Apuntes básicos de SQL

Unai Estébanez
unai@unainet.net

Índice de contenido

Licencia:

Reconocimiento 3.0 España

1. Definiciones

a. La *obra* es la creación literaria, artística o científica ofrecida bajo los términos de esta licencia.
b. En esta licencia se considera una *prestación* cualquier interpretación, ejecución, fonograma, grabación audiovisual, emisión o transmisión, mera fotografía u otros objetos protegidos por la legislación de propiedad intelectual vigente aplicable.
c. La aplicación de esta licencia a una *colección* (definida más adelante) afectará únicamente a su estructura en cuanto forma de expresión de la selección o disposición de sus contenidos, no siendo extensiva a éstos. En este caso la colección tendrá la consideración de obra a efectos de esta licencia.
d. El *titular originario* es:
    i. En el caso de una obra literaria, artística o científica, la persona natural o grupo de personas que creó la obra.
    ii. En el caso de una obra colectiva, la persona que la edite y divulgue bajo su nombre, salvo pacto contrario.
    iii. En el caso de una interpretación o ejecución, el actor, cantante, músico, o cualquier otra persona que represente, cante, lea, recite, interprete o ejecute en cualquier forma una obra.
    iv. En el caso de un fonograma, el productor fonográfico, es decir, la persona natural o jurídica bajo cuya iniciativa y responsabilidad se realiza por primera vez una fijación exclusivamente sonora de la ejecución de una obra o de otros sonidos.
    v. En el caso de una grabación audiovisual, el productor de la grabación, es decir,

la persona natural o jurídica que tenga la iniciativa y asuma la responsabilidad de las fijaciones de un plano o secuencia de imágenes, con o sin sonido.

vi. En el caso de una emisión o una transmisión, la entidad de radiodifusión.

vii. En el caso de una mera fotografía, aquella persona que la haya realizado.

viii. En el caso de otros objetos protegidos por la legislación de propiedad intelectual vigente, la persona que ésta señale.

e. Se considerarán *obras derivadas* aquellas obras creadas a partir de la licenciada, como por ejemplo: las traducciones y adaptaciones; las revisiones, actualizaciones y anotaciones; los compendios, resúmenes y extractos; los arreglos musicales y, en general, cualesquiera transformaciones de una obra literaria, artística o científica. Para evitar la duda, si la obra consiste en una composición musical o grabación de sonidos, la sincronización temporal de la obra con una imagen en movimiento (*synching*) será considerada como una obra derivada a efectos de esta licencia.

f. Tendrán la consideración de *colecciones* la recopilación de obras ajenas, de datos o de otros elementos independientes como las antologías y las bases de datos que por la selección o disposición de sus contenidos constituyan creaciones intelectuales. La mera incorporación de una obra en una colección no dará lugar a una derivada a efectos de esta licencia.

g. El *licenciador* es la persona o la entidad que ofrece la obra o prestación bajo los términos de esta licencia y le concede los derechos de explotación de la misma conforme a lo dispuesto en ella.

h. *Usted* es la persona o la entidad que ejercita los derechos concedidos mediante esta licencia y que no ha violado previamente los términos de la misma con respecto a la obra o la prestación, o que ha recibido el permiso expreso del licenciador de ejercitar los derechos concedidos mediante esta licencia a pesar de una violación anterior.

i. La *transformación* de una obra comprende su traducción, adaptación y cualquier otra modificación en su forma de la que se derive una obra diferente. La creación resultante de la transformación de una obra tendrá la consideración de obra derivada.

j. Se entiende por *reproducción* la fijación directa o indirecta, provisional o permanente, por cualquier medio y en cualquier forma, de toda la obra o la prestación o de parte de ella, que permita su comunicación o la obtención de copias.

k. Se entiende por *distribución* la puesta a disposición del público del original o de las copias de la obra o la prestación, en un soporte tangible, mediante su venta, alquiler, préstamo o de cualquier otra forma.

l. Se entiende por *comunicación pública* todo acto por el cual una pluralidad de personas, que no pertenezcan al ámbito doméstico de quien la lleva a cabo, pueda tener acceso a la obra o la prestación sin previa distribución de ejemplares a cada una de ellas. Se considera comunicación pública la puesta a disposición del público de obras o prestaciones por procedimientos alámbricos o inalámbricos, de tal forma que cualquier persona pueda acceder a ellas desde el lugar y en el momento que elija.

m. La *explotación* de la obra o la prestación comprende la reproducción, la distribución, la comunicación pública y, en su caso, la transformación.

2. Límites de los derechos. Nada en esta licencia pretende reducir o restringir cualesquiera límites legales de los derechos exclusivos del titular de los derechos de propiedad intelectual de acuerdo con la Ley de propiedad intelectual o cualesquiera otras leyes aplicables, ya sean derivados de usos legítimos, tales como la copia privada o la cita, u otras limitaciones como la resultante de la primera venta de ejemplares (agotamiento).

3. Concesión de licencia. Conforme a los términos y a las condiciones de esta licencia, el licenciador concede, por el plazo de protección de los derechos de propiedad intelectual y a título gratuito, una licencia de ámbito mundial no exclusiva que incluye los derechos siguientes:

a. Derecho de reproducción, distribución y comunicación pública de la obra o la prestación.
b. Derecho a incorporar la obra o la prestación en una o más colecciones.
c. Derecho de reproducción, distribución y comunicación pública de la obra o la prestación lícitamente incorporada en una colección.
d. Derecho de transformación de la obra para crear una obra derivada siempre y cuando se incluya en ésta una indicación de la transformación o modificación efectuada.
e. Derecho de reproducción, distribución y comunicación pública de obras derivadas creadas a partir de la obra licenciada.
f. Derecho a extraer y reutilizar la obra o la prestación de una base de datos.
g. Para evitar cualquier duda, el titular originario:
    i. Conserva el derecho a percibir las remuneraciones o compensaciones previstas por actos de explotación de la obra o prestación, calificadas por la ley como irrenunciables e inalienables y sujetas a gestión colectiva obligatoria.
    ii. Renuncia al derecho exclusivo a percibir, tanto individualmente como mediante una entidad de gestión colectiva de derechos, cualquier remuneración derivada de actos de explotación de la obra o prestación que usted realice.

Estos derechos se pueden ejercitar en todos los medios y formatos, tangibles o intangibles, conocidos en el momento de la concesión de esta licencia. Los derechos mencionados incluyen el derecho a efectuar las modificaciones que sean precisas técnicamente para el ejercicio de los derechos en otros medios y formatos. Todos los derechos no concedidos expresamente por el licenciador quedan reservados, incluyendo, a título enunciativo pero no limitativo, los derechos morales irrenunciables reconocidos por la ley aplicable. En la medida en que el licenciador ostente derechos exclusivos previstos por la ley nacional vigente que implementa la directiva europea en materia de derecho sui generis sobre bases de datos, renuncia expresamente a dichos derechos exclusivos.

4. Restricciones. La concesión de derechos que supone esta licencia se encuentra sujeta y limitada a las restricciones siguientes:

a. Usted puede reproducir, distribuir o comunicar públicamente la obra o prestación solamente bajo los términos de esta licencia y debe incluir una copia de la misma, o su Identificador Uniforme de Recurso (URI). Usted no puede ofrecer o imponer ninguna condición sobre la obra o prestación que altere o restrinja los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Usted no puede sublicenciar la obra o prestación. Usted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías. Usted no puede reproducir, distribuir o comunicar públicamente la obra o prestación con medidas tecnológicas que controlen el acceso o el uso de una manera contraria a los términos de esta licencia. Esta sección 4.a también afecta a la obra o prestación incorporada en una colección, pero ello no implica que ésta en su conjunto quede automáticamente o deba quedar sujeta a los términos de la misma. En el caso que le sea requerido, previa comunicación del licenciador, si usted incorpora la obra en una colección y/o crea una obra derivada, deberá quitar cualquier crédito requerido en el apartado 4.b, en la medida de lo

posible.

b. Si usted reproduce, distribuye o comunica públicamente la obra o la prestación, una colección que la incorpore o cualquier obra derivada, debe mantener intactos todos los avisos sobre la propiedad intelectual e indicar, de manera razonable conforme al medio o a los medios que usted esté utilizando:
    i. El nombre del autor original, o el seudónimo si es el caso, así como el del titular originario, si le es facilitado.
    ii. El nombre de aquellas partes (por ejemplo: institución, publicación, revista) que el titular originario y/o el licenciador designen para ser reconocidos en el aviso legal, las condiciones de uso, o de cualquier otra manera razonable.
    iii. El título de la obra o la prestación si le es facilitado.
    iv. El URI, si existe, que el licenciador especifique para ser vinculado a la obra o la prestación, a menos que tal URI no se refiera al aviso legal o a la información sobre la licencia de la obra o la prestación.
    v. En el caso de una obra derivada, un aviso que identifique la transformación de la obra en la obra derivada (p. ej., "traducción castellana de la obra de Autor Original," o "guión basado en obra original de Autor Original").

    Este reconocimiento debe hacerse de manera razonable. En el caso de una obra derivada o incorporación en una colección estos créditos deberán aparecer como mínimo en el mismo lugar donde se hallen los correspondientes a otros autores o titulares y de forma comparable a los mismos. Para evitar la duda, los créditos requeridos en esta sección sólo serán utilizados a efectos de atribución de la obra o la prestación en la manera especificada anteriormente. Sin un permiso previo por escrito, usted no puede afirmar ni dar a entender implícitamente ni explícitamente ninguna conexión, patrocinio o aprobación por parte del titular originario, el licenciador y/o las partes reconocidas hacia usted o hacia el uso que hace de la obra o la prestación.
c. Para evitar cualquier duda, debe hacerse notar que las restricciones anteriores (párrafos 4.a y 4.b) no son de aplicación a aquellas partes de la obra o la prestación objeto de esta licencia que únicamente puedan ser protegidas mediante el derecho sui generis sobre bases de datos recogido por la ley nacional vigente implementando la directiva europea de bases de datos

5. Exoneración de responsabilidad

A MENOS QUE SE ACUERDE MUTUAMENTE ENTRE LAS PARTES, EL LICENCIADOR OFRECE LA OBRA O LA PRESTACIÓN TAL CUAL (ON AN "AS-IS" BASIS) Y NO CONFIERE NINGUNA GARANTÍA DE CUALQUIER TIPO RESPECTO DE LA OBRA O LA PRESTACIÓN O DE LA PRESENCIA O AUSENCIA DE ERRORES QUE PUEDAN O NO SER DESCUBIERTOS. ALGUNAS JURISDICCIONES NO PERMITEN LA EXCLUSIÓN DE TALES GARANTÍAS, POR LO QUE TAL EXCLUSIÓN PUEDE NO SER DE APLICACIÓN A USTED.

6. Limitación de responsabilidad. SALVO QUE LO DISPONGA EXPRESA E IMPERATIVAMENTE LA LEY APLICABLE, EN NINGÚN CASO EL LICENCIADOR SERÁ RESPONSABLE ANTE USTED POR CUALESQUIERA DAÑOS RESULTANTES, GENERALES O ESPECIALES (INCLUIDO EL DAÑO EMERGENTE Y EL LUCRO CESANTE), FORTUITOS O CAUSALES, DIRECTOS O INDIRECTOS, PRODUCIDOS EN CONEXIÓN CON ESTA LICENCIA O EL USO DE LA OBRA O LA PRESTACIÓN, INCLUSO SI EL LICENCIADOR HUBIERA SIDO INFORMADO DE LA POSIBILIDAD DE TALES DAÑOS.

7. Finalización de la licencia

a. Esta licencia y la concesión de los derechos que contiene terminarán automáticamente en caso de cualquier incumplimiento de los términos de la misma. Las personas o entidades que hayan recibido de usted obras derivadas o colecciones bajo esta licencia, sin embargo, no verán sus licencias finalizadas, siempre que tales personas o entidades se mantengan en el cumplimiento íntegro de esta licencia. Las secciones 1, 2, 5, 6, 7 y 8 permanecerán vigentes pese a cualquier finalización de esta licencia.
b. Conforme a las condiciones y términos anteriores, la concesión de derechos de esta licencia es vigente por todo el plazo de protección de los derechos de propiedad intelectual según la ley aplicable. A pesar de lo anterior, el licenciador se reserva el derecho a divulgar o publicar la obra o la prestación en condiciones distintas a las presentes, o de retirar la obra o la prestación en cualquier momento. No obstante, ello no supondrá dar por concluida esta licencia (o cualquier otra licencia que haya sido concedida, o sea necesario ser concedida, bajo los términos de esta licencia), que continuará vigente y con efectos completos a no ser que haya finalizado conforme a lo establecido anteriormente, sin perjuicio del derecho moral de arrepentimiento en los términos reconocidos por la ley de propiedad intelectual aplicable.

8. Miscelánea

a. Cada vez que usted realice cualquier tipo de explotación de la obra o la prestación, o de una colección que la incorpore, el licenciador ofrece a los terceros y sucesivos licenciatarios la concesión de derechos sobre la obra o la prestación en las mismas condiciones y términos que la licencia concedida a usted.
b. Cada vez que usted realice cualquier tipo de explotación de una obra derivada, el licenciador ofrece a los terceros y sucesivos licenciatarios la concesión de derechos sobre la obra objeto de esta licencia en las mismas condiciones y términos que la licencia concedida a usted.
c. Si alguna disposición de esta licencia resulta inválida o inaplicable según la Ley vigente, ello no afectará la validez o aplicabilidad del resto de los términos de esta licencia y, sin ninguna acción adicional por cualquiera las partes de este acuerdo, tal disposición se entenderá reformada en lo estrictamente necesario para hacer que tal disposición sea válida y ejecutiva.
d. No se entenderá que existe renuncia respecto de algún término o disposición de esta licencia, ni que se consiente violación alguna de la misma, a menos que tal renuncia o consentimiento figure por escrito y lleve la firma de la parte que renuncie o consienta.
e. Esta licencia constituye el acuerdo pleno entre las partes con respecto a la obra o la prestación objeto de la licencia. No caben interpretaciones, acuerdos o condiciones con respecto a la obra o la prestación que no se encuentren expresamente especificados en la presente licencia. El licenciador no estará obligado por ninguna disposición complementaria que pueda aparecer en cualquier comunicación que le haga llegar usted. Esta licencia no se puede modificar sin el mutuo acuerdo por escrito entre el licenciador y usted.

Introducción

Estos apuntes son una introducción muy básica al SQL, en ningún caso sustituyen a un buen manual. Simplemente pueden dar una idea rápida de que es el SQL o bien ayudar a recordar conceptos que no se utilizaban hace mucho tiempo.

SQL responde a Structured Query Language, es un lenguaje declarativo para acceder a las bases de datos relacionales (hay otros tipos pero no vamos a hablar aquí de ellas). Nos permitirá crear nuevas bases de datos, modificar bases de datos existentes y consultar y añadir información.

Estándares de SQL

SQL es un estándar ANSI(American National Standards Institute) desde 1986 y esto implica que conociendo este lenguaje podemos manipular cualquier base de datos de cualquier fabricante. Por supuesto cada fabricante tendrá extensiones al lenguaje SQL pero si nos apegamos al estándar conseguiremos hacer que nuestra consultas funcionen en cualquier base de datos. Este estándar fué también adoptado por la ISO en 1987.

Los estándares de SQL evolucionaron de la siguiente forma:

Año	Nombre
1986	SQL-86
1989	SQL-89
1992	SQL-92
1999	SQL:1999
2003	SQL:2003
2006	SQL:2006
2008	SQL:2008

En la actualidad SQL es el estándar de facto de la mayoría de los fabricantes de bases de datos relacionales y aunque cada uno tiene sus propias extensiones la mayoría tiene un buen soporte al menos para SQL-92.

Las tablas

Los datos en una base de datos relacionales se guardarán siempre en una o más tablas. Una tabla consistirá en una o más columnas y una o más filas.

Con las columnas especificamos los datos que vamos a guardar y con cada fila introducimos los datos en sí mismos.

Ejemplo:

Nombre	Apellidos	Edad
Unai	Estébanez	32
Jon	Zaragoza	30

Aquí podemos ver una tabla con tres columnas y dos filas (de datos).

Cada tabla se identificará con un nombre, por ejemplo "Contactos".

A cada fila se le denomina comúnmente **"Registro"**.

A cada parte de la fila (cada columna de una fila) se le llama **"Campo"**.

Sentencias SQL

Para acceder a las tablas y consultar o modificar los datos usaremos sentencias SQL.

Ejemplo:

SELECT * FROM Contactos; ← A esto se le llama sentencia SQL.

Esta sentencia SQL nos permite obtener todas los datos de la tabla "Contactos". A una sentencia se le llama a una serie de comandos junto con unas cláusulas, operadores, funciones, etc.. Todo ello forma una "frase" (sentencia) y nos permite realizar una operación.

Ten en cuenta que....

- SQL no entiende de mayúsculas o minúsculas, le da igual lo que pongas, es lo mismo "SELECT" que "select". Usaré las mayúsculas para las palabras propias del lenguaje SQL.
- En algunas bases de datos es necesario poner un punto y coma para finalizar la sentencia, en otras no. Aún así, el punto y coma es la forma estándar de separar dos sentencias.

Notación para las sentencias SQL

Cuando las sentencias SQL que describamos sean más complejas utilizaremos la siguiente notación:

Los corchetes "[]" indican que lo que hay dentro son palabras clave opcionales.

Estas opciones pueden ser separados por una barra "|" que indica una OR, es decir debemos poner una u otra opción.

Las llaves "{}"indican que es obligatorio poner lo que hay dentro pero podemos elegir cual de las opciones queremos si hay barras "|".

DDL y DML

SQL se puede dividir en dos lenguajes DDL y DML. Cada parte tiene unos comandos para realizar operaciones de distinta naturaleza.

Un comando es una "orden", le indicará a la base de datos lo que esperamos que haga y tal y como hemos dicho tenemos dos grupos:

1. DDL (Data Definition Language): Estos nos permitirán crear nuevas bases de datos, modificarlas y eliminarlas. Estos comandos afectan a tablas, campos e índices.
    - CREATE Crea nuevas tablas, campos e índices.
    - ALTER Modifica las tablas
    - DROP Elimina tablas e índices.
2. DML (Data Manipulation Language): Nos permiten manipular(consultar,ordenar,filtrar,etc) los datos existentes en una base de datos.
    - SELECT Consulta filas en la base de datos
    - UPDATE Modifica valores de una fila
    - INSERT Inserta una nueva fila
    - DELETE Elimina filas

Tipos de datos

En cada columna vamos a meter datos, por ejemplo, un nombre o una edad. Esto hay que decírselo a la base de datos cuando creamos cada tabla[1].

Cada base de datos tiene unos tipos definidos, en nuestro caso, resumiendo (consultar la documentación para más información) tenemos los siguientes tipos:

- NULL
- INTEGER para guardar números enteros.
- REAL para números reales.
- TEXT para guardar texto
- BLOB para guardar datos binarios

Para más información sobre SQLite versión 3 se puede consultar la documentación sobre tipos de datos en (http://www.sqlite.org/datatype3.html).

Manos a la obra

Vamos a jugar un poco con los comandos para entenderlos. Yo he realizado los ejemplos en la consola de SQLite que puedes descargarte en (http://www.sqlite.org/). En principio no debieras tener pegas para usar cualquier otra base de datos, yo he utilizado esta porque es la que estoy usando en mi proyecto actual.

Notas específicas de SQLite

En la consola de SQLite si tecleas **.help** obtendrás ayuda sobre los comandos NO SQL de la base de datos que nos permiten interactuar con esta base de datos concreta. Fíjate en que todos estos comandos van precedidos por un punto y no terminan en punto y coma.

Comandos que nos resultarán útiles son:

1. .tables Este nos muestra que tablas tenemos definidas en la base de datos abierta en este momento.
2. .backup <nombre archivo> Nos permite guardar en disco la base de datos con la que estamos trabajando.
3. .restore <nombre archivo> Nos permite cargar en memoria una base de datos previamente guardada en disco.
4. .dump <nombre tabla> Nos muestra la definición en lenguaje SQL de la tabla que le indicamos.

1 Esto no es cierto para todas las bases de datos, actualmente es posible no indicar que tipo de datos metemos y la base de datos por si misma determinar que tipo es cada dato a medida que se los mete. Aún así es buena costumbre especificarlos y seguir la definición inicial que hacemos.

Creando una tabla (CREATE)

Vamos a crear una tabla de ejemplo:

CREATE TABLE Contactos (Nombre TEXT,Apellidos TEXT,Edad INTEGER);

Esta sentencia nos crea una tabla con tres columnas. Nombre,Apellidos y Edad donde iremos metiendo datos.

Si ahora tecleamos .**tables** veremos que se ha creado una nueva tabla llamada Contactos.

Insertando datos en una tabla (INSERT)

Metamos unos contactos en la tabla:

INSERT INTO Contactos VALUES ("Unai","Estebanez",32);

INSERT INTO Contactos VALUES ("Jon","Zaragoza",30);

INSERT INTO Contactos VALUES ("Jon","Fernandez",28);

INSERT INTO Contactos VALUES ("Unai","Bilbao",32);

INSERT INTO Contactos VALUES ("Basilio","Ramos",60);

INSERT INTO Contactos VALUES ("Andres","Aguirre",31);

Actualizando datos de un registro (UPDATE)

Cambiamos a Basilio por Arantza:

UPDATE Contactos SET Nombre="Arantza" WHERE Nombre="Basilio";

Borrando registros(DELETE)

Borramos a Andres:

DELETE FROM Contactos WHERE Nombre="Andres";

Recuperando datos de una tabla (SELECT)

Para recuperar datos de una tabla utilizaremos el comando SELECT:

El formato general de esta sentencia es:

SELECT <columnas separadas por comas> FROM <tabla>;

Ejemplos:

SELECT * FROM Contactos; ← Recupera todas las columnas de la tabla

SELECT Nombre FROM Contactos; ← Recupera todos los nombres de la tabla

SELECT Nombre,Edad FROM Contactos; ← Recupera todos los nombres y edades de la tabla

Borrando una tabla (DROP)

Supongamos que tenemos una tabla llamada USUARIOS y queremos eliminarla, utilizaremos la sentencia DROP y antes comprobaremos que exista (IF EXISTS):

```
DROP TABLE IF EXISTS USUARIOS;
```

Consultas con predicado

Se llama predicado a lo que va entre el SELECT y el primer nombre de columna que vamos a recuperar. Sirven para modificar el comportamiento de las consultas.

Ejemplo:

```
SELECT DISTINCT Nombre FROM Contactos;
```

Nos devolverá todos los nombre de la tabla contactos pero omitirá las filas cuyos campos seleccionados coincidan totalmente.

En este otro caso solo nos omitirá las filas en las que coincidan los dos campos.

```
SELECT DISTINCT Nombre,Edad FROM Contactos;
```

Predicados comunes son:

- DISTINCT Omitirá las filas cuyos campos seleccionados coincidan totalmente
- ALL Devuelve todos los campos de la tabla
- TOP Devuelve un determinado número de registros de la tabla. En SQLite se soporta como LIMIT.
  - SELECT * FROM Contactos LIMIT 2;
- DISTINCTROW Omite las filas duplicadas basándose en la totalidad del registro y no solo en los campos seleccionados. Esto no lo soporta SQLite.

Cláusulas

Son condiciones de modificación para definir los datos que deseamos seleccionar o manipular. Ya las hemos utilizado sin darnos cuenta, FROM es una de ellas.

Efectivamente FROM "especifica" de que tabla queremos obtener los datos, es por tanto una cláusula.

Cláusulas comunes son

- FROM Especifica la tabla de la que cogemos los registros
- WHERE Indica condiciones que deben cumplir los registros seleccionados
- GROUP BY Separa los registros seleccionados en grupos específicos.

- HAVING Indica la condición que debe cumplir cada grupo.
- ORDER BY Ordena los registros según lo que le indiquemos.

Ejemplos:

Los mayores de 30 años: SELECT * FROM Contactos WHERE Edad > 30;

Los que se llaman Unai: SELECT * FROM Contactos WHERE Nombre="Unai";

Operadores para la cláusula WHERE

La cláusula WHERE admite combinaciones para especificar mejor lo que quieres seleccionar:

Operador	Significado
<>	Distinto
<	Menor
>	Mayor
<=	Menor o igual
>=	Mayor o igual
BETWEEN	Indica un rango (numérico, texto,fecha,...)
LIKE	Busca un patrón
IN	Indica un valor concreto
=	Igual
AND	Y lógico
OR	O lógico
NOT	Negación lógica

Ejemplos:

Todos los que se llamen Jon y tengan 30 años o más.

SELECT * FROM Contactos WHERE Nombre="Jon" AND Edad >= 30;

El AND y el OR se pueden combinar usando paréntesis para agrupar las condiciones:

SELECT * FROM Contactos WHERE (Nombre="Jon" OR Nombre="Unai") AND (Edad >= 30);

Nos da todos los nombre ordenados ascendente:
SELECT * FROM Contactos ORDER BY Nombre;
Esto es equivalente a:
SELECT * FROM Contactos ORDER BY Nombre ASC;

Nos da todos los nombre ordenados descendente:

SELECT * FROM Contactos ORDER BY Nombre DESC;

Todos los que tengan un nombre entre "Arantza" y "Jon":
SELECT * From Contactos WHERE Nombre BETWEEN "Arantza" AND "Jon";

Todos los que tengan 30 o 32 años:
SELECT * From Contactos WHERE Edad IN (30,32);

Todos cuyo nombre acabe en "n": (El % indica cualquier número de caracteres antes o después, en este caso antes).
SELECT * From Contactos WHERE Nombre LIKE '%n';

Todos los que NO acaben en "n":
SELECT * From Contactos WHERE Nombre NOT LIKE '%n';

Comodines para el LIKE

Comodín	Descripción
%	Sustituto para cero o más caracteres.
_	Sustituto para exactamente un carácter
[lista caracteres]	Cualquier carácter de la lista
[^lista caracteres] o bien [!lista caracteres]	Cualquier carácter que no esté en la lista

El GROUP BY el HAVING los veremos en un apartado específico puesto que requieren algo más de conocimientos.

Alias

Sirven para asignar un nombre alternativo a alguna tabla o columna.

Es útil cuando tenemos nombre muy largos de columna o tabla y queremos referirnos a ellos de forma más cómoda:

Ejemplo:

SELECT Apellidos AS A FROM Contactos WHERE A LIKE "%a%";

Hemos seleccionado todos los apellidos, les hemos asignado un alias "A" y hemos hecho una selección de todos los que tienen una "a".

Sin alias hubiese quedado:

SELECT Apellidos FROM Contactos WHERE Apellidos LIKE "%a%";

En este caso puede no parecer muy impresionante puesto que ocupa prácticamente lo mismo, pero en consultas más complejas y con varios campos puede ahorrarnos bastante espacio.

Operando con varias tablas

Hasta ahora hemos visto sencillas operaciones sobre una sola tabla, pero este no es el caso más común. Habitualmente tendremos varias tablas sobre las que operar para obtener los datos que necesitamos.

Para poder relacionar entre si varias tablas tendremos que hacer que tengan algo en común.

Conceptos de claves (Primary key)

Esos datos en común entre varias tablas son las claves. Cuando una columna tiene valores que son únicos para cada registro, de forma que se pueda distinguir dos registros que de otra forma serían iguales, se dice que esa columna es la clave primaria.

Una **primary key** no tiene porque estar formada por una sola columna, podría estar compuesta por más de una.

Para ello debemos especificar al crear la tabla cual es nuestra primary key.

En otras bases de datos es posible añadir una columna que sea primary key una vez creada la base de datos. En nuestro caso, con SQLite, no podemos hacer eso. Debemos por tanto crear la tabla de nuevo desde cero.

Borramos la tabla que ya teníamos:

```
DROP TABLE Contactos;
```

Creamos la nueva tabla con una clave primaria y que no puede repetirse(c_id) :

```
CREATE TABLE Contactos ( c_id INTEGER NOT NULL PRIMARY KEY ASC,nombre TEXT,apellidos TEXT,edad INTEGER );
```

Insertamos los valores de antes:

```
INSERT INTO Contactos (nombre,apellidos,edad) VALUES ("Unai","Estebanez",32);
INSERT INTO Contactos (nombre,apellidos,edad) VALUES ("Jon","Zaragoza",30);
INSERT INTO Contactos (nombre,apellidos,edad) VALUES ("Jon","Fernandez",28);
INSERT INTO Contactos (nombre,apellidos,edad) VALUES ("Unai","Bilbao",32);
INSERT INTO Contactos (nombre,apellidos,edad) VALUES ("Basilio","Ramos",60);
INSERT INTO Contactos (nombre,apellidos,edad) VALUES ("Andres","Aguirre",31);
```

Si os dais cuenta,no hemos indicado el valor del campo clave primaria, el solo se auto incrementa pues así se lo hemos indicado.

Una tabla con varias primary keys

Podemos definir que una tabla tenga varios campos como primary key, veámoslo con un ejemplo:

Supongamos que queremos definir tres tablas, una de usuarios de un servicio de telefonía, otra que asocia a cada usuario con uno o varios móviles y otra que asocia datos de tiempo real obtenidos del móvil. Esta última estará asociada tanto con cada usuario como con cada móvil, además para un momento(fecha y hora) dados tendremos un conjunto de datos concreto para cada usuario y móvil.

Las sentencias SQL de creación de las tablas quedarían tal y como sigue:

CREATE TABLE USUARIOS (ID TEXT NOT NULL PRIMARY KEY);

CREATE TABLE TELEFONOS (IP TEXT, NUMERO TEXT NOT NULL PRIMARY KEY, BATERIA INTEGER, ID TEXT NOT NULL,FOREIGN KEY (ID) REFERENCES USUARIOS (ID));

CREATE TABLE DATOS_TR (ID TEXT NOT NULL,NUMERO TEXT NOT NULL, TIMESTAMP TEXT NOT NULL, LONGITUD REAL,LATITUD REAL,PRECISION_HOR INTEGER,ALTITUD FLOAT,PRECISION_ALT INTEGER,VELOCIDAD INTEGER,RUMBO INTEGER,FOREIGN KEY (ID) REFERENCES USUARIOS (ID),FOREIGN KEY (NUMERO) REFERENCES TELEFONOS(NUMERO),PRIMARY KEY (ID,NUMERO,TIMESTAMP));

Conceptos de claves (Foreign key)

Se trata de una clave que no es primaria en esta tabla pero que si lo es en otra tabla a la que se hace referencia. Lo aclaro con un ejemplo:

c_id (primary_key)	Nombre	Apellidos	Edad
1	Unai	Estébanez	32
2	Jon	Zaragoza	30

o_id(primary_key)	Orden de compra	c_id(foreign key)
1	OR-1	2
2	OR-2	2
3	OR-3	1

En este caso tenemos dos tablas, una con contactos y otra con órdenes de compra. Se supone que cada orden de compra que tenemos la ha realizado uno de nuestros contactos. Como debemos saber a quien pertenece cada orden añadimos una columna que es una clave extranjera (foreign key) para poder cruzar los datos.

Creemos una nueva tabla con las ordenes donde le indicamos los dos tipos de claves:

CREATE TABLE Ordenes (o_id INTEGER NOT NULL PRIMARY KEY ASC,orden TEXT,c_id INTEGER, FOREIGN KEY (c_id) REFERENCES Contactos (c_id));

Insertamos nuestra primera orden de compra, que por el momento no está asociada a nadie:
INSERT INTO Ordenes (orden) VALUES ("OR-1");

Ahora le asignamos esa orden a "Unai Estébanez":
UPDATE Ordenes SET c_id=1;

Joins

Join es el comando que utilizaremos para obtener datos de varias tablas relacionadas entre sí.

Los tipos de join son:

- JOIN (INNER JOIN) Devuelve los registros que tienen al menos una coincidencia en las dos tablas
- LEFT JOIN Devuelve los registros de la tabla izquierda incluso si no hay coincidencia con la tabla derecha
- RIGHT JOIN Devuelve los registros de la tabla derecha incluso si no hay coincidencia con la tabla izquierda. No soportada en SQLite.
- FULL JOIN Devuelve los registros que tienen coincidencia en una de las tablas. No soportada en SQLite.

Ejemplos:

Buscamos el nombre de las personas que tienen una orden de compra asociada:

SELECT Contactos.nombre, Ordenes.Orden FROM Contactos INNER JOIN Ordenes ON Contactos.c_id = Ordenes.c_id;

Esto nos devuelve "Unai" y su orden de compra "OR-1".

Esta otra sentencia SQL nos devolverá todos los datos de la tabla de la izquierda:

SELECT Contactos.nombre, Ordenes.Orden FROM Contactos LEFT JOIN Ordenes ON Contactos.c_id = Ordenes.c_id;

Esto nos devuelve todos los nombres pero solo "Unai" tendrá una orden, el resto tendrá vacío ese campo.

Uniones (UNION y UNION ALL)

Los operadores UNION y UNION ALL sirve para combinar los datos de dos consultas.

Por ejemplo, los que tienen 28 y los que tienen 32 años:

```
SELECT Nombre,Apellidos,Edad FROM Contactos Where Edad = 28
UNION ALL
SELECT Nombre,Apellidos,Edad FROM Contactos Where Edad = 32;
```

El UNION ALL permite duplicados y el UNION no.

Funciones

Sirven para hacer cálculos sobre grupos de datos[2]. Hay dos tipos de funciones:

1. Funciones escalares Devuelven un valor único resultante de procesar un dato.
   - UCASE En SQLite es upper
   - LCASE En SQLite es lower
   - MID
   - LEN
   - ROUND
   - NOW
   - FORMAT
2. Funciones agregadas Devuelven un valor único resultante de procesar una columna.
   - AVG
   - COUNT
   - FIRST
   - LAST
   - MAX
   - MIN
   - SUM

Ejemplos de funciones agregadas:

Cuenta las columnas "nombre" de la tabla contactos.

```
SELECT COUNT(nombre) FROM Contactos;
```

La media de los precios de las ordenes:

```
SELECT AVG(precio) FROM Ordenes;
```

Ejemplos de funciones escalares:

Saca los nombres en mayúsculas:

```
SELECT upper(nombre) FROM Contactos;
```

2 Los nombres de funciones pueden cambiar con cada base de datos. Consultad el manual correspondiente.

GROUP BY

La sentencia GROUP BY sirve para agrupar resultados en conjuntos y de esta forma aplicar operaciones sobre esos conjuntos. Se usan en conjunción con las funciones agregadas.

Antes, vamos a añadir una columna a la tabla de órdenes, se trata del precio de cada orden:

```
ALTER TABLE Ordenes ADD precio REAL;
```

Y ahora añadimos un precio:

```
UPDATE ordenes SET precio=12 WHERE o_id=1;
```

Vamos a añadir algunas ordenes mas para hacer pruebas con el group by:

```
INSERT INTO Ordenes (orden,c_id,precio) VALUES ("OR-2",1,25);
INSERT INTO Ordenes (orden,c_id,precio) VALUES ("OR-3",2,13.6);
```

A continuación vamos a obtener todas las ordenes de compra y vamos a agruparlas por nombre para además sumar los precios que corresponden a cada nombre:

```
SELECT nombre,apellidos,SUM(Ordenes.precio) FROM Contactos INNER JOIN Or
denes ON Contactos.c_id = Ordenes.c_id GROUP BY nombre;
```

Esto podemos hacerlo porque se ha realizado una agrupación por nombre. Si no hacemos la agrupación el resultado es totalmente distinto ya que suma todos los precios de la columna precios que sale tras la selección.

HAVING

El having tiene su sentido porque no es posible usar funciones agregadas en el WHERE.

Por ejemplo:

Vamos a seleccionar de la tabla Contactos todos aquellos que tengan el mismo nombre, los agrupamos y sumamos su edad:

```
SELECT nombre,SUM(edad) FROM Contactos GROUP BY nombre HAVING COUNT(edad) > 1;
```

Índices

Recordemos nuestra tabla de Contactos, en esta tabla ejecutar la sentencia:

SELECT * FROM Contactos WHERE nombre="Unai";

Esto supone recorrer la tabla, fila por fila, hasta dar con cada entrada cuyo nombre es Unai. Esto, si la tabla es grande, puede hacer ineficiente la consulta.

Los índices son usados para acelerar las consultas frecuentes en una Base de Datos. Un índice ayuda a la Base de Datos a identificar y obtener filas mucho más rápido.

Los índices lo que hacen es crear una estructura de datos, en el caso de sqlite un btree, que permite buscar el valor que le pedimos de forma muy rápida con tan solo unos pocos accesos a la estructura de datos.

Pero atención,un uso incorrecto de estos podría causar que el sistema se sobrecargue, por lo que hay que tener mucho cuidado al aplicarlos.

Podemos crear un índice con la siguiente sentencia:

CREATE INDEX indice_nombre ON Contactos (nombre);

Esto hace que se cree el índice. En principio ya no debemos hacer nada más, cuando la tabla es modificada, ya sea por una inserción, modificación o eliminación, el índice es actualizado automáticamente

Tecleando en la consola de sqlite:

.indices Contactos

Se nos muestra "indice_nombre" que indica que ya se ha creado el índice en la tabla.

Para borrar un índice:

DROP INDEX indice_nombre;

Es posible crear índices por varios campos separando estos por comas dentro de los paréntesis:

CREATE INDEX indice_nombre ON Contactos (nombre,apellidos);

Constraints - Restricciones

Se llama así al restringir los valores que puede tomar una determinada columna de la tabla. Estas restricciones las podemos añadir al crear una tabla con CREATE_TABLE o bién al modificarla mediante ALTER_TABLE.

Restricciones típicas pueden ser por ejemplo que una columna no pueda ser NULL, que no tome un valor determinado o que no tenga una longitud mayor que una dada.

Entre estas restricciones típicas tenemos:

- NOT NULL (Para que no se permita un valor NULL)
- PRIMARY KEY (¿alguien no sabe a estas alturas para que sirve esto?)
- UNIQUE (Para obligar a que no se repitan valores en esa columna)
- CHECK (Para verificar una condición dada)

Ejemplos:

```
CREATE TABLE RSU (ID_RSU TEXT UNIQUE PRIMARY KEY,
                  Descr TEXT,
                  RSU_type TEXT,
                  ComsParams TEXT,
                  Road TEXT,
                  PK INTEGER,
                  Dir TEXT check (length(Dir) <= 2),
                  POS_X REAL,
                  POS_Y REAL,
                  POS_Z REAL,
                  RT_Timestamp TEXT,
                  RT_Status TEXT,
                  Algorithms_path TEXT check(length(Algorithms_path) <= 260));
```

Tenemos en esta tabla cuatro restricciones, una PRIMARY_KEY y UNIQUE y dos CHECKs.

Como se puede ver el mecanismo es sencillo, se le indica que esos campos no pueden tener más de una longitud dada y ya esta.

Al intentar introducir datos que no cumplen las condiciones se produce un error como este "SQL error: constraint failed" en el caso de intentar meter un texto más grande del debido o como este "SQL error: column ID_RSU is not unique" en el caso de querer meter una columna con valores duplicados.

Triggers

Un trigger (“gatillo”) consiste en ejecutar una secuencia de sentencias cuando ocurre un evento dado.

La idea de esto es dar de alta un trigger para que se ejecute cuando ocurra un evento que indiquemos nosotros y de esta forma se realicen operaciones en la base de datos de forma automática.

Se pueden ejecutar cuando se produzcan un DELETE un INSERT o un UPDATE de una tabla concreta o bien cuando se produzca un UPDATE de una o más columnas de una tabla.

Tenemos también la posibilidad de ejecutar el trigger cada vez que se actualiza,inserta o borra una fila mediante una sentencia que implique lanzar el trigger.

Otra opción de “configuración del trigger” que tenemos disponible es la de indicar si queremos que el trigger se produzca antes o después de la acción que estamos indicando.

Cuando se ejecute el trigger puede que en ese momento queramos hacer referencia a los nuevos datos o a los antiguos (los que van a ser reemplazados, eliminados o los que se van a insertar nuevos), para ello podremos usar referencias del tipo “NEW.nombre_columna” o “OLD.nombre_columna”.

Los casos que tenemos son:

- Cuando hacemos una INSERT las referencias NEW son válidas
- Cuando hacemos un UPDATE las referencias NEW y OLD son válidas
- Cuando hacemos un DELETE las referencias OLD son válidas.

Supongamos que creamos una tabla para almacenar las ordenes de compras sospechosas de ser erróneas porque tienen un precio muy caro:

```
CREATE TABLE Sospechosas (o_id INTEGER NOT NULL PRIMARY KEY,
                          FOREIGN KEY (o_id) REFERENCES Ordenes(o_id) );
```

Ahora crearemos un trigger para que en cada inserción se compruebe si el precio es de más de 1000 euros y en ese caso metemos la orden en la lista de sospechosas:

```
CREATE TRIGGER check_sospechosas AFTER INSERT ON Ordenes WHEN (NEW.preci
o >= 1000)
BEGIN
INSERT INTO "Sospechosas" VALUES(NEW.o_id);
END;
```

De esta forma al insertar una orden de más de 1000 euros:

```
INSERT INTO "Ordenes" VALUES(4,"OR-4",1,1200);
```

Veremos que se ha creado una entrada en la tabla “Sospechosas” para esta orden de

compra.

Para eliminar un trigger se puede utilizar la sentencia DROP TRIGGER <nombre trigger> o bien se borrará junto con la tabla cuando esta se elimine.

La sintaxis para un trigger es:

CREATE [TEMP | TEMPORARY] TRIGGER [IF NOT EXISTS] [<nombre de la base de datos>.] <nombre del trigger> [BEFORE | AFTER | INSTEAD OF] { DELETE | INSERT | UPDATE [OF <nombre_columna>,...una o más]} ON <nombre tabla> [FOR EACH ROW] [WHEN <expresión>] BEGIN <sentencias finalizadas con;>END;

Es posible cancelar la query en curso si dentro del trigger usamos una sentencia RAISE.

RAISE (IGNORE | { {ROLLBACK | ABORT | FAIL} , <mensaje error>});

Aconsejo echar un vistazo al manual de cada base de datos para verificar el comportamiento de cada una de estas opciones.

Consultas algo más complejas

Poco a poco, al ir peleándonos con la base de datos iremos teniendo necesidad de hacer consultas más complejas, recojo aquí algunas de las que me han surgido a mí por si os pueden ser de utilidad:

Anidando consultas

En mi caso he tenido que anidar consultas para alimentar a una consulta con la salida de otra. Supongamos que tenemos una tabla con un histórico de variables analógicas, la tabla podría ser como esta:

TimeStamp (PK)	OwnerType (PK,FK1)	OwnerCode (PK,FK1)	D_AnalogVar (FK1)	Value	Alm
Cuando se actualizó el valor	Hace referencia a la defínción de la variable en otra tabla	Hace referencia a la defínción de la variable en otra tabla	Hace referencia a la defínción de la variable en otra tabla	Valor de la variable	Indica si se ha producido alarma o no debido al valor de la variable

Ahora necestiamos los valores más antiguos del histórico:
SELECT * FROM AnalogHistorical WHERE ID_AnalogVar="Temperatura" AND OwnerType="MET" AND OwnerCode=1 AND TimeStamp = (aquí hay que anidar);

Lo que debiearmos anidar sería:
SELECT MIN(TimeStamp) FROM AnalogHistorical WHERE ID_AnalogVar="Temperatura" AND OwnerType="MET" AND OwnerCode=1

De esta forma la consulta quedaría así:
SELECT * FROM AnalogHistorical WHERE ID_AnalogVar="Temperatura" AND OwnerType="MET" AND OwnerCode=1 AND TimeStamp = (LECT MIN(TimeStamp) FROM AnalogHistorical WHERE ID_AnalogVar="Temperatura" AND OwnerType="MET" AND OwnerCode=1);

Limitar el número de elementos en la consulta

Atención, la sintaxis de este apartado es exclusiva de SQLite, en otras bases de datos se realizará de otra forma.

Se trata de limitar el número de elementos que queremos obtener, nos interesa por ejemplo coger los 10 más recientes.

SELECT * FROM AnalogHistorical Where ID_AnalogVar="Temperatura" AND OwnerType="MET" AND OwnerCode=1 ORDER BY TimeStamp LIMIT 10;

También podría interesarnos escoger 10 valores a partir de un offset dado:

SELECT * FROM AnalogHistorical Where ID_AnalogVar="Temperatura" AND OwnerType="MET" AND OwnerCode=1 ORDER BY TimeStamp LIMIT 10 OFFSET 2;