Manual de lenguaje SQL aplicado a la base de datos de Varilex

Antonio Ruiz Tinoco
a-ruiz@hoffman.cc.sophia.ac.jp
Universidad Sophia, Tokio

0. Introducción

El proyecto Varilex lleva ya siete años en marcha desde la primera presentación en el X Congreso de ALFAL en Veracruz el año 1993. Desde entonces, se han venido reuniendo los datos enviados por colaboradores de todos los países hispanohablantes. La abrumadora cantidad de datos nos hizo considerar una base de datos digitalizada a través de internet, que como anunciamos en Varilex 7 y posteriormente en el XII Congreso de ALFAL en Santiago de Chile, empezó a funcionar parcialmente en el verano de 1999.

Para llevar a cabo nuestra tarea, instalamos un servidor en http://varilex.call.sophia.ac.jp, con el sistema operativo LINUX para PC (kernel 2.2), un programa de administración de base de datos de lenguaje SQL, MySQL ver. 3.22.15, junto con PHP3 ver. 3.0.6. Después de considerar varias posibilidades, nos decidimos por este sistema principalmente por su estabilidad, la posibilidad de utilizar los caracteres propios de la lengua española, la facilidad de uso, y, cómo no, por ser gratuitos. No obstante, para poderle sacar un mayor rendimiento a nuestra base compartida de variación léxica, hemos creído útil la confección de un manual simple dirigido a las personas interesadas en los datos pero que no conocen los detalles del lenguaje SQL y de esta manera animarlas a usar esta poderosa herramienta de investigación.

SQL es la abreviatura de *Structure Query Language*, un lenguaje desarrollado para administrar bases de datos relacionales. Existen muchas variantes de este lenguaje, pero el programa que hemos escogido, MySQL sigue en general las normas SQL-92 y el nivel 0-2 ODBC, *Open Database Connectivity* de Microsoft, lo cual será de gran utilidad para los usuarios debido a la compatibilidad que nos ofrece con otros muchos sistemas y programas de administración de bases de datos. Para ampliar información sobre estos sistemas,

recomendamos una lista de publicaciones en las referencias y enlaces a través de internet al final de este manual.

Empezaremos con la presentación del formato de los datos sin extendernos demasiado y nos concentraremos en los comandos relacionados con el sistema de consulta, especialmente el comando SELECT, su estructura y ejemplos de uso con datos reales de Varilex.

1. Estructura de los datos

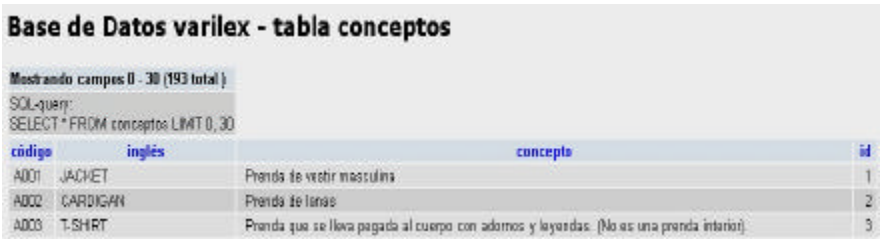
Se puede entrar en la base de datos a través de la página de entrada del proyecto en http://varilex.call.sophia.ac.jp, y siguiendo después con el enlace **Base de Datos**. Esta página, además de permitir la consulta de la estructura de los datos, dispone de una ventana, como se muestra en la Fig. 1, que nos será muy útil tanto para la práctica del lenguaje SQL como para la obtención de los datos léxicos que nos interesen.

Como se puede ver en la Fig. 1, la base de datos de Varilex consta de 4 tablas principales que se pueden consultar simplemente pulsando los enlaces a su derecha: **Examinar** y **Seleccionar**. También hay un enlace al manual general de MySQL en inglés, técnicamente muy detallado, pero no muy apropiado como introducción.

Fig. 1 ventana para comandos SQL

Para practicar las indicaciones que damos en este manual, aconsejamos la utilización de esta ventana. Los comandos se irán escribiendo dentro en la forma indicada y al terminar de introducirlos se pulsará el botón **Ejecutar** a la derecha. De esta manera podremos efectuar consultas más complejas según las necesidades de cada investigador.

Cada una de estas tablas tiene la estructura que se puede apreciar pulsando el enlace **Examinar** y que comentamos brevemente a continuación. En las figuras siguientes se pueden ver las primeras líneas del resultado, suficientes para nuestro propósito.



Base de Datos varilex - tabla conceptos

Mostrando campos 0 - 30 (193 total)

SQL-query:
SELECT * FROM conceptos LIMIT 0, 30

código	inglés	concepto	id
A001	JACKET	Prenda de vestir masculina	1
A002	CARDIGAN	Prenda de lanas	2
A003	T-SHIRT	Prenda que se lleva pegada al cuerpo con adornos y leyendas. (No es una prenda interior)	3

Fig. 2 tabla conceptos

Tabla conceptos (Fig. 2)

- **código**: tiene la forma de una letra que indica el orden de la encuesta, y tres dígitos para el orden del concepto utilizado.
- **inglés**: queremos aclarar que la palabra en inglés que incluimos en este campo no es una traducción de los vocablos utilizados. Es solamente una referencia útil y neutra para representar lo que llamamos *ámbito conceptual*[1], y al mismo tiempo evitamos utilizar cualquiera de las variantes léxicas de la lengua española.
- **concepto**: descripción del *ámbito conceptual*, para el que se buscan variaciones léxicas.
- **id:** número de orden para las entradas.

[1] *ámbito conceptual*: el proyecto Varilex no colecciona sinónimos en el sentido tradicional del término, sino variaciones léxicas que responden a su descripción verbal, y a veces visual, tal como se hace en las encuestas.

Base de Datos varilex - tabla contestaciones

Mostrando campos 0 - 30 (45148 total)

SQL-query:
SELECT * FROM contestaciones LIMIT 0, 30

orden	codciu	país	ciudad	informante	forma	respuesta
A001	2	ES	SCO	e155	1	americana
A001	2	ES	SCO	e156	1	americana
A001	2	ES	SCO	e161	1	americana

Fig. 3 tabla de contestaciones

Tabla de contestaciones (Fig.3)

- **orden:** coincide con el campo **código** de la tabla **conceptos**. Es necesario para hacer búsquedas complejas con dos o más tablas.
- **codciu:** número de código de la ciudad donde se realizó la encuesta.
- **país:** código de dos letras del país donde se realizó la encuesta. Existe un proyecto de incluir Filipinas, pero en la actualidad solamente tenemos datos de los países siguientes:
-
- AR Argentina
  BO Bolivia
  CH Chile
  CO Colombia
  MA Marruecos
  CR Costa Rica
  CU Cuba
  EC Ecuador
  ES España
  EL El Salvador
  EU EE.UU.
  GE Guinea Ecuatorial
  GU Guatemala
  HO Honduras
  MX México

NI	Nicaragua
PA	Paraguay
PE	Perú
PN	Panamá
PR	Puerto Rico
RD	Rep. Dominicana
UR	Uruguay
VE	Venezuela

ciudad: código de tres letras de la ciudad donde se realizó la encuesta.
Se puede ver la lista completa en cualquiera de las publicaciones de Varilex o en su página web. Posiblemente en el futuro añadiremos una tabla de ciudades que hará más ágil la búsqueda compleja de datos.

- **informante**: código de referencia del informante, que coincide con el campo **código** de la tabla **informantes.**
- **forma**: código identificador de formas
- **respuesta**: respuesta obtenida en la encuesta a esa pregunta.

Fig. 4 tabla de cuadros

Tabla cuadros (Fig. 4)

- **orden:** la parte de la izquierda coincide con el campo **orden** de la tabla **contestaciones** y con el campo **código** de la tabla **conceptos**. Separado hay un número que indica el número de la variación léxica ofrecida en la encuesta para ese concepto.

Es de utilidad técnica para búsquedas complejas, como veremos más adelante.

- **entrada**: variación léxica ofrecida en la encuesta para el concepto indicado en la parte izquierda del campo **orden.**

Base de Datos varilex - tabla informantes

Mostrando campos 0 - 30 (177 total)

SQL-query:
SELECT * FROM informantes LIMIT 0, 30

código	codciu	país	ciudad	sexo	edad	tipo	ocupación	id
e001	39	EL	SSA	1	33	2	Psicólogo	1
e002	39	EL	SSA	2	33	2	Máster en Educación	2
e003	39	EL	SSA	1	40	2	Abogado	3

Fig. 5 tabla de informantes

Tabla de informantes (Fig. 5)

- **código**: código de referencia del informante, que coincide con el campo **informante** de la tabla de **contestaciones**
- **codciu**: código de la ciudad
- **país**: código del país. Coincide con el anteriormente explicado
- **ciudad**: código de la ciudad. Coincide con el anteriormente explicado
- **sexo**: sexo del informante: 1 es para mujer y 2 para hombre
- **edad**: edad del informante
- **tipo**: tipo de ocupación del informante
- **ocupación**: descripción de la ocupación del informante
- **id**: número de orden dentro de la tabla

Sin duda alguna la estructura de la base de datos se puede modificar y mejorar. Sin embargo, creemos que, a pesar de haber dejado alguna redundancia, resulta más cómoda de manejar para los principiantes con el único sacrificio de un poco más de espacio en el disco duro. Tal vez en el futuro, y según las sugerencias recibidas, se modifique la estructura actual. En cualquier caso, la utilidad de este manual no variará, ya que sólo trataremos nociones generales de uso.

2. Técnicas de búsqueda

2.1. Cláusula SELECT {campo} FROM {tabla} WHERE {condición}

Para la búsqueda de datos ordenados utilizaremos el potente comando o cláusula de selección **SELECT**. Escribiremos con mayúsculas los comandos básicos y con *minúscula cursiva* los parámetros que se deben incluir y que varían según el objetivo de nuestra búsqueda. En realidad no es necesario escribir los datos con minúscula ni en cursiva. Una barra vertical | indicará que se debe escoger una de las opciones, los corchetes [ ] contienen elementos opcionales y las llaves { } contienen elementos obligatorios. Vamos a ver ejemplos concretos de búsqueda. Las hay muy simples como SELECT 3 + 4, que nos dará como resultado 7, la suma de los dos dígitos; pero para nosotros, que nos importa el tratamiento del léxico, la fórmula de búsqueda más simple y frecuente tendrá la estructura siguiente:

SELECT *inglés, concepto* FROM *conceptos* WHERE *inglés* LIKE "*jacket*"

La fórmula de arriba significa que queremos hacer una lista con los datos de los campos *inglés* y *concepto*, por este orden, de la tabla de *conceptos*, seleccionada por la cláusula FROM, y que cumpla la condición expresada en la cláusula WHERE, de que contenga la secuencia "jacket" en el campo *inglés*. La cláusula WHERE va seguida de una condición para estrechar los límites de la búsqueda. Normalmente el lenguaje SQL requiere que se termine con un punto y coma (;) la fórmula, pero en la base de datos de Varilex no es necesario. El resultado será algo similar a la Fig. 6.

Base de Datos varilex

inglés	concepto
JACKET	Prenda de vestir masculina

Fig. 6

Para obtener todos los campos de la tabla, es decir, *código*, *inglés*, *concepto* e *id*, utilizamos un asterisco (*), que funciona como comodín y significa "todos los campos", como veremos en el apartado siguiente.

2.2. Cláusula LIKE, operadores y comodines

En lugar de la expresión LIKE se puede utilizar simplemente el signo de igualdad (=); pero preferimos LIKE ya que admite comodines como veremos a continuación:

SELECT * FROM *conceptos* WHERE *inglés* LIKE "*jacket*"

Supongamos ahora que queremos ver todos los datos correspondientes a la entrada A003 en la tabla de contestaciones. Será suficiente la siguiente fórmula:

SELECT * FROM *contestaciones* WHERE *orden* LIKE "A003"

Las primeras líneas del resultado se muestran en la Fig. 7:

Mostrando campos - 30 (total)

SQL-query:
SELECT * FROM contestaciones WHERE orden LIKE "A003"

orden	codciu	país	ciudad	informante	forma	respuesta
A003	1	ES	COR	e009	2	camiseta
A003	1	ES	COR	e010	2	camiseta
A003	1	ES	COR	e011	2	camiseta

Fig. 7

También es posible utilizar números y expresiones matemáticas, muy útiles para el tratamiento estadístico de los datos. Veamos algunas expresiones simples. Si queremos una lista de los datos de los informantes mayores de 32 años, mujeres, la expresión de búsqueda y el resultado serán:

SELECT * FROM *informantes* WHERE *edad* > 32 AND *sexo* = 2

Mostrando campos - 30 (total)

SQL-query:
SELECT * FROM informantes WHERE edad > 32 AND sexo = 2

código	codciu	país	ciudad	sexo	edad	tipo	ocupación	id
e002	39	EL	SSA	2	33	2	Máster en Educación	2
e004	39	EL	SSA	2	60	2	Maestra	4
e008	5	ES	SLM	2	71	3	Ama de casa	8

Fig. 8

Si queremos los datos de los informantes entre 25 y 48 años, mujeres, y procedentes de España, necesitaremos una expresión algo más compleja. La expresión será:

SELECT * FROM *informantes* WHERE *país* LIKE “ES”
AND *edad* BETWEEN 25 AND 48 AND *sexo* = 2

Mostrando campos - 30 (total)

SQL-query:
SELECT * FROM informantes WHERE país LIKE "ES"
AND edad BETWEEN 25 AND 48 AND sexo = 2

código	codciu	país	ciudad	sexo	edad	tipo	ocupación	id
e005	5	ES	SLM	2	29	2	Funcionaria	5
e011	1	ES	COR	2	28	2	Telefonista	11
e019	17	ES	PAL	2	29	2	Profesora de universidad	19

Fig. 9

Hemos unido la segunda condición por medio del operador AND y de nuevo hemos hecho lo mismo con la tercera.

Para buscar los datos de palabras que terminan en “–ero”, o que empiezan por “des-“ o que contienen la secuencia “-bla-“ podemos utilizar el comodín %, que significa una secuencia de cero o más caracteres, incluidos los signos de puntuación. El otro símbolo comodín _ significa un solo carácter, como en los ejemplos siguientes:

SELECT * FROM *contestaciones* WHERE *respuesta* LIKE “%ero”
SELECT * FROM *contestaciones* WHERE *respuesta* LIKE “des%”

SELECT * FROM *contestaciones* WHERE *respuesta* LIKE "%bla%"
SELECT * FROM *contestaciones* WHERE *respuesta* LIKE "meser_"

La última fórmula nos seleccionará respuestas que incluirían términos como "mesero" y "mesera". La negación de la condición se puede expresar también como *respuesta* NOT LIKE "%bla%", o bien con fórmulas del tipo *respuesta* <> "%bla%", o *respuesta* != "%bla%". En lugar de AND se puede utilizar && alternativamente. También podemos utilizar la expresión OR, o || (dos barras verticales), como en el siguiente ejemplo, con el que queremos seleccionar datos que contengan "chaqueta" o "chaquetón":

SELECT * FROM *contestaciones*
WHERE *respuesta* LIKE "chaqueta" OR respuesta LIKE "chaquetón"

2.3 Cláusula LIMIT

Con frecuencia el resultado de nuestra búsqueda podría ocupar un gran espacio y no cabría en la pantalla o no se podría imprimir en una página. En caso de querer limitar la búsqueda podemos añadir al final la cláusula LIMIT.

LIMIT [*x*,] *y* (Ejemplo: LIMIT 0, 30)

Esta útil fórmula solamente significa que se deben mostrar los datos según el orden indicado. En el ejemplo de arriba, la búsqueda muestra los 30 primeros resultados. Para ver los siguientes 30 resultados sería suficiente cambiar los parámetros de la cláusula a LIMIT 31, 60, y así sucesivamente.

2.4. Cláusula DISTINCT

Con frecuencia, nos encontraremos con una lista en la que los datos se repiten numerosas veces. Si lo que queremos es una lista que contenga fichas con datos distintos podemos indicarlo así en la fórmula de búsqueda, como en el ejemplo siguiente, en el que

queremos saber simplemente las respuestas diferentes obtenidas para la entrada A003 (T-shirt) en Cuba.

SELECT DISTINCT *orden, respuesta*
FROM *contestaciones*
WHERE *orden* LIKE “A003” AND *país* LIKE “CU”

Por la respuesta mostrada en la Fig. 10, sabemos que entre los datos de Cuba, el concepto A003, T-SHIRT, sólo se encuentra la variación de “pulóver”.

orden	respuesta
A003	pulóver

Fig. 10

Si no hubiéramos empleado la cláusula DISTINCT, la respuesta habría sido la misma pero el tiempo de búsqueda hubiera sido mayor y el resultado práctico el mismo, según se puede apreciar en la Fig. 11 a continuación.

Mostrando campos - 30 (total)

SQL-query:
SELECT orden, respuesta
FROM contestaciones
WHERE orden LIKE "A003" AND país LIKE "CU"

orden	respuesta
A003	pulóver
A003	pulóver
A003	pulóver
A003	pulóver
A003	pulóver
A003	pulóver
A003	pulóver
A003	pulóver

Fig. 11

Por supuesto, el resultado de la búsqueda, además de verse sobre el monitor, se puede imprimir o guardar utilizando las

funciones normales de cualquier navegador, como Netscape o Internet Explorer. Para los usuarios que quieran utilizar su propio programa de administración de bases de datos, como Access, Excel, FileMaker Pro, etc., hemos preparado los datos en formato CSV (comma separated value) que se pueden bajar en la página **Download** de Varilex. Por el momento está disponible una parte de los datos correspondientes a cada tabla por separado o todas las tablas en formato comprimido zip.

2.5. Búsquedas utilizando dos o más tablas.

Hasta ahora hemos visto algunas fórmulas simples para búsquedas dentro de una misma tabla. Sin embargo, lo más interesante de las bases de datos relacionales es la posibilidad de utilizar diferentes tablas al mismo tiempo. Para ello es suficiente que compartan uno de los campos. Vamos a ver algunos ejemplos.

2.5.a. Confección de una lista de variación léxica para una zona determinada.

Supongamos que necesitamos una lista con todos los términos aparecidos en las encuestas de Cuba con la palabra en inglés que representa el concepto a modo de un diccionario de variaciones léxicas cubanas. Comprobamos que el campo *inglés* se encuentra en la tabla *conceptos* y que los campos *país* y *respuesta* se encuentran en la tabla *contestaciones*. Para ello haremos simplemente lo siguiente:

SELECT DISTINCT *inglés, respuesta*
FROM *conceptos, contestaciones*
WHERE *código = orden*
AND *país* LIKE “CU”
LIMIT 0,30

Mostrando campos - 30 (total)

SQL-query:

```
SELECT DISTINCT inglés, respuesta
FROM conceptos,contestaciones
WHERE código = orden
AND país LIKE "CU"
LIMIT 0,30
```

inglés	respuesta
JACKET	americana
JACKET	chaqueta
JACKET	saco
JACKET	saco de traje
JACKET	traje
CARDIGAN	abrigo
CARDIGAN	cárdigan
CARDIGAN	jersey
CARDIGAN	suéter
T-SHIRT	pulóver
SWEATER	enguatada
SWEATER	suéter
WINDBREAKER	abrigo

Fig. 12

Hemos dividido la expresión de búsqueda en varias líneas para comodidad en la explicación aunque no es necesario. La primera línea pide a la base de datos que haga una lista con los campos *inglés* y *respuesta*, que se pueden obtener de las tablas expresadas en la segunda línea en la cláusula FROM, es decir, *conceptos* y *contestaciones*. Como condición principal, a tener en cuenta especialmente en este tipo de búsqueda a través de dos o más tablas, es la expresión WHERE *código = orden*, ya que son los campos que ponen en relación –de ahí su denominación de bases de datos relacionales-- las dos tablas. La otra condición añadida es que el campo, *país*, contenga "CU", es decir, Cuba. Como podemos ver, en lugar de CU podemos poner AR para Argentina, ES para España, o bien otras condiciones como

WHERE *ciudad* LIKE “SJU”
que nos confeccionaría el “diccionario de variación léxica” de San Juan de Puerto Rico. Por supuesto, el operador OR nos permite decidir la zona de la queremos obtener una lista. Supongamos que queremos una lista de variación léxica del español del Caribe. En tal caso, la cláusula WHERE sería como sigue:

WHERE *país* LIKE “CU” OR *país* LIKE “PR” OR *país* LIKE “RD”

2.5.b. ¿Qué variación léxica existe para “chaqueta”, el concepto A001, en una zona determinada?

Ahora el objetivo de nuestra búsqueda es una lista con las variaciones léxicas de una zona determinada. Para empezar, vamos a ver cómo preguntar por todas las variaciones de un concepto. Queremos que la lista contenga el código del concepto, su representación en inglés, las variaciones léxicas, la ciudad y el país. También queremos que las respuestas seleccionadas sean diferentes, ya que en nuestra pregunta no nos interesa por el momento la frecuencia de uso ni su distribución.

En primer lugar, si no lo sabemos, debemos averiguar cuál es el código del concepto para “chaqueta”. Podemos hacerlo de varias maneras. Por ejemplo, aquí vamos a utilizar la siguiente fórmula:

SELECT DISTINCT *orden, respuesta* FROM *contestaciones*
WHERE *respuesta* LIKE “chaqueta”

Esta pregunta, en principio, nos da las referencias a varios conceptos, A001, A002, A005 y A009 y decidimos, por ejemplo, buscar el concepto cuyo código es el A001. A continuación seleccionamos los campos que nos interesan en la cláusula SELECT y después añadimos las condiciones en la cláusula WHERE:

SELECT DISTINCT *código, inglés, respuesta, ciudad, país*
FROM *conceptos, contestaciones*
WHERE *código = orden*
AND *código* LIKE “A001”

Mostrando campos - 30 (total)

SQL-query:
SELECT DISTINCT código, inglés, respuesta, ciudad, país
FROM conceptos, contestaciones
WHERE código = orden
AND código LIKE "A001"

código	inglés	respuesta	ciudad	país
A001	JACKET	americana	SCO	ES
A001	JACKET	americana	OVI	ES
A001	JACKET	americana	SLM	ES
A001	JACKET	americana	BAR	ES
A001	JACKET	americana	GDL	ES
A001	JACKET	americana	ZAR	ES

Fig. 13

Para restringir la búsqueda a una zona determinada, cualquier conjunto de ciudades o países, resulta obvio que la única condición que tenemos que añadir es la descripción de la zona en cuestión. Por ejemplo, si queremos restringir la búsqueda a Perú y Ecuador, podemos añadir la siguiente condición:

AND *país* LIKE “PE” OR *país* LIKE “EC”

De esta manera, podremos saber fácilmente qué variaciones existen en Perú y Ecuador para lo que en otros lugares se suele expresar con “chaqueta”.

2.5.c. ¿Qué significa “chompa” y dónde se utiliza?

Como variante del tipo de búsqueda anterior podríamos pensar en el siguiente problema: buscar el significado del término “chompa” y su distribución geográfica. Como hemos hecho anteriormente en 2.5.b, en primer lugar buscamos en la tabla de contestaciones el código correspondiente a “chompa” si no queremos hacerlo manualmente.

SELECT DISTINCT *orden, respuesta* FROM *contestaciones*
WHERE *respuesta* LIKE “chompa”

A esta pregunta, la base de datos nos responde que “chompa” se encuentra en las respuestas a los conceptos A002, A004, A005 y A009. A continuación, hacemos una pregunta a la base de datos de forma similar a 2.5.b. y supongamos que esta vez nos interesan sólo los conceptos A005 y A009. La fórmula sería:

SELECT *código, respuesta, ciudad, país*
FROM *conceptos, contestaciones*
WHERE *código = orden*
AND *código* LIKE “A005” OR *código* LIKE “A009”

La respuesta obtenida se muestra en la Fig. 14 donde podemos ver las distribuciones por ciudades y países de las primeras respuestas obtenidas ya que no las mostramos todas aquí por motivos de espacio. Para comprobar el significado de “chompa” solamente habría que añadir el campo *concepto* a la cláusula SELECT, ya que es el que da la descripción del ámbito conceptual que buscamos, o bien consultar después los códigos A005 y A009.


Mostrando campos - 30 (total)
SQL-query:
SELECT código, respuesta, ciudad, país
FROM conceptos, contestaciones
WHERE código = orden
AND código LIKE "A005" or código LIKE "A009"

código	respuesta	ciudad	país
A009	americana	SCO	ES
A009	americana	SCO	ES
A009	americana	SCO	ES
A009	americana	SCO	ES
A009	americana	OVI	ES
A009	americana	SLM	ES
A009	americana	SLM	ES

Fig. 14

2.5.d. ¿Qué formas utilizan para el término A001 (“jacket”) los encuestados varones de más de 40 años en Argentina?

Aunque para responder a esta pregunta hay que buscar datos en tres tablas diferentes, procederemos de forma análoga a las anteriores indicando los campos que queremos que nos muestre y las condiciones. Por ejemplo, una forma simple de buscar los datos deseados sería la siguiente.

SELECT *conceptos.código, inglés, respuesta, sexo, edad, informantes.país*
FROM *conceptos, contestaciones, informantes*
WHERE *contestaciones.orden = conceptos.código*
AND *contestaciones.orden* LIKE "A001"
AND *informantes.país* LIKE "AR"
AND *sexo* = 1
AND *edad* > 40

En primer lugar, nos llaman la atención las expresiones del tipo *informantes.país, contestaciones.orden,* etc. con los nombres de la tabla y el campo separados por un punto. De este modo se evita la ambigüedad que pueda existir al haber nombres de campos idénticos en tablas diferentes. A continuación mostramos en la Fig. 15 las primeras líneas del resultado obtenido.

Mostrando campos - 30 (total)

SQL-query:
SELECT DISTINCT conceptos.código, inglés, respuesta, sexo, edad, informantes.país
FROM conceptos, contestaciones, informantes
WHERE contestaciones.orden = conceptos.código
AND contestaciones.orden LIKE "A001"
AND informantes.país LIKE "AR"
AND sexo = 1
AND edad > 40

código	inglés	respuesta	sexo	edad	país
A001	JACKET	americana	1	57	AR
A001	JACKET	americana	1	45	AR
A001	JACKET	americana	1	42	AR
A001	JACKET	chaqueta	1	57	AR
A001	JACKET	chaqueta	1	45	AR
A001	JACKET	chaqueta	1	42	AR
A001	JACKET	traje	1	57	AR
A001	JACKET	traje	1	45	AR
A001	JACKET	traje	1	42	AR
A001	JACKET	chacheta	1	57	AR
A001	JACKET	chacheta	1	45	AR
A001	JACKET	chacheta	1	42	AR
A001	JACKET	El 'traje' lo formaría también el pantalón.	1	57	AR

Fig. 15

2.5.e ¿Cómo ordenar los resultados por orden alfabético?

Para ordenar los resultados obtenidos según el orden alfabético de uno de los campos utilizados en la búsqueda es suficiente añadir una cláusula ORDER BY *campo* {ASC|DESC}. Por ejemplo, si queremos ordenar por orden alfabeto ascendente (en este caso no es necesario añadir ASC a la cláusula) o en orden descendente (hay que añadir DESC) las respuestas obtenidas en el ejemplo de 2.5.d tendremos que añadir la cláusula ORDER BY correspondiente, como se ve en la siguiente fórmula para orden descendente.

SELECT *conceptos.código, inglés, respuesta, sexo, edad, informantes.país*
FROM *conceptos, contestaciones, informantes*
WHERE *contestaciones.orden = conceptos.código*
AND *contestaciones.orden* LIKE "A001"
AND *informantes.país* LIKE "AR"
AND *sexo* = 1
AND *edad* > 40
ORDER BY respuesta DESC

Con esta cláusula añadida, el resultado que obtenemos es el que se muestra en la Fig. 16 a continuación.

SQL-query:
SELECT conceptos.código, inglés, respuesta, sexo, edad, informantes.país
FROM conceptos, contestaciones, informantes
WHERE contestaciones.orden = conceptos.código
AND contestaciones.orden LIKE "A001"
AND informantes.país LIKE "AR"
AND sexo = 1
AND edad > 40
order by respuesta DESC

código	inglés	respuesta	sexo	edad	país
A001	JACKET	vestón	1	57	AR
A001	JACKET	vestón	1	45	AR
A001	JACKET	vestón	1	42	AR
A001	JACKET	vestón	1	57	AR
A001	JACKET	vestón	1	45	AR
A001	JACKET	vestón	1	42	AR

Fig. 16

2.6. Búsquedas utilizando expresiones regulares

Por medio de las llamadas expresiones regulares podemos realizar búsquedas con condiciones extremadamente complejas y al mismo tiempo breves y rápidas. La expresión regular la utilizaremos tras la cláusula REGEXP en lugar de la cláusula LIKE que hemos utilizado hasta ahora. Vamos a ver algunos ejemplos simples pero prácticos.

El acento circunflejo (^) indica el principio de una secuencia de caracteres, que para nosotros será en muchos casos sinónimo de comienzo de palabra. El símbolo $ significa final de secuencia, o sea, de palabra. Por ejemplo, para expresar la condición de palabras que empiezan por “manc”, podemos utilizar la expresión REGEXP “^manc”. En una búsqueda como:

```
SELECT DISTINCT respuesta, ciudad, país
FROM contestaciones
WHERE respuesta REGEXP "^manc"
```

encontraremos “mancuerna”, “mancuernas”, “mancornillas”, etc., pero no “amancebarse”, que sí entraría si la condición fuera WHERE respuesta REGEXP "manc" sin el acento circunflejo que señala el principio de palabra. Las primeras líneas del resultado de la búsqueda sería como la Fig. 17

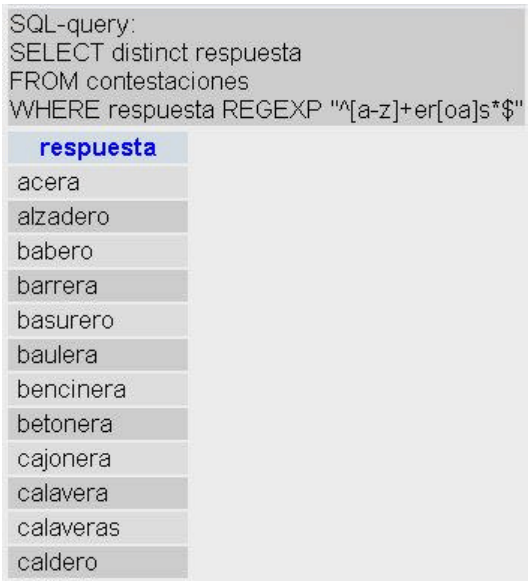
SQL-query:
SELECT distinct respuesta, ciudad, país
FROM contestaciones
WHERE respuesta REGEXP "^manc"

respuesta	ciudad	país
mancornas	MED	CO
mancornas	TAC	VE
mancuernas	AGS	MX
mancuernas	ARI	CH
mancuernas	MAN	NI

Fig. 17

Supongamos ahora que queremos buscar formas terminadas en “-ero”, con sus formas masculina, femenina, singular y plural. Tal condición se podrá expresar según la fórmula siguiente:

```
SELECT DISTINCT respuesta, ciudad, país
FROM contestaciones
WHERE respuesta REGEXP "^[a-z]+er[oa]s*$"
```


SQL-query:
SELECT distinct respuesta
FROM contestaciones
WHERE respuesta REGEXP "^[a-z]+er[oa]s*$"

respuesta
acera
alzadero
babero
barrera
basurero
baulera
bencinera
betonera
cajonera
calavera
calaveras
caldero

Fig. 18

El primer acento circunflejo (^) de la expresión significa, como hemos visto antes, principio de palabra. Esto nos evita que en nuestra búsqueda entren algunas respuestas que no constan de un solo término, sino que tienen una explicación o varias palabras. El siguiente corchete, [a-z]+, significa una secuencia de caracteres entre la a y la z, es decir, cualquier letra del alfabeto, y el signo + significa que pueden ser varias las letras. Después, obligatoriamente debe aparecer la secuencia “er” seguido opcionalmente de “a” o de “o”, lo cual se expresa poniendo estas alternativas dentro de un corchete sin separarlas como hemos hecho antes con un guión. Más adelante “s*” significa que puede haber una “s”, varias o ninguna. Finalmente, el símbolo $ indica el final de la secuencia. Por supuesto, la expresión se puede simplificar a REGEXP "er[oa]s*$", indicando solamente el

final. En este caso, si el lector hace la prueba comprobará que algunas de las respuestas constan de más de una palabra, como era de esperar.

Otro ejemplo de uso de corchetes podría ser la expresión de "chompa", "chumpa" y "chomba", como "ch[ou]m[bp]a". Podemos ver el resultado de una búsqueda de ciudades donde se utiliza al menos una de las tres formas, en este caso completo, en la Fig. 19 a continuación.

```
SQL-query:
SELECT distinct orden, respuesta, ciudad, país
FROM contestaciones
WHERE respuesta REGEXP "ch[ou]m[bp]a$"
```

orden	respuesta	ciudad	país
A002	chompa	ARE	PE
A002	chompa	PAZ	BO
A002	chumpa	SSA	EL
A004	chomba	SCH	CH
A004	chompa	ARE	PE
A004	chompa	PAZ	BO
A005	chompa	MED	CO
A005	chompa	NYK	EU
A005	chumpa	LEO	NI
A005	chumpa	MAN	NI
A005	chumpa	NAC	HO
A005	chumpa	NOR	EU
A005	chumpa	NYK	EU
A005	chumpa	SSA	EL
A009	chompa	PAZ	BO
A015	chumpa	MAN	NI

Fig. 19

En la Fig. 19 podemos apreciar, además de los lugares donde se utilizan estas tres formas, que pertenecen a conceptos (ámbitos conceptuales) diferentes, expresados en el número de orden de la izquierda.

Como último ejemplo de expresión regular, vamos a ver en qué ciudades no se utiliza (es decir, que no tenemos datos) ni "chaqueta" ni "saco". Por supuesto, queremos que se refiere al concepto A001. Para ello, utilizaremos la siguiente fórmula y el resultado lo veremos en la Fig. 20:

```
SELECT DISTINCT respuesta, ciudad, país
FROM contestaciones
WHERE respuesta NOT REGEXP "chaqueta|saco"
AND orden LIKE “A001”
ORDER BY ciudad
```

SQL-query:

```
SELECT distinct orden, respuesta, ciudad, país
FROM contestaciones
WHERE respuesta NOT REGEXP "chaqueta|saco"
AND orden LIKE "A001"
order by ciudad
```

orden	respuesta	ciudad	país
A001	americana	ALM	ES
A001	americana	ARE	PE
A001	vestón	ARI	CH
A001	americana	BAR	ES
A001	traje	BUE	AR
A001	vestón	CON	CH
A001	traje	COR	ES
A001	chaleco	DOR	PR
A001	gabán	DOR	PR
A001	americana	GDL	ES

Fig. 20

3. Conclusión

Las bases de datos son una herramienta de trabajo muy útiles en cualquier campo, ya que nos ayuda no solamente a ordenarlos sino también a descubrir distribuciones de las que no somos conscientes, a preparar listas con las características que deseemos en breve tiempo, etc. Sin embargo, para poder sacar el máximo provecho de las bases de datos son necesarios algunos conocimientos técnicos a los que no siempre estamos acostumbrados. Creemos que las bases de datos relacionales, administradas por medio de programas de lenguaje SQL pueden ser de gran utilidad al investigador de dialectología, lexicología, sociolingüística, etc. Hemos preparado una breve presentación de la estructura de la base de datos de Varilex y algunas técnicas simples de búsqueda para mostrar su funcionamiento, que esperamos sean de utilidad. Esperamos también la colaboración de los usuarios de la base de datos para su mejora, para intercambiar

fórmulas de búsqueda, para mejorar su estructura, así como también intentaremos responder a las consultas que nos lleguen.

Referencias

Bowman, Judith S.; Emerson, Sandra L; Darnovsky, Marcy: *The Practical Sql Handbook : Using Structured Query Language*, Addison-Wesley Pub, 1996

Dubois, Paul; Widenius Monty: *MySQL*, New Riders Publishing, 1999

Gruber, Martin: *Understanding SQL*,

Ruiz Tinoco, Antonio, *El Proyecto Varilex en internet, base de datos compartida de variación léxica*, presentado en el XII Congreso de ALFAL, Santiago de Chile, 1999. (una ampliación del texto aparecido en Varilex 7 aparecerá en las Actas de ALFAL)

Yarger, Randy Jay; Reese, George & King, Tim: *MySQL & mSQL*, O'Reilly & Associates, 1999.

Algunas direcciones útiles:

Página del Proyecto Varilex:
http://varilex.call.sophia.ac.jp
Página del Prof. Hiroto Ueda, director del proyecto Varilex
http://gamp.c.u-tokyo.ac.jp/ueda
SQL Tutorial (en inglés), por Chuo-Han Lee
http://homepages.go.com/~chlee1h/sql.html
SQL Tutorial (en inglés), por James Hoffman
http://w3.one.net/~jhoffman/sqltut.htm
Manual de SQL en inglés, por internet.com Corp.
http://sqlcourse.com/