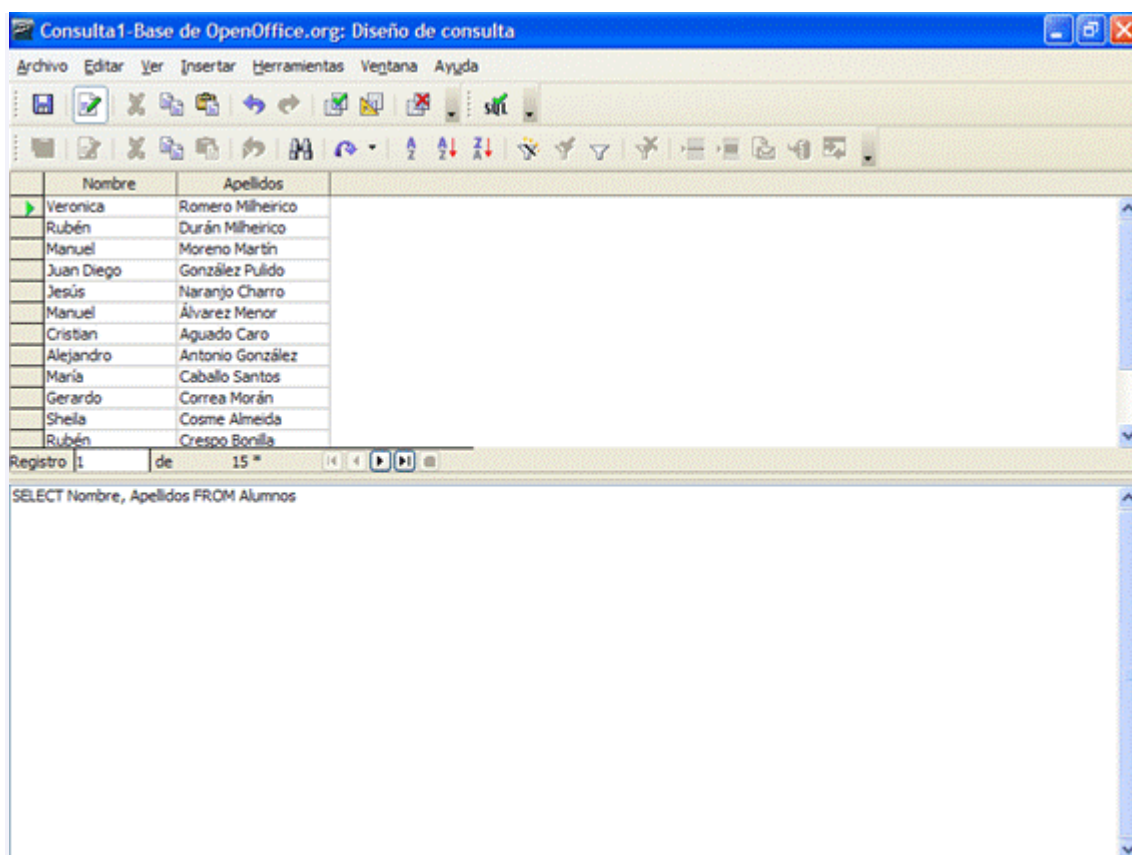


Figura 6.04

Si lo crees conveniente puedes anteponer el nombre de la tabla al nombre del campo en la definición de la consulta del siguiente modo:

SELECT Alumnos.Nombre, Alumnos.Apellidos FROM Alumnos

Si bien es cierto que sólo es realmente necesario cuando hay más de una tabla en la consulta y se hace indispensable identificar el origen de cada campo.

Filtrar registros, cláusula WHERE

Como ya sabes una de las funciones principales de una consulta es seleccionar sólo aquellos registros de la tabla o tablas seleccionadas que necesites en cada caso. Pues bien, si quieres hacerlo mediante sentencias SQL es imprescindible añadir la cláusula WHERE al comando SELECT.

Por ejemplo, a continuación mostramos cómo diseñar la consulta SQL que nos permitiría seleccionar aquellos alumnos cuya fecha de nacimiento sea menor que el 31 de Diciembre de 1992.

1. Selecciona la opción Crear consulta en vista SQL.
2. Empieza escribiendo: SELECT.
3. A continuación debes indicar los campos de la tabla que deseas mostrar en la consulta. En este caso utiliza el nombre, los apellidos y la fecha de nacimiento.
4. La siguiente palabra reservada de la instrucción es "FROM". Escríbela a continuación, dejando un espacio en blanco entre el elemento anterior.
5. Ahora indica el nombre de la tabla desde la que obtendrás la información, en este caso "Alumnos".
6. Después, añade la cláusula WHERE para indicar la condición de filtrado.

El criterio sería el siguiente: FechaNacimiento <= '1992-12-31'. El formato es distinto al que conoces hasta ahora pero es la sintaxis que acepta el intérprete de SQL para las fechas.

1. Por lo tanto, después de estos pasos la sentencia SQL quedaría de la siguiente forma:
SELECT Nombre, Apellidos, FechaNacimiento FROM Alumnos WHERE FechaNacimiento <= '1992-12-31'
2. Ejecuta la consulta para comprobar los resultados (figura 6.05).

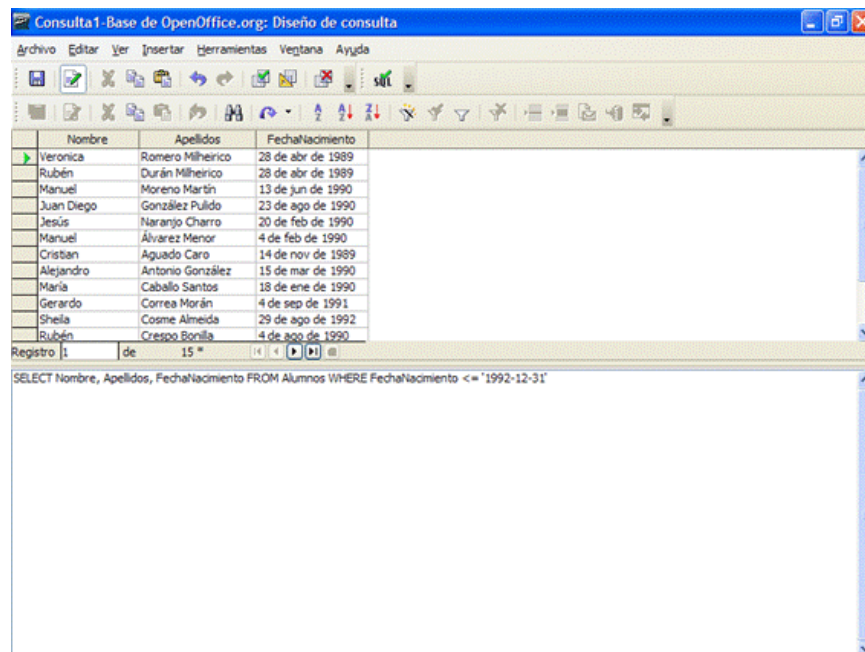


Figura 6.05

Actividad 2

Diseña una consulta SQL que permita seleccionar el nombre y apellidos de los profesores que vivan en la provincia de Badajoz.

Operadores lógicos

Los operadores lógicos que puedes utilizar en sentencias SQL son los mismos que ya conoces:

Operador	Se lee	Resultado
=	<i>Igual a</i>	Devuelve los registros que coinciden con la condición.
<	<i>Menor que</i>	Muestra aquellos registros cuyos valores son estrictamente menores que la condición.
>	<i>Mayor que</i>	Muestra aquellos registros cuyos valores son estrictamente mayores que la condición.
<=	<i>Menor o igual que</i>	Devuelve los registros cuyos valores son menores o iguales a la condición.
>=	<i>Mayor o igual que</i>	Devuelve los registros cuyos valores son mayores o iguales a la condición.
<>	<i>Distinto de</i>	Sólo muestra aquellos registros que tienen valores distintos a la condición.

Condiciones Y (AND) y O (OR) en la cláusula WHERE

Al igual que ocurre con las consultas en el modo Diseño, mediante sentencias SQL también podemos utilizar los operadores Y (AND) y O (OR) y por supuesto, no se ve alterada su lógica de funcionamiento.

Veamos en primer lugar un ejemplo práctico donde se utiliza el operador OR. Concretamente, diseñaremos una consulta SQL que nos permita conocer las tutorías que tienen lugar los Martes o los Jueves.

Actividad 3

1. Selecciona la opción **Crear consulta en vista SQL** y empieza escribiendo: SELECT.
2. A continuación debes indicar los campos de la tabla que deseas mostrar en la consulta. En este caso utiliza *Profesor*, *Curso*, *DiaSemana* y *HoraTutoria*.
3. La siguiente palabra reservada de la instrucción es "FROM". Escríbela a continuación, dejando un espacio en blanco entre el elemento anterior.
4. Ahora indica el nombre de la tabla desde la que obtendrás la información, en este caso *Tutorias*.
5. Después añade la cláusula WHERE para indicar la primera condición de filtrado.
6. El criterio sería el siguiente: *DiaSemana* = 'Martes'. No olvides las comillas simples para encerrar el literal de la condición de búsqueda.

7. Ahora escribe el operador lógico OR y deja un espacio.
8. A continuación indica la segunda condición: *DiaSemana* = 'Jueves'.
- 9.

Por lo tanto, después de estos pasos la sentencia SQL quedaría de la siguiente forma:

**SELECT Profesor, Curso, DiaSemana, HoraTutoria
FROM Tutorias WHERE DiaSemana = 'Martes' OR
DiaSemana = 'Jueves'**

1. Comprueba los resultados ejecutando la consulta. En la figura 6.06 puedes ver el aspecto de la sentencia junto con los resultados obtenidos.

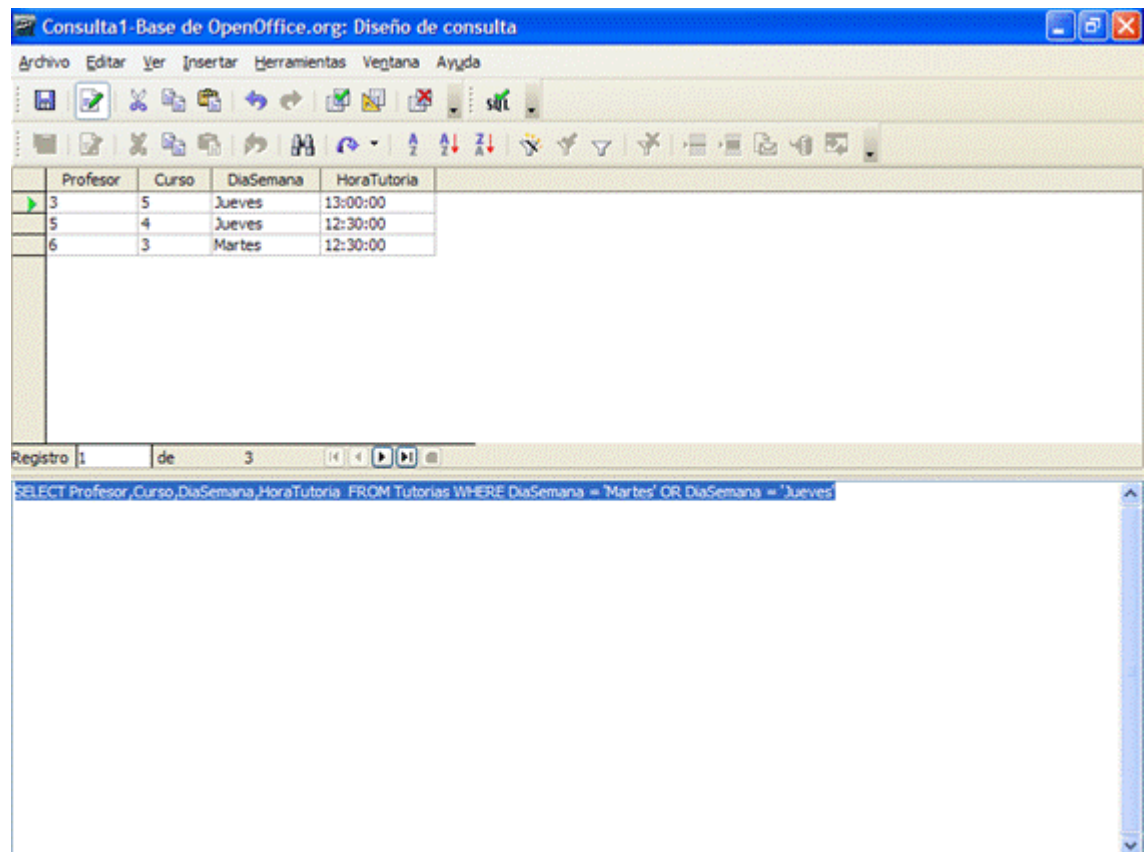


Figura 6.06

Quizás todos estos pasos te puedan parecer un poco complejos al principio pero si prestas un poco de atención comprobarás que incluso se pueden leer: SELECCIONA LOS CAMPOS PROFESOR CURSO, DIASEMANA Y HORATUTORIA DE LA TABLA TUTORIAS DONDE EL DIA DE LA SEMANA ES MARTES O JUEVES.

Ahora realiza otro ejemplo, pero esta vez con el operador AND. La idea es diseñar una consulta que permita conocer los encuentros que se disputarán entre dos fechas concretas.

Actividad 4

1. Selecciona la opción **Crear consulta en vista SQL** y empieza escribiendo: SELECT.
2. A continuación indica los campos de la tabla que deseas mostrar en la consulta. En este caso utiliza Jornada, Lugar, Fecha, EquipoA y EquipoB.

3. Escribe "FROM", dejando un espacio en blanco entre el elemento anterior y el siguiente.
4. Ahora indica el nombre de la tabla desde la que obtendrás la información, en este caso "Calendario".
5. Después añade la cláusula WHERE para indicar la primera condición de filtrado.
- 6.

El criterio sería el siguiente: Fecha >= '2007-02-01'. Evidentemente te parecerá un poco extraño el formato utilizado para las fechas, pero es así como lo entiende el intérprete de SQL de OpenOffice Base.

1. Ahora escribe el operador lógico AND y deja un espacio.
2. A continuación indica la segunda condición: Fecha <= '2007-02-19'.
3. Después de estos pasos la sentencia SQL quedaría de la siguiente forma:

```
SELECT Jornada, Fecha, Lugar, EquipoA,  
EquipoB FROM Calendario WHERE Fecha  
>= '2007-02-01' AND Fecha <= '2007-02-  
19'
```

4. Ejecuta la consulta. De nuevo, puedes ver en la figura 6.07 el resultado de la consulta.

Como has podido comprobar el interprete SQL de OpenOffice Base tampoco utilizas las almohadillas para representar las fechas, como hacías en las consultas. En este caso, siempre debes usar comillas simples, separar con un guión el año, mes y día y escribirlo en este orden. Estas instrucciones son imprescindibles a la hora incluir fechas en tus sentencias SQL, y teniendo en cuenta que es algo distinto de lo que conocemos, nuestra recomendación es que practiques un poco con todo lo visto hasta ahora.

CursoOpenOfficeBase04: SQL encuentros a disputar entre dos fechas

Archivo Editar Ver Herramientas Ventana Ayuda



	Jornada	Fecha	Lugar	EquipoA	EquipoB	
▶	0	12/02/07	Pista 1	0	1	
	1	12/02/07	Pista 2	2	3	
	2	14/02/07	Pista 1	4	5	
	3	14/02/07	Pista 2	0	3	

Figura 6.07

Hay que insistir en la importancia de las comillas simples a la hora de expresar las fechas en una consulta de criterios. Por este motivo para que no tengas ninguna duda en la figura 6.08 puedes ver la tecla concreta a la que hacemos referencia.



Figura 6.08

Una vez diseñada la consulta y comprobados sus resultados utiliza el botón Guardar para dar un nombre y archivar la consulta en la base de datos.

Con el objetivo de abarcar todas las posibilidades, veamos a continuación como sería el diseño de una consulta donde se combinen al mismo tiempo operadores AND y OR. Concretamente, el objetivo es conocer los encuentros que se juegan el día 12/02/2007 o el 14/02/2007 pero sólo en la Pista 1.

Actividad 5

1. Selecciona la opción **Crear consulta en vista SQL** y empieza escribiendo: SELECT.
2. A continuación indica los campos de la tabla que desees mostrar en la consulta. En este caso utiliza Jornada, Lugar, Fecha, EquipoA y EquipoB.
3. Escribe "FROM", dejando un espacio en blanco entre el elemento anterior y el siguiente.
4. Ahora indica el nombre de la tabla desde la que obtendrás la información, en este caso "Calendario".
5. Después añade la cláusula WHERE para indicar la primera condición de filtrado.
6. El criterio sería el siguiente: Fecha = '2007-02-12'.
7. Ahora escribe el operador lógico AND y deja un espacio.
8. A continuación escribe Lugar = 'Pista 1'. De esta forma compones el primer criterio.
9. Deja un espacio, escribe OR y deja otro espacio.
10. El criterio siguiente sería: Fecha = '2007-02-14'.
11. Ahora escribe el operador lógico AND y deja un espacio.
12. A continuación escribe Lugar = 'Pista 1'. De esta forma compones el segundo criterio.
13. Después de estos pasos la sentencia SQL quedaría de la siguiente forma. Revisa con cuidado todo lo escrito y no olvides incluir los paréntesis tal y como puedes ver a continuación:

```
SELECT Jornada, Fecha, Lugar, EquipoA, EquipoB FROM  
Calendario WHERE ((Fecha = '2007-02-12' AND Lugar =  
'Pista 1') OR (Fecha = '2007-02-14' AND Lugar = 'Pista  
1'))
```

14. Comprueba los resultados ejecutando la consulta y observa la figura 6.09.

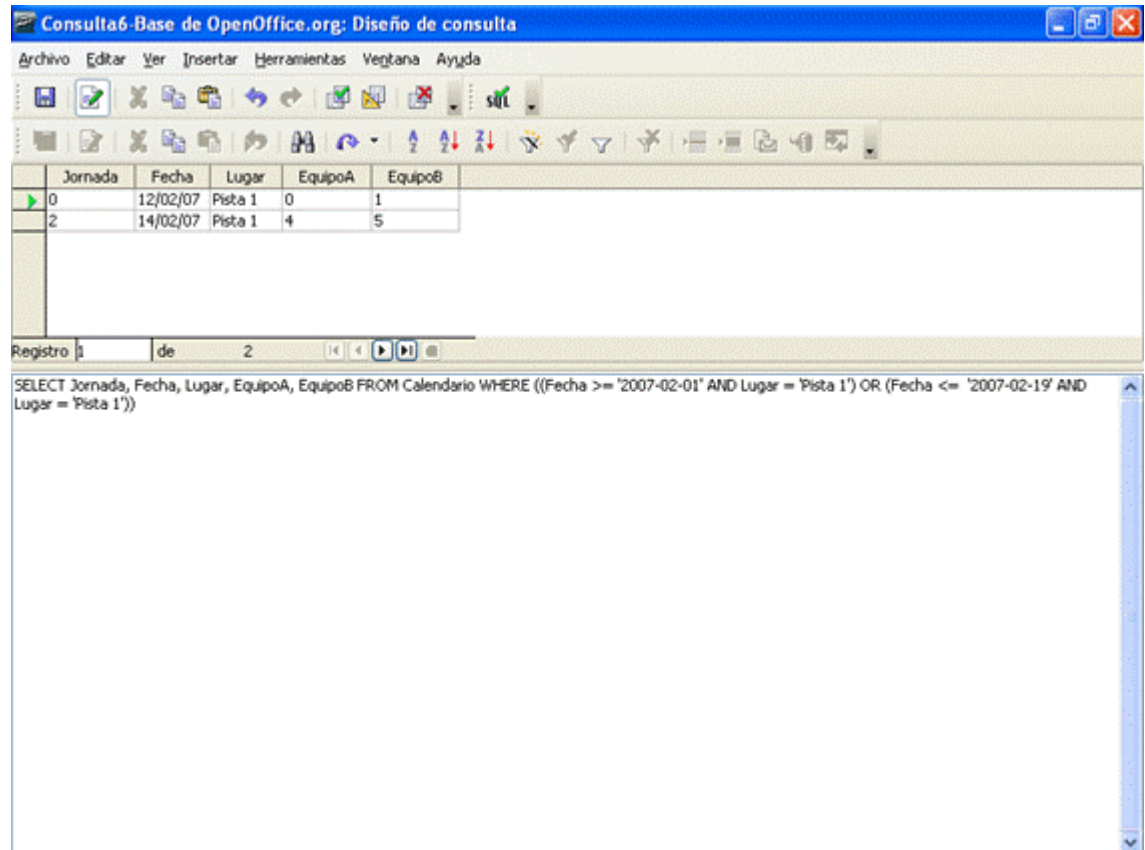


Figura 6.09

Cuando anidas varios criterios de filtrado es imprescindible utilizar los paréntesis para agrupar cada uno de los bloques que conforman la consulta a partir de la cláusula WHERE.

Ordenar registros, cláusula ORDER BY

Otro aspecto importante cuando diseñas consultas son los criterios de ordenación. Con ellos, el resultado presentará los datos clasificados de la forma que necesites en cada caso. En SQL la cláusula dedicada a este fin se denomina ORDER BY y debes situarla después de todos los campos de filtrados asociados a la cláusula WHERE.

Como siempre, lo mejor será ver un ejemplo pero en este caso no demasiado complicado: obtendremos un listado ordenado alfabéticamente de todos los alumnos que pertenezcan al grupo 2.

1. Selecciona la opción **Crear consulta en vista SQL** y empieza escribiendo: **SELECT**.
2. A continuación indica los campos de la tabla que desees mostrar en la consulta. En este caso utiliza el * para añadir todos los campos.
3. Escribe "FROM", dejando un espacio en blanco entre el elemento anterior y el siguiente.
4. Indica el nombre de la tabla desde la que obtendrás la información, en este caso "Alumnos".
5. Después añade la cláusula WHERE para indicar la primera condición de filtrado.
6. El criterio sería el siguiente: Grupo = 2.
7. Ahora deja un espacio, escribe la cláusula ORDER BY y deja otro espacio.
8. El campo que usarás como patrón de ordenación será el Apellido, por lo tanto escríbelo después de ORDER BY.
9. Después de estos pasos la sentencia SQL quedaría de la siguiente forma:

SELECT * FROM Alumnos WHERE Grupo = 2 ORDER BY Apellidos

10. Comprueba los resultados ejecutando la consulta y observa la figura 6.10.

The screenshot shows the 'Consulta1-Base de OpenOffice.org: Diseño de consulta' window. It displays a table with columns: Expediente, Nombre, Apellidos, Grupo, and FechaNacimiento. The data is filtered to show only students in 'Grupo 2' and is ordered by 'Apellidos' (last name). The SQL statement at the bottom is: `SELECT * FROM Alumnos WHERE Grupo = 2 ORDER BY Apellidos`.

Expediente	Nombre	Apellidos	Grupo	FechaNacimiento
49	Francisco Manuel	Aranha Chaves	2	4 de feb de 1989
43	Coral	Cardenas Montero	2	2 de jul de 1995
41	Verónica	Carrero Carrio	2	28 de nov de 1995
48	Jonathan	Franco Calderón	2	19 de feb de 1989
47	Samuel	Garrido Bueno	2	15 de jun de 1994
50	Esther María	Jaramago Flores	2	29 de abr de 1990
39	David	Lindo González	2	9 de abr de 1990
40	Gema	Marcos López	2	2 de may de 1990
26	Marta	Muniz Alba	2	12 de ene de 1994
27	Rubén	Nicolas Pablos	2	17 de oct de 1994
28	Jessica	Nogales Gordillo	2	8 de sep de 1995
29	José Antonio	Pardo Montes	2	23 de ago de 1994

Figura 6.10

El campo Grupo, al ser un valor estrictamente numérico, no necesita añadirle comillas simples en la sentencia SQL.

Cuando quieras ordenar por más de un campo, escríbelos uno a continuación de otro y sepáralos con una coma.

Actividad 6

Diseña una consulta que nos permita conocer todos aquellos profesores que viven en Badajoz o en Cáceres. Además, ordena el resultado en primer lugar por la provincia y en segundo lugar por el apellido del profesor.

Cambiar el nombre de los campos en el resultado con ALIAS

Haz un poco de memoria y recuerda que al tratar las consultas en OpenOffice Base utilizabas la propiedad "alias" para modificar el nombre de cualquier campo en el resultado de la consulta. Cuando diseñas consultas mediante SQL también se encuentra disponible esta característica. Compruébalo en el siguiente ejemplo.

Actividad 7

1. Selecciona la opción **Crear consulta en vista SQL** y empieza escribiendo: **SELECT**.
2. A continuación indica los campos de la tabla que deseas mostrar en la consulta. En este caso utiliza el campo *Nombre*.
3. Deja un espacio y escribe la palabra reservada **AS** y después el nombre con el que representarás el campo en la consulta, por ejemplo "Nombre Departamento".

Es necesario encerrar entre comillas el nombre al estar compuesto por más de una palabra. Si el alias está compuesto por una única palabra no sería necesario utilizar comillas dobles.

4. Escribe "FROM", dejando un espacio en blanco entre el elemento anterior y el siguiente.
5. Indica el nombre de la tabla desde la que obtendrás la información, en este caso "Departamentos".
6. A continuación, ordena el resultado alfabéticamente por el nombre del departamento. Para ello, añade la cláusula **ORDER BY** y a continuación el campo **Nombre**.
7. Después de estos pasos la sentencia SQL quedaría de la siguiente forma:
SELECT Nombre AS "Nombre Departamento" FROM Departamentos ORDER BY Nombre
8. Ejecuta la consulta y comprueba los resultados sean similares a los que muestra la figura 6.11.

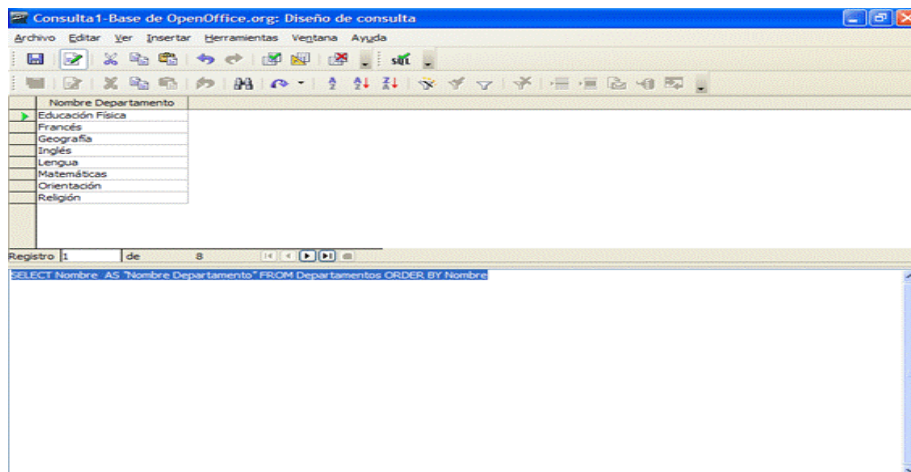


Figura 6.11

SQL permite utilizar en la cláusula **ORDER BY** el nombre del alias en lugar del nombre original del campo. De este modo también mejorarás la comprensión general de la instrucción. La sentencia anterior quedaría del siguiente modo:

SELECT Nombre AS "Nombre del equipo" FROM Equipos ORDER BY "Nombre del equipo"

Obtener información de varias tablas

De igual modo que existe la posibilidad en la vista Diseño de vincular dos o más tablas para obtener resultados combinados, SQL también dispone de las herramientas necesarias para realizar este tipo de tareas. El proceso es sin duda más complicado que lo visto con el diseñador de consultas, pero a cambio tendrás mucho más control sobre todo el proceso.

SQL propone un método para identificar la tabla asociada a cada uno de los campos cuando trabajas con varias tablas. La sintaxis consiste en anteponer el nombre de la tabla al nombre del campo y separarlos por un punto:

NombreTabla.NombreCampo

Antes de ver el primer ejemplo, revisa los tipos de relación que se podían definir entre las tablas.

Interna: El resultado muestra sólo los registros en los que el campo vinculado de ambas tablas sea el mismo.

Izquierda: En este caso, el resultado muestra todos los registros de la tabla izquierda, y sólo aquellos de la tabla derecha donde coincida el campo vinculado.

Derecha: Con este modelo ocurriría justo lo contrario que en el anterior, aparecerían todos los registros de la tabla derecha y sólo aquellos de la tabla izquierda en los que coincidan los campos vinculados.

Completa: Muestra todos los registros de ambas tablas.

Por supuesto, en SQL también existe la posibilidad de reproducir cada uno de estos tipos de unión pero en estos momentos no es necesario añadir este nivel de complejidad al curso, así que utiliza la fórmula más sencilla, la vinculación de tablas mediante la cláusula WHERE. Este método equivale a seleccionar aquellos registros donde ambas tablas tengan coincidencia que, por otra parte, es la situación más común cuando trabajas con bases de datos.

En nuestro primer ejemplo, utilizaremos las tablas Alumnos y Cursos para representar en una consulta el grupo al que pertenece cada uno de los alumnos que forman nuestra base de datos.

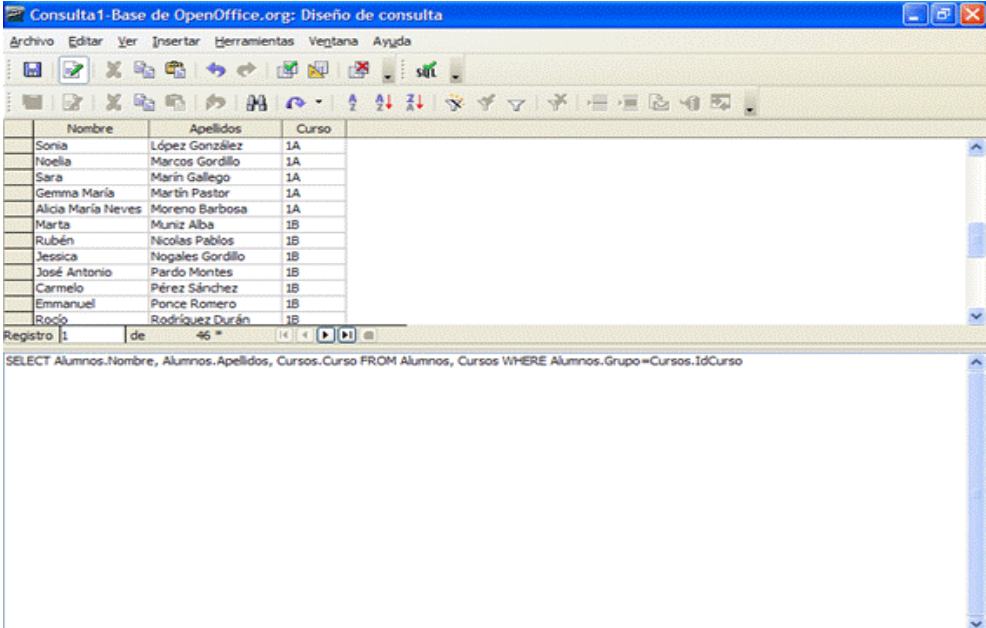
1. Selecciona la opción **Crear consulta en vista SQL** y empieza escribiendo: SELECT.
2. A continuación indica los campos de la tabla que desees mostrar en la consulta, pero recuerda que ahora debes anteponer el nombre de la tabla. Incluye los campos Nombre y Apellidos de la tabla Alumnos.
3. De la tabla Cursos utiliza sólo el campo "Curso".
4. Ahora añade la cláusula FROM y escribe el nombre de las dos tablas implicadas: Alumnos y Cursos.
- 5.

Esta es la parte más delicada de la consulta, donde debes vincular los campos de ambas tablas que permitirán obtener los resultados correctos. En este caso es necesario relacionar el campo Grupo de la tabla Alumnos con el campo IdCurso de la tabla Curso. Escribe en primer lugar la cláusula WHERE y después Alumnos.Grupo=Cursos.IdCurso.

1. La instrucción SQL queda del siguiente modo:
SELECT Alumnos.Nombre, Alumnos.Apellidos,

Cursos.Curso FROM Alumnos, Cursos WHERE Alumnos.Grupo=Cursos.IdCurso

2. Por ahora es suficiente, ejecuta la consulta. En la figura 6.12 puedes ver su aspecto y el resultado obtenido.



Consulta1-Base de OpenOffice.org: Diseño de consulta

Nombre	Apellidos	Curso
Sonia	López González	1A
Noelia	Marcos Gordillo	1A
Sara	Marín Gallego	1A
Gemma María	Martín Pastor	1A
Alicia María Neves	Moreno Barbosa	1A
Marta	Muniz Alba	1B
Rubén	Nicolas Pablos	1B
Jessica	Nogales Gordillo	1B
José Antonio	Pardo Montes	1B
Carmelo	Pérez Sánchez	1B
Emmanuel	Ponce Romero	1B
Rocio	Rodríguez Durán	1B

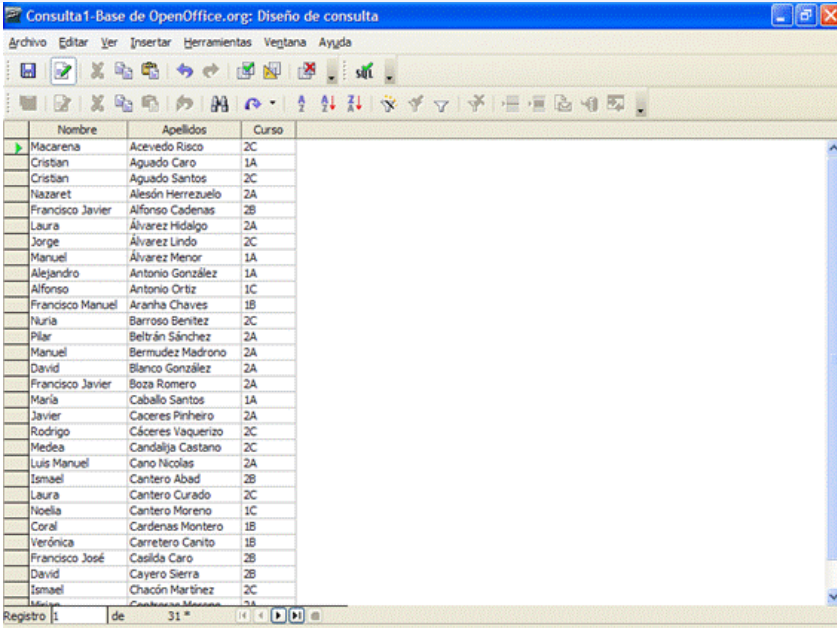
Registro 1 de 46

SELECT Alumnos.Nombre, Alumnos.Apellidos, Cursos.Curso FROM Alumnos, Cursos WHERE Alumnos.Grupo=Cursos.IdCurso

Figura 6.12

Actividad 8

Pero puedes depurar un poco más el ejemplo. ¿Qué tal si ordenas el resultado alfabéticamente por los apellidos del alumno? La figura 6.13 muestra el resultado de la nueva consulta. Una vez terminada, guarda esta consulta con el nombre de Actividad8.



Consulta1-Base de OpenOffice.org: Diseño de consulta

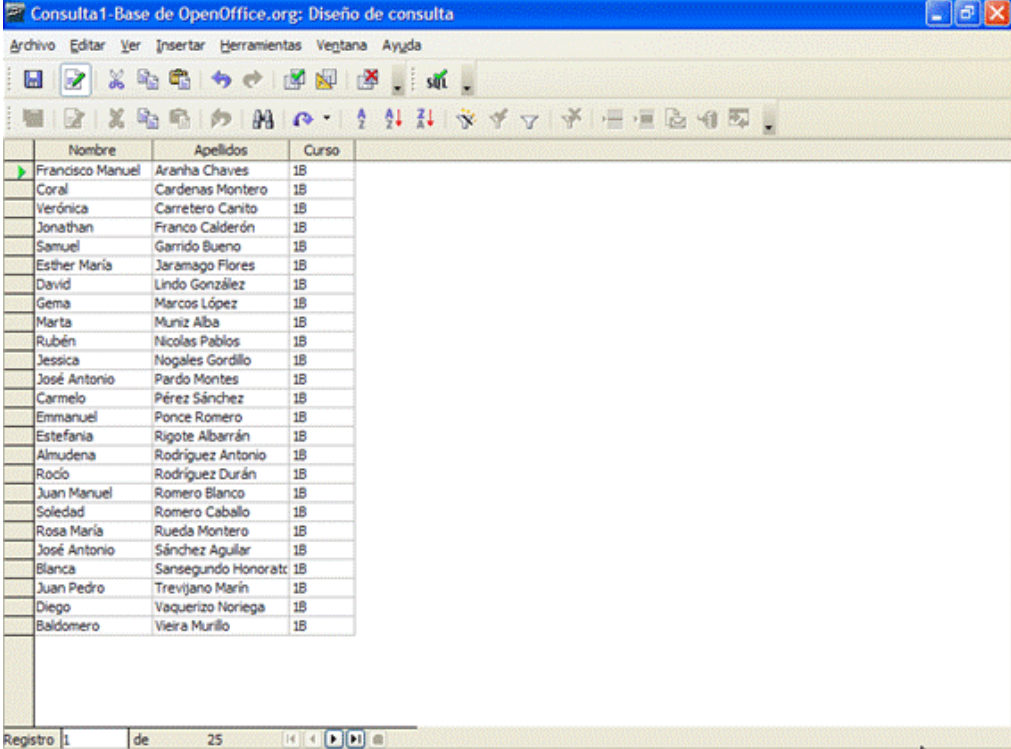
Nombre	Apellidos	Curso
Macarena	Acevedo Risco	2C
Cristian	Aguado Caro	1A
Cristian	Aguado Santos	2C
Nazaret	Alesón Herrezuelo	2A
Francisco Javier	Alfonso Cadenas	2B
Laura	Álvarez Hidalgo	2A
Jorge	Álvarez Lindo	2C
Manuel	Álvarez Menor	1A
Alejandro	Antonio González	1A
Alfonso	Antonio Ortiz	1C
Francisco Manuel	Aranha Chaves	1B
Nuria	Barroso Benitez	2C
Pilar	Beltrán Sánchez	2A
Manuel	Bermudez Madrono	2A
David	Blanco González	2A
Francisco Javier	Boza Romero	2A
María	Caballo Santos	1A
Javier	Caceres Pinheiro	2A
Rodrigo	Cáceres Vaquerizo	2C
Medea	Candalija Castano	2C
Luis Manuel	Cano Nicolás	2A
Ismael	Cantero Albad	2B
Laura	Cantero Curado	2C
Noelia	Cantero Moreno	1C
Coral	Cardenas Montero	1B
Verónica	Carretero Canito	1B
Francisco José	Casilda Caro	2B
David	Cayero Sierra	2B
Ismael	Chacón Martínez	2C

Registro 1 de 31

Figura 6.13

Actividad 9

También puedes añadir una condición a la consulta anterior. Por ejemplo, muestra únicamente los alumnos de 1B. Para ello, debes añadir una nueva condición a la cláusula WHERE, antes de ORDER BY y enlazarla mediante el operador AND. El resultado que obtengas debe ser similar al que muestra la figura 6.14. Una vez terminada, guarda esta consulta con el nombre de Actividad9.



The screenshot shows the 'Consulta1-Base de OpenOffice.org: Diseño de consulta' window. It displays a table with three columns: 'Nombre', 'Apellidos', and 'Curso'. The data is sorted by 'Curso' in ascending order. The table contains 25 records, all of which are in '1B'.

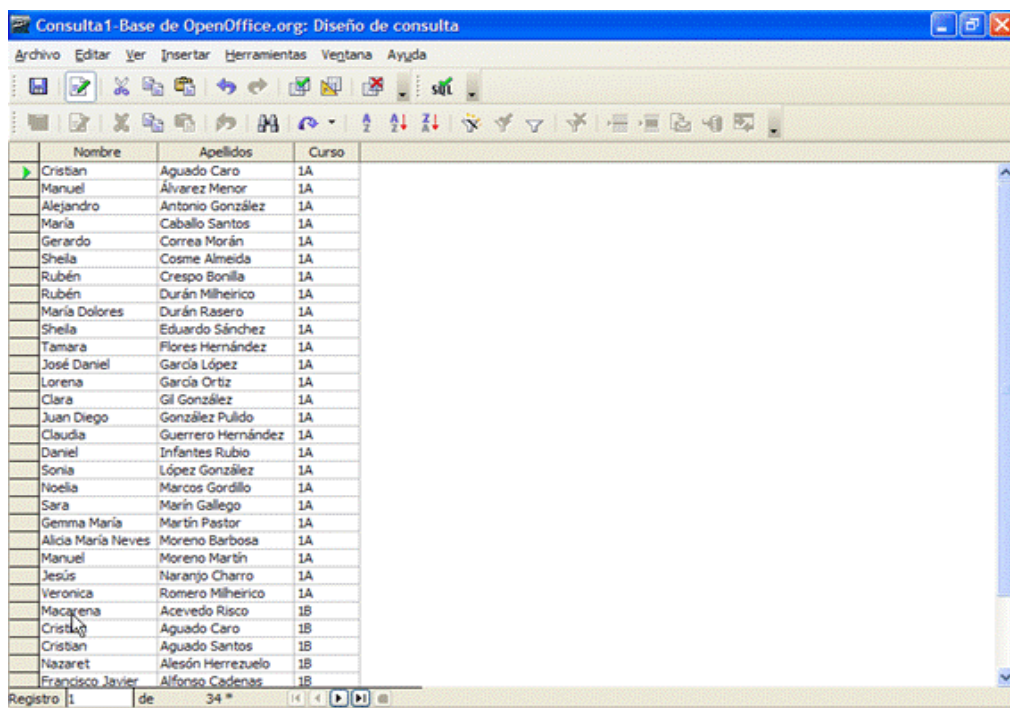
Nombre	Apellidos	Curso
Francisco Manuel	Aranha Chaves	1B
Coral	Cardenas Montero	1B
Verónica	Carretero Canito	1B
Jonathan	Franco Calderón	1B
Samuel	Garrido Bueno	1B
Esther María	Jaramago Flores	1B
David	Lindo González	1B
Gema	Marcos López	1B
Marta	Muniz Alba	1B
Rubén	Nicolás Pablos	1B
Jessica	Nogales Gordillo	1B
José Antonio	Pardo Montes	1B
Carmelo	Pérez Sánchez	1B
Emmanuel	Ponce Romero	1B
Estefanía	Rigote Albarrán	1B
Almudena	Rodríguez Antonio	1B
Rocío	Rodríguez Durán	1B
Juan Manuel	Romero Blanco	1B
Soledad	Romero Caballo	1B
Rosa María	Rueda Montero	1B
José Antonio	Sánchez Aguilar	1B
Blanca	Sansegundo Honorati	1B
Juan Pedro	Trevijano Marín	1B
Diego	Vaquerizo Noriega	1B
Baldomero	Vieira Murillo	1B

Figura 6.14

Recuerda que el criterio "1B" al tratarse de un literal es necesario encerrarlo entre comillas simples, si no lo haces así el intérprete SQL de OpenOffice Base generará un error de sintaxis.

Actividad 10

Ahora riza un poco más el rizo, y modifica la consulta diseñada en la actividad 9, para que muestre los alumnos que pertenezcan a 1A o a 1B, pero esta vez los ordenarás primero por el grupo y a continuación por el apellido. La forma de conseguirlo es utilizar un nuevo criterio enlazado con el anterior mediante el operador OR ya que quieres mostrar cualquiera de las dos opciones. Observa el resultado en la figura 6.15. Guarda esta consulta con el nombre de Actividad10 y la base de datos completa como *mibase6.odt*.



The screenshot shows the 'Consulta1-Base de OpenOffice.org: Diseño de consulta' window. It displays a table with three columns: 'Nombre', 'Apellidos', and 'Curso'. The table contains 34 records, with the first 30 records having a 'Curso' of '1A' and the last 4 records having a 'Curso' of '1B'. The status bar at the bottom indicates 'Registro 1 de 34 *'.

Nombre	Apellidos	Curso
Cristian	Aguado Caro	1A
Manuel	Álvarez Menor	1A
Alejandro	Antonio González	1A
Maria	Caballo Santos	1A
Gerardo	Correa Morán	1A
Sheila	Cosme Almeida	1A
Rubén	Crespo Bonilla	1A
Rubén	Durán Milheirico	1A
Maria Dolores	Durán Rasero	1A
Sheila	Eduardo Sánchez	1A
Tamara	Flores Hernández	1A
José Daniel	García López	1A
Lorena	García Ortiz	1A
Clara	Gil González	1A
Juan Diego	González Pulido	1A
Claudia	Guerrero Hernández	1A
Daniel	Infantes Rubio	1A
Sonia	López González	1A
Noelia	Marcos Gordillo	1A
Sara	Marín Gallego	1A
Gemma Maria	Martín Pastor	1A
Alicia María Neves	Moreno Barbosa	1A
Manuel	Moreno Martín	1A
Jesús	Naranjo Charro	1A
Veronica	Romero Milheirico	1A
Macarena	Acevedo Risco	1B
Cristian	Aguado Caro	1B
Cristian	Aguado Santos	1B
Nazaret	Alesón Herrezuelo	1B
Francisco Javier	Alfonso Cadenas	1B

Figura 6.15

Quizás te parezcan un poco complicadas las últimas actividades, pero analízalas con atención y comprobarás que todas siguen una pauta lógica y a medida que adquieras más destreza, las entenderás sin mayor problemas.

Consultas de agrupación y totales con SQL

En el apartado anterior tratamos la forma de relacionar varias tablas en una sentencia SQL, pero aún hay más. Mediante la cláusula GROUP BY puedes utilizar uno o varios campos para agrupar registros y realizar determinadas operaciones con ellos. Algunas de estas funciones son las mismas que ya tratamos en el capítulo de consultas y permiten:

COUNT: Cuenta el total de elementos de un grupo.

SUM: Suma los valores numéricos de los registros agrupados.

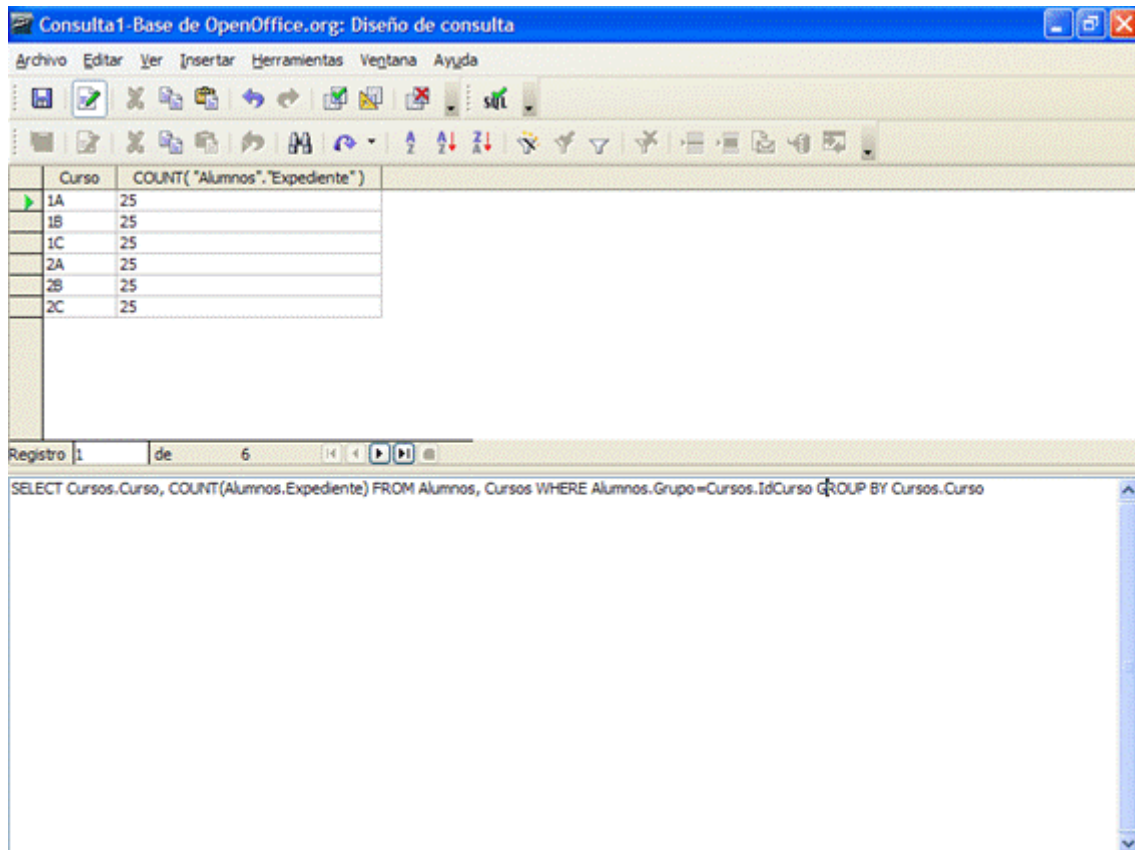
MIN: Muestra el valor más pequeño de un grupo.

MAX: Muestra el valor máximo de todo el conjunto de registros agrupados.

AVERAGE: Calcula la media de todos los valores seleccionados.

A continuación, diseñemos una nueva consulta donde el objetivo será contar el número total de alumnos que componen cada grupo (1A, 1B, 1C, 2A...):

1. Selecciona la opción **Crear consulta en vista SQL** y empieza escribiendo:
SELECT.
2. A continuación indica los campos de la tabla que deseas mostrar en la consulta. En primer lugar indica el nombre del grupo.
3. En segundo lugar, debes escribir la sentencia que calcule el total de alumnos:
COUNT (Alumnos.Expediente).
4. Ahora añade la cláusula FROM y escribe el nombre de las dos tablas implicadas: Alumnos y Cursos.
5. Toca el turno de vincular los campos de ambas tablas que te permitirán obtener los resultados correctos. En este caso es necesario relacionar el campo Grupo de la tabla Alumnos con el campo IdCurso de la tabla Curso. Escribe en primer lugar la cláusula WHERE y después
Alumnos.Grupo=Cursos.IdGrupo.
6. Por último debes indicar el campo que utilizarás como elemento de agrupación. Escribe GROUP BY Cursos.Curso.
7. La instrucción SQL queda del siguiente modo:
SELECT Cursos.Curso, COUNT(Alumnos.Expediente)
FROM Alumnos, Cursos
WHERE Alumnos.Grupo=Cursos.IdCurso
GROUP BY Cursos.Curso
8. Ejecútala y comprueba que los resultados sean similares a los que puedes ver en la figura 6.16.



Curso	COUNT(Alumnos.Expediente)
1A	25
1B	25
1C	25
2A	25
2B	25
2C	25

Figura 6.16

Una buena idea podría ser añadir un alias al campo calculado y así mejorar la compresión de los resultados:

```
SELECT Cursos.Curso, COUNT(Alumnos.Expediente) AS "Total Alumnos por Grupo"  
FROM Alumnos, Cursos  
WHERE Alumnos.Grupo=Cursos.IdCurso  
GROUP BY Cursos.Curso
```

Recuerda las dobles comillas en el nombre de alias cuando se trata de más de una palabra.

Criterios de filtrado en consultas de agrupación y totales, cláusula HAVING

Cuando se trata de consultas de agrupación y totales, la forma de utilizar criterios de ordenación es algo distinta de lo que has visto hasta ahora. La diferencia se encuentra en que debes utilizar una nueva cláusula: HAVING (figura 6.17). Por ejemplo, si sólo deseas contar los alumnos de 1A, la consulta quedaría del siguiente modo:

```
SELECT Cursos.Curso, COUNT (Alumnos.Expediente)  
FROM Alumnos, Cursos  
WHERE Alumnos.Grupo=Cursos.IdCurso  
GROUP BY Cursos.Curso HAVING Cursos.Curso = '1A'
```

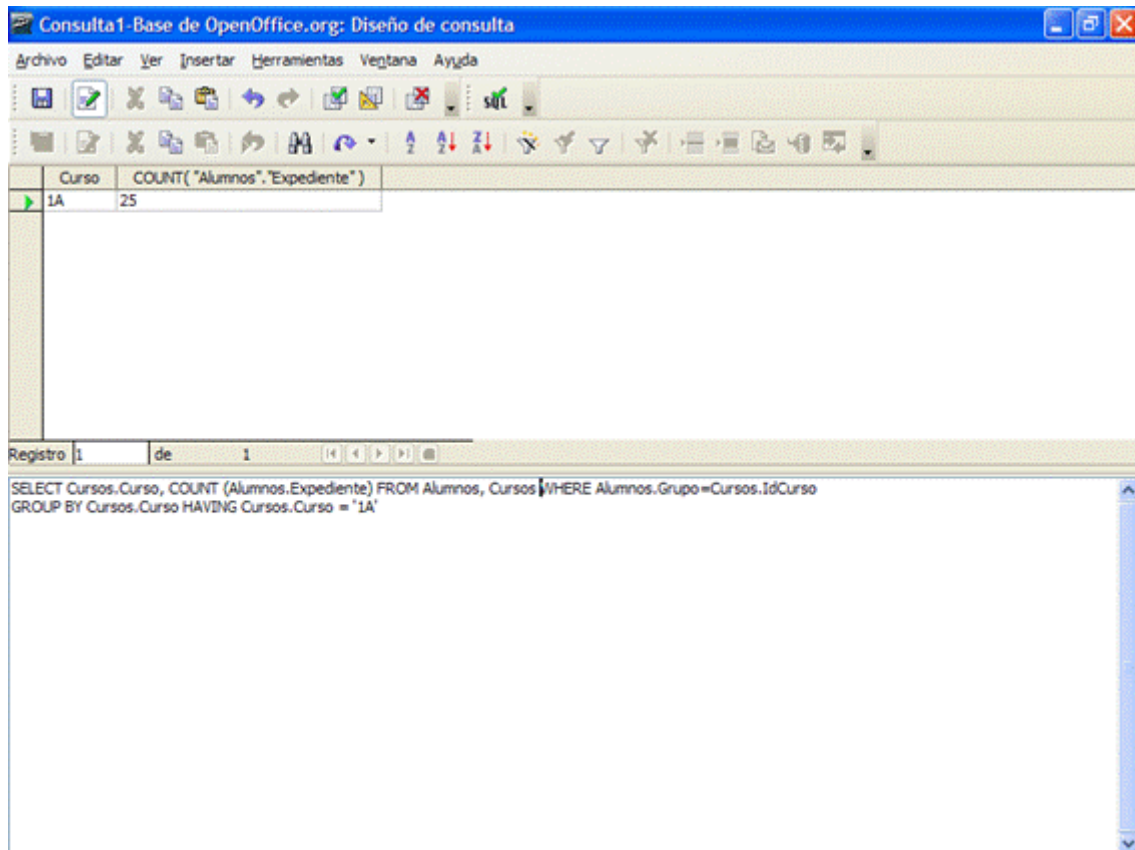


Figura 6.17

Si lo deseas puedes complicar los criterios añadiendo nuevos filtrados mediante los operadores AND y OR. La siguiente consulta es idéntica a la anterior pero obtiene el total de alumnos de los grupos 1A y 1B.

```
SELECT Cursos.Curso, COUNT(Alumnos.Expediente)  
FROM Alumnos, Cursos  
WHERE Alumnos.Grupo=Cursos.IdCurso  
GROUP BY Cursos.Curso HAVING Cursos.Curso = '1A'  
OR Cursos.Curso = '1B'
```

Es posible que llegados a este punto te estés planteando, por qué debes complicarte tanto la vida con SQL si lo puedes hacer de forma mucho más sencilla con el diseñador de consultas. La explicación la encuentras por una parte en las limitaciones que aún presenta OpenOffice Base y que únicamente se pueden suplir mediante sentencias SQL. También es necesario tener ciertas nociones de SQL ya que al tratarse de un lenguaje universal y estándar puedes utilizar las consultas en cualquier otra base de datos.

Eliminar registros, comando DELETE

Como ya se ha comentado, en la versión actual OpenOffice no se permite diseñar visualmente consultas de eliminación, inserción o actualización. Por lo tanto, para realizar este tipo de tareas en la base de datos debes recurrir a SQL.

La sentencia DELETE permite eliminar registros de una tabla. Esta es sin duda una operación delicada por lo que debes estar muy seguro de lo que haces pues una vez ejecutada no existe la posibilidad de deshacer la operación.

La recomendación en este sentido es que en primer lugar realices una consulta de selección para comprobar que estás seleccionando realmente los datos que deseas eliminar.

Los comandos DELETE, INSERT y UPDATE no se pueden ejecutar del modo que lo has estado haciendo hasta ahora con las consultas (SELECT). OpenOffice Base no permite hacerlo, pero sí incluye una herramienta que ofrece la posibilidad de ejecutar cualquier sentencia SQL.

1. En la ventana principal del programa, en la parte superior, haz clic sobre el menú **Herramientas**.
2. A continuación, selecciona el comando **SQL** para tener acceso al cuadro de diálogo que muestra la figura 6.18.



Figura 6.18

En esta ventana encuentras en primer lugar el cuadro denominado **Comando a ejecutar**. Será aquí donde debes escribir la sentencia SQL que deseas aplicar sobre la base de datos. Bajo este espacio se encuentra el botón Ejecutar que tendrás que utilizar para llevar a cabo las operaciones indicadas en el comando SQL.

A continuación dispones de una lista desplegable con las últimas sentencias SQL ejecutadas, una especie de histórico que puede servir para no tener que escribir varias veces la misma

instrucción si ya la has utilizado con anterioridad. Observa este elemento resaltado en la figura 6.19.

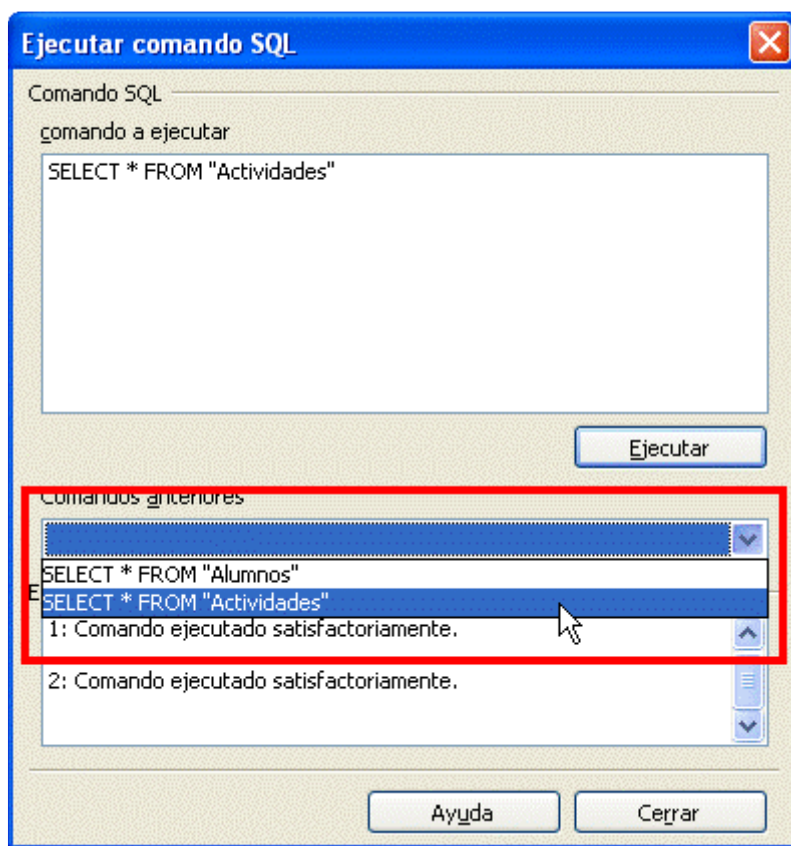


Figura 6.19

Por último, la sección **Estado** (figura 6.20) mostrará todos los mensajes relacionados con la ejecución de la sentencia SQL.

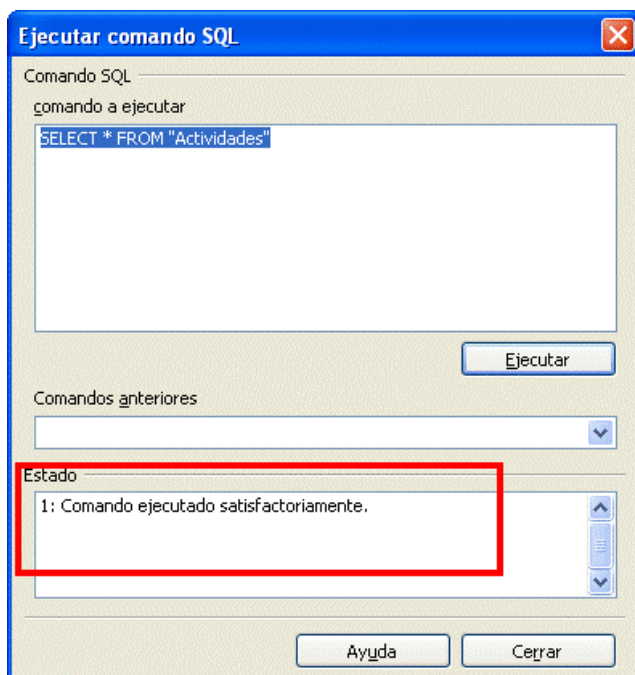


Figura 6.20

Sintaxis de las sentencias en el intérprete SQL

El nivel de exigencia del intérprete SQL en cuanto a la sintaxis de las instrucciones es muy alto y necesitas actuar de modo algo distinto a lo que has hecho hasta ahora. Algunos datos importantes que debes tener en cuenta:

Los nombres de tablas y campos deben estar entre comillas dobles. Por ejemplo:

```
SELECT "Nombre", "Apellidos" FROM "Alumnos"
```

En cuanto a las condiciones, los literales deben estar incluidos entre comillas simples y cada una de las condiciones encerradas entre paréntesis. Por ejemplo:

```
SELECT "Nombre", "Apellidos" FROM "Alumnos"  
WHERE ("Nombre" = 'Antonio')
```

El término *literal* dentro del diseño de consultas mediante lenguaje SQL, hace referencia al texto de una condición, en el ejemplo anterior *Antonio* sería un literal.

Si la condición fuera una fecha no habría demasiados cambios:

```
SELECT "Nombre", "Apellidos" FROM "Alumnos"  
WHERE ("FechaNacimiento" = '2008-12-01')
```

Cuando hay varias condiciones es conveniente encerrar todo el contenido de la cláusula WHERE entre paréntesis:

```
SELECT "Nombre", "Apellidos" FROM "Alumnos"  
WHERE (("FechaNacimiento" = '2008-12-01')  
AND ("FechaNacimiento" = '2008-12-31'))
```

Es esencial que sigas al pie de la letra la sintaxis requerida por el intérprete de SQL incluido en OpenOffice; de lo contrario un mensaje indicará que algo no está bien.

Ejemplo de eliminación de registros

A continuación un ejemplo del proceso completo de eliminación de registros con OpenOffice Base. Eso sí, aplicaremos el principio de prudencia, es decir, crearemos en primer lugar una consulta de selección estándar y la convertiremos después en una consulta de eliminación. De esta forma podemos comprobar si los registros que serán eliminados corresponden exactamente con los que deseamos borrar de la base de datos. La idea es eliminar los alumnos cuya fecha de nacimiento sea inferior a 31/12/1990. El resultado lo debes ordenar alfabéticamente. Empecemos por diseñar la consulta.

Actividad 11

1. Selecciona la opción **Crear consulta en vista SQL** y empieza escribiendo: **SELECT**.
2. A continuación indica los campos de la tabla que desees mostrar en la consulta, en este caso utilizarás los campos Nombre y Apellidos y FechaNacimiento de la tabla Alumnos.
3. Ahora añade la cláusula FROM y escribe el nombre de la única tabla implicada: Alumnos.
4. Ahora añade la sentencia WHERE y escribe el primer criterio de filtrado: FechaNacimiento <= '1990-12-31'.
5. Finalmente para ordenar el resultado, añade ORDER BY Apellidos.
6. La instrucción SQL queda del siguiente modo:
SELECT Nombre, Apellidos, FechaNacimiento FROM Alumnos WHERE FechaNacimiento <= '1990-12-31' ORDER BY Apellidos

Observa el aspecto de la ventana diseño en la figura 6.21.

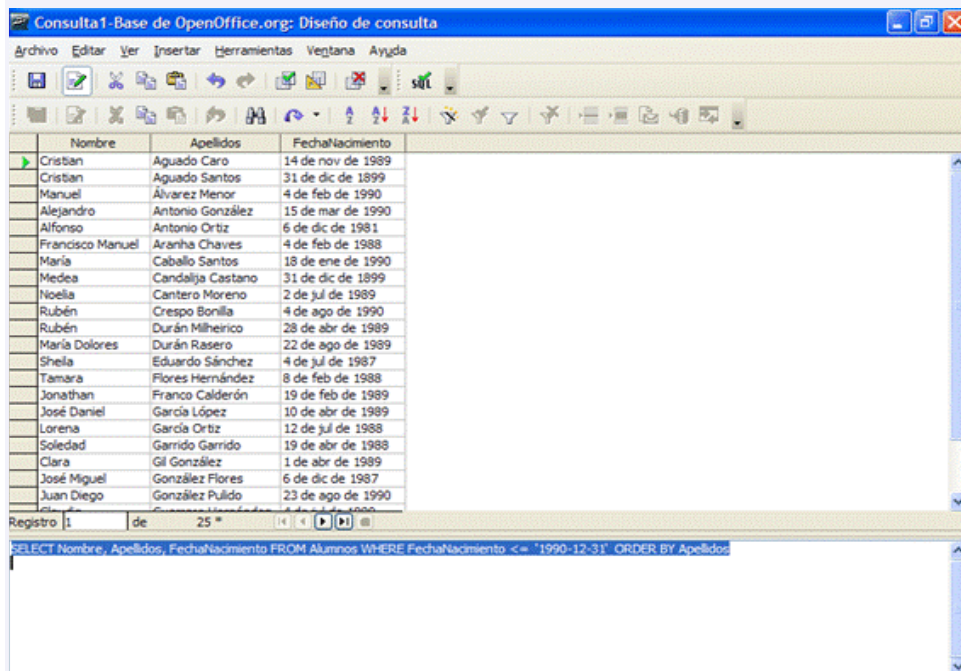


Figura 6.21

7. Ejecuta la consulta y comprueba que los resultados mostrados corresponden realmente con la información que deseas eliminar de la base de datos.
8. Una vez que estés completamente seguro de que la selección es correcta, un pequeño truco para ahorrar algo de trabajo.
9. Selecciona la instrucción SQL completa y después utiliza el comando **Copiar**, ya sea con la combinación de teclas **Control+C** o desde el menú **Editar**.
10. Cierra la ventana de consulta, guarda el resultado si lo crees conveniente y vuelve a la ventana principal de la base de datos.
11. En el menú **Herramientas** selecciona el comando **SQL** como muestra la figura 6.22.

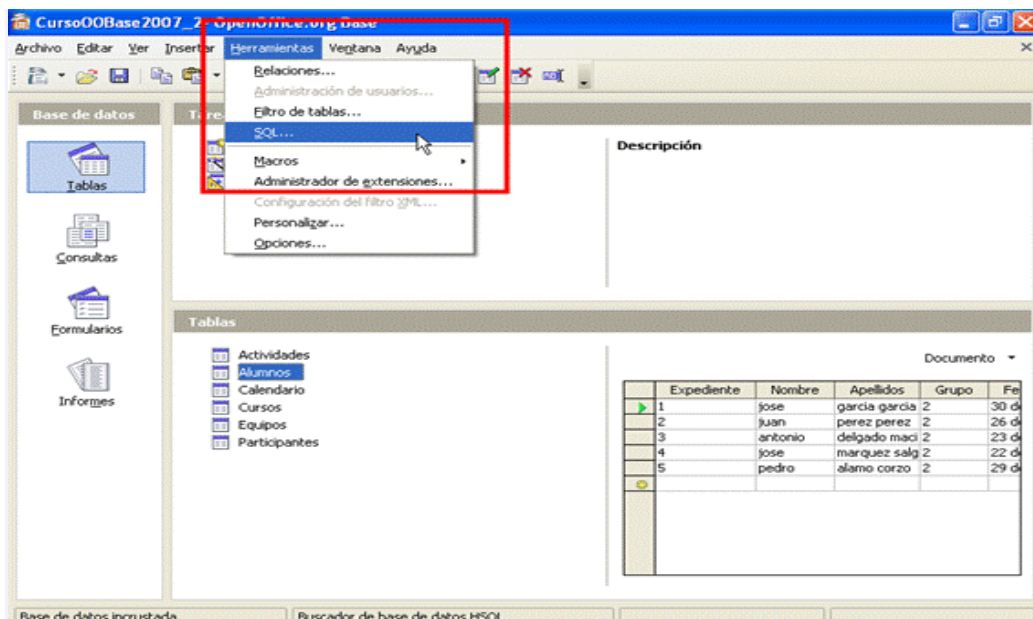


Figura 6.22

12. Haz clic dentro de la primera sección del cuadro de diálogo y ejecuta el comando Pegar (Control+V). Al instante aparece la instrucción de selección.
13. Pero si lo que deseas es eliminar registros y no seleccionarlos, debes hacer algunos cambios. Concretamente debes eliminar el comando SELECT y cambiarlo por DELETE. Además elimina los nombres de campos y la cláusula ORDER BY. Una vez hechos todos estos cambios la sentencia quedaría de la siguiente forma:

DELETE FROM "Alumnos" WHERE ("FechaNacimiento" <= '1990-12-31')

14. Si has seguido todos los pasos correctamente, el resultado de ejecutar la sentencia anterior será un error. Bueno, en realidad no se trata de un error, si no más bien de un sistema de protección de la base de datos. Las reglas de integridad referencial de la base de datos no te deja llevar a cabo este proceso para no dejar registros "colgados".
15. Para solucionarlo abre la ventana de relaciones y haz clic con el botón derecho sobre la línea que une la tabla alumnos con todas aquellas tablas donde se han utilizado datos de esta tabla.
16. Selecciona el comando Editar para que Base muestre el cuadro de diálogo Relaciones. En la parte inferior derecha tienes la sección Opciones de eliminación, pues bien aquí puedes controlar comportamiento del programa cuando se intenta eliminar un registro que tiene vínculos en otras tablas. Si seleccionas la opción Eliminar en cascada ya no tendría que aparecer el error y el alumno desaparecerá de todas aquellas tablas donde se encontraba.

En la parte inferior del cuadro de diálogo **Ejecutar comandos SQL** aparece un mensaje indicando que todo ha sido correcto como puedes ver en la figura 6.23.

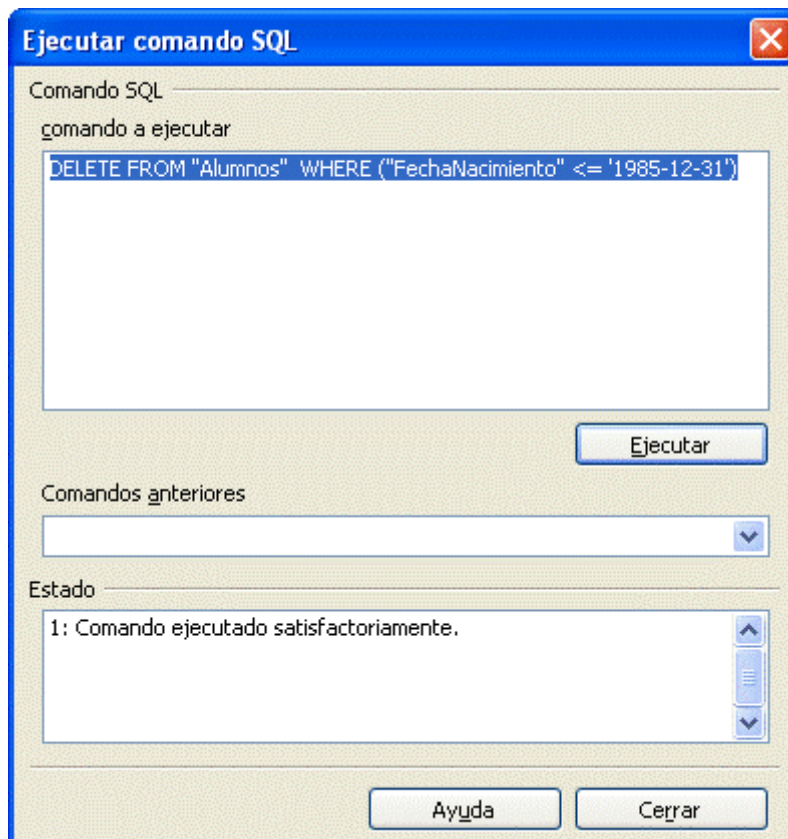


Figura 6.23

Si no has eliminado la consulta diseñada anteriormente, para comprobar los datos antes de eliminarlos puedes ejecutarla de nuevo y si todo es correcto no debería devolver ningún registro.

Antes de terminar con este apartado, otro pequeño truco. Si quieres conocer la sentencia SQL asociada a cualquier consulta mientras trabajas en la vista Diseño, ejecuta el comando **Ver** y después utiliza **Activar o desactivar la vista Diseño** como puedes ver en la figura 6.24. Para recuperar el aspecto inicial de la ventana, repite la misma operación.

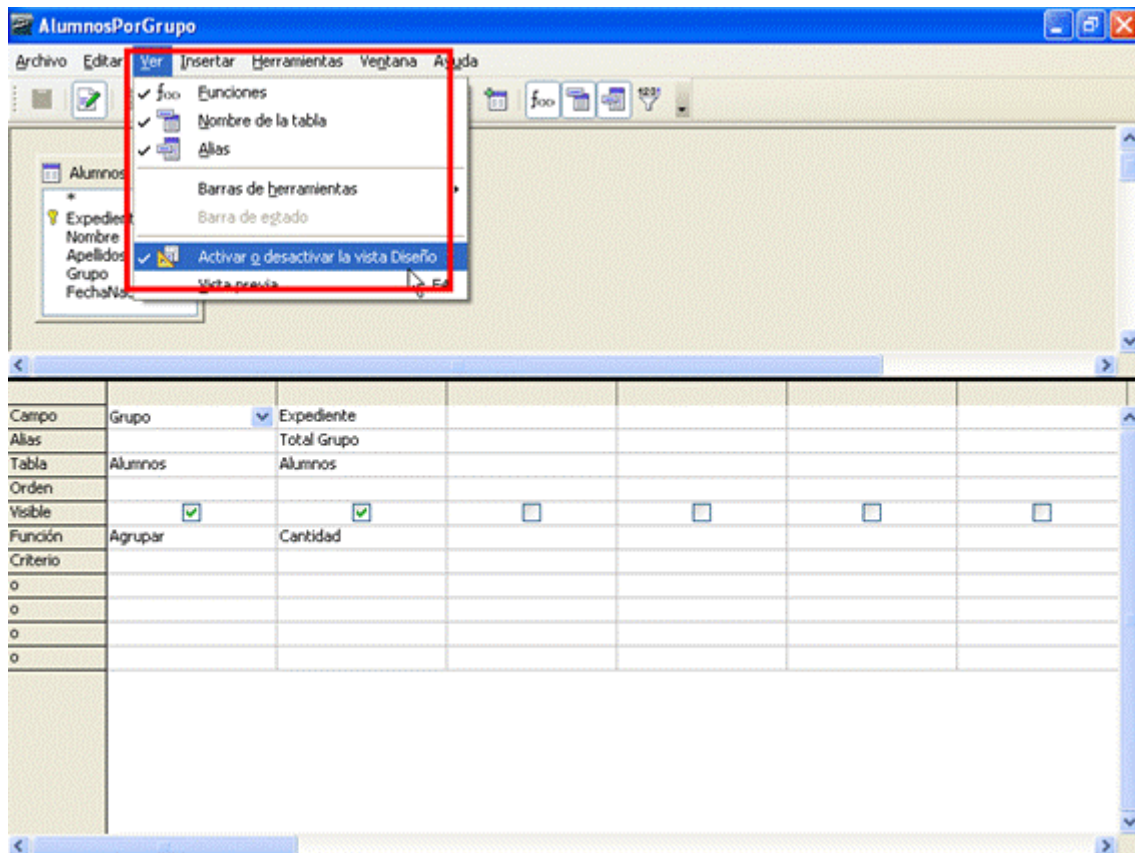


Figura 6.24

Insertar nuevos registros, sentencia INSERT

El comando SQL de inserción de registros puede ser tremendamente útil a la hora de llenar tablas cuya información obtienes mediante otra fuente de datos como otras tablas o consultas.

Un ejemplo sencillo: cómo añadir un registro a la tabla *Actividades* que únicamente tiene un campo ya que el identificador es autonumérico y por lo tanto, se genera automáticamente al añadir un nuevo registro:

Actividad 12

1. En la ventana principal de la base de datos haz clic sobre el menú **Herramientas** y después ejecuta el comando **SQL**.
2. En la primera sección del cuadro de diálogo escribe la siguiente instrucción: **INSERT INTO "Actividades" ("Actividad") VALUES ('Voleibol')**
3. Después de escribir la sentencia SQL, haz clic en el botón **Ejecutar**.
4. Si no existe ningún problema aparecerá en la parte inferior un mensaje indicando que el comando se ha ejecutado correctamente.
5. Revisa la tabla *Actividades* y comprueba que se ha añadido el registro.

Así planteado quizás no te parezca demasiado interesante el comando INSERT pero, ¿qué te parecería añadir una consulta a esta misma sentencia y que pudieras incluir en la tabla seleccionada el resultado de la consulta? Por ejemplo, imagina que has obtenido una tabla de deportes (*MisDeportes*) y quieres utilizarla para rellenar la tabla de actividades deportivas (*Actividades*).

La instrucción SQL debería tener el siguiente aspecto:

```
INSERT INTO "Actividades" ("Actividad") SELECT "NombreDeporte" FROM "MisDeportes"
```

Tras haber ejecutado lo anterior, comprueba que efectivamente la tabla *Actividades*, que anteriormente sólo tenía cuatro registros, tiene ahora también los registros de la tabla *MisDeportes*.

Por supuesto, a la consulta que sirve para añadir registros puedes añadirle tantas condiciones y cláusulas como necesites. Por ejemplo, puedes elegir todos aquellos deportes que no sean de tipo individual (Tenis, Atletismo...)

```
INSERT INTO "Actividades" ("Actividad") SELECT "NombreDeporte" FROM "MisDeportes" WHERE ("Tipo" = 'Individual')
```

Lo más importante en este tipo de consultas es que el número de campos que devuelve la consulta sea el mismo que intentas insertar. También es fundamental que los tipos de los campos coincidan.

Actualizar registros, sentencia UPDATE

El concepto de actualización consiste en modificar algunos de los datos que contiene el registro. Por ejemplo, imagina que necesitamos cambiar la dirección de un alumno concreto o el grupo al que pertenece. Este tipo de operaciones pueden realizarse mediante actualizaciones y más concretamente con el comando UPDATE.

Con las operaciones de actualización mediante el comando UPDATE deberías tener las mismas precauciones que ya viste para el comando DELETE. El motivo es el mismo, es decir, las modificaciones realizadas por la consulta no tienen posibilidad de deshacerse. En este sentido, se recomienda de nuevo ejecutar en primer lugar una consulta de selección para comprobar los registros que se verán afectados y una vez comprobado que todo es correcto llevar a cabo la actualización.

OpenOffice no dispone de ninguna herramienta específica para realizar cualquier consulta de actualización sobre la base de datos por lo que igual que ocurría en el apartado anterior debes recurrir al intérprete de consultas. Desde la ventana principal de OpenOffice Base, haz clic en el menú **Herramientas** y después selecciona el comando **SQL**.

Observa en el siguiente ejemplo cómo cambiar el nombre del departamento “Geografía” por “Geografía e Historia”

1. En la ventana principal de la base de datos haz clic sobre el menú **Herramientas** y después ejecuta el comando **SQL**.
2. En la primera sección del cuadro de diálogo, escribe la siguiente instrucción:
3. `UPDATE "Departamentos" SET "Nombre" = 'Geografía e Historia' WHERE "Nombre" = 'Geografía'`
4. Después de escribir la sentencia SQL haz clic en el botón **Ejecutar**.
5. Si no existe ningún problema aparecerá en la parte inferior un mensaje indicando que el comando se ha ejecutado correctamente.
6. Revisa la tabla *Departamentos* y comprueba que se ha modificado el registro tal y como lo hemos indicado en la sentencia SQL.

Debes tener cuidado si vas a copiar las instrucciones directamente desde esta página y pegarlas en la ventana SQL de OpenOffice Base. Revisa las comillas tanto dobles como simples.