SQL con SQLite
Vistas e índices

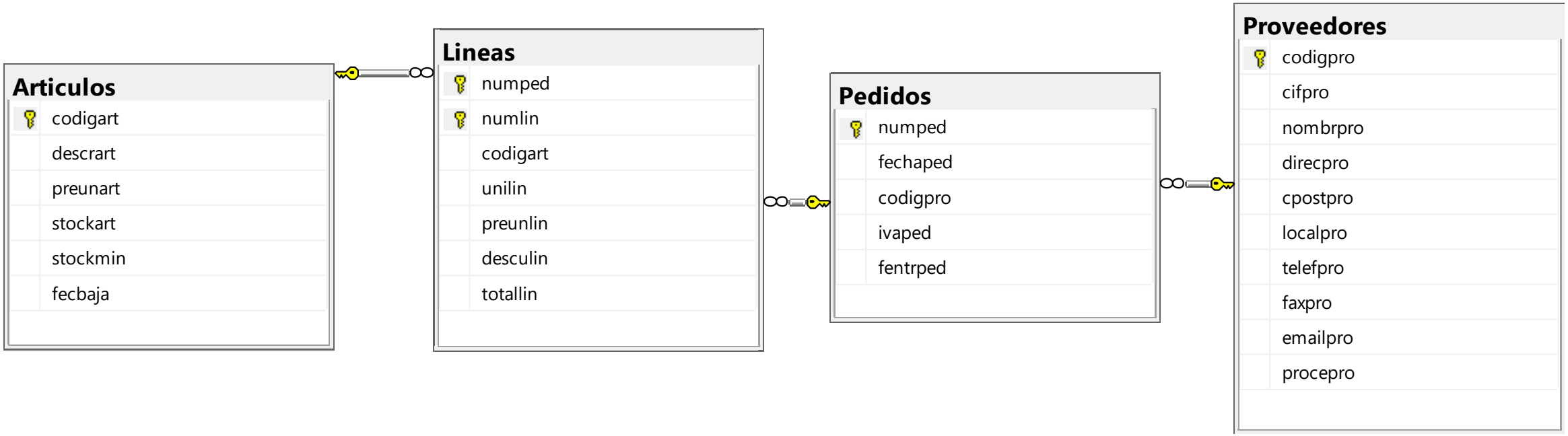
Máster en Data Science

M1967 - Modelos de Datos y Sistemas de Información 2018-2019

Diego García Saiz
Grupo de Ingeniería de Software y Tiempo Real
Departamento de Ingeniería Informática y Electrónica

Vistas e índices

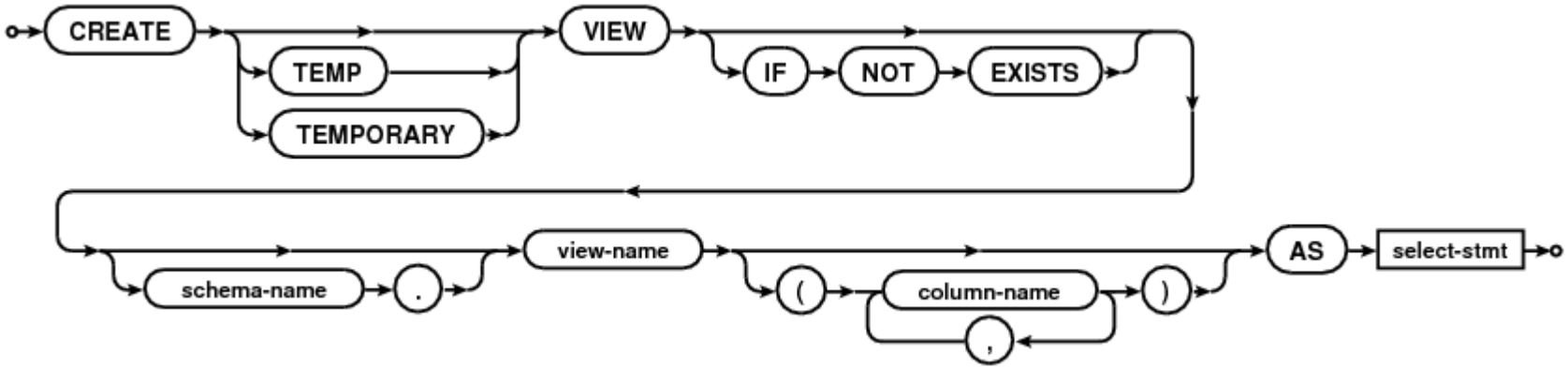
- En este tema, veremos como crear vistas e índices en bases de datos SQL.
- Trabajaremos con la siguiente base de datos, cuyos ficheros .sql para su creación e inserción de datos se encuentran colgados en Moodle:



SECCIÓN 1: VISTAS

Consultas en SQL: Vistas

- Las vistas son un mecanismo que permite encapsular consultas (SELECT) de forma lógica en la base de datos.
- Permite por tanto el almacenamiento y reutilización de consultas.
- Provee de una capa de seguridad, al poder ser usadas como interfaces para los diferentes usuarios.



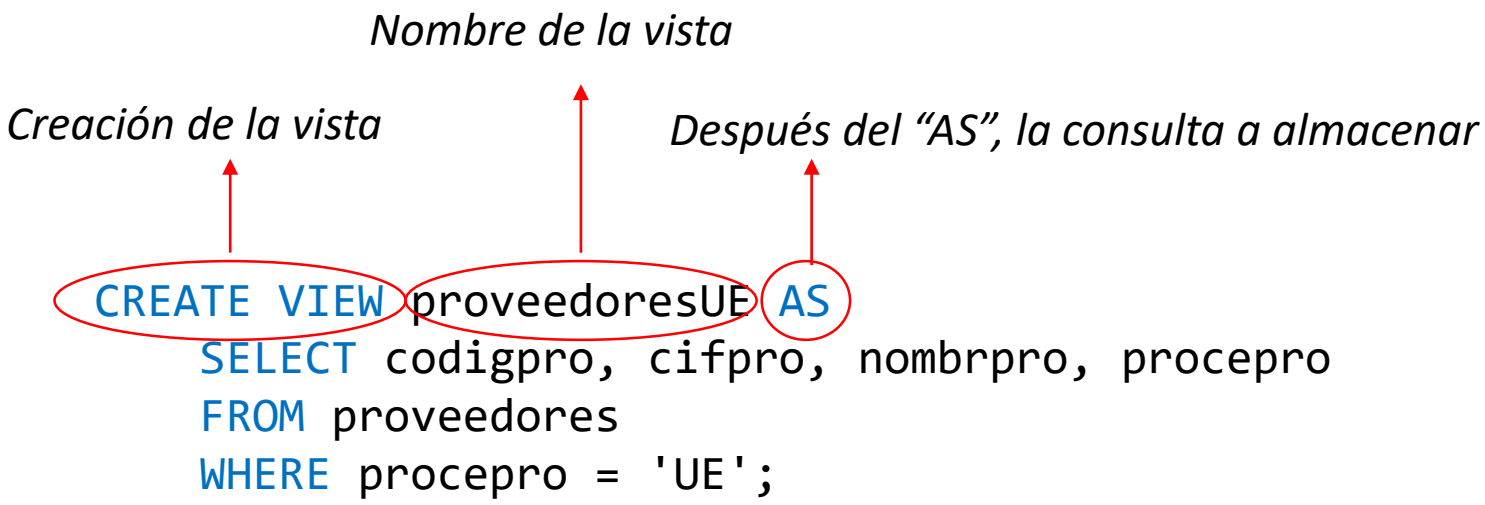
Consultas en SQL: Vistas

- Las vistas pueden encapsular “casi” cualquier consulta sobre una base de datos.
  - Ejemplo 1: crear una vista que almacene el código, cif, nombre y procedencia de todos los proveedores que sean de la UE.

```
CREATE VIEW proveedoresUE AS
    SELECT codigpro, cifpro, nombrpro, procepro
    FROM proveedores
    WHERE procepro = 'UE';
```

Consultas en SQL: Vistas

- Las vistas pueden encapsular “casi” cualquier consulta sobre una base de datos.
  - Ejemplo 1: crear una vista que almacene el código, cif, nombre y procedencia de todos los proveedores que sean de la UE.



Consultas en SQL: Vistas

- Las vistas pueden encapsular “casi” cualquier consulta sobre una base de datos.
  - Ejemplo 1: crear una vista que almacene el código, cif, nombre y procedencia de todos los proveedores que sean de la UE.

```
CREATE VIEW proveedoresUE AS
    SELECT codigpro, cifpro, nombrpro, procepro
    FROM proveedores
    WHERE procepro = 'UE';
```

Las vistas se usan en el FROM de una consulta, al igual que las tablas

```
SELECT pUE.* FROM proveedoresUE pUE;
```

	codigpro	cifpro	nombrpro	procepro
1	P001	A39184215	Bau Pi, Pablo	UE
2	P002	A48162311	Zar Luna, Ana	UE
3	P003	B28373212	Gras Leon, Luz	UE
4	P004	B85392314	Gil Laso, Luis	UE

Consultas en SQL: Vistas

- Las vistas pueden encapsular “casi” cualquier consulta sobre una base de datos.
  - Ejemplo 2: crear una vista que devuelva los datos del proveedor ‘P001’, junto con los datos de sus pedidos.

```
CREATE VIEW proveedoresPedidos AS
    SELECT pr.*, pe.*
    FROM Proveedores pr
        JOIN Pedidos pe
    ON pr.codigpro = pe.codigpro
    WHERE pr.codigpro = ‘P001’;
```

Consultas en SQL: Vistas

- Las vistas pueden encapsular “casi” cualquier consulta sobre una base de datos.
  - Ejemplo 3: crear una vista que devuelva la media, máximo y mínimo precio pagado por cada artículo

```
CREATE VIEW articulosAgrupacion AS
SELECT l.codigart, MAX(l.preunlin) maximo, MIN(l.preunlin) minimo,
       AVG(l.preunlin) media
FROM Lineas l
GROUP BY l.codigart;
```

Consultas en SQL: Vistas

- Las vistas pueden encapsular “casi” cualquier consulta sobre una base de datos.
  - Ejemplo 4: crear una vista que devuelva los datos de los artículos ordenados por stock disponible

```
CREATE VIEW articulosOrdenados AS
SELECT a.*
FROM Articulos a
ORDER BY a.stockart;
```

**Algunos gestores, como T-SQL, no permiten crear vistas con ORDER BY*

Consultas en SQL: Vistas

- Las vistas pueden ser temporales. En ese caso, son eliminadas cuando cerramos la conexión con la base de datos.
  - Ejemplo: crear una vista temporal que devuelva los datos de los artículos ordenados por stock disponible

```
CREATE TEMP VIEW articulosOrdenados AS
SELECT a.*
FROM Articulos a
ORDER BY a.stockart;
```

**Algunos gestores, como T-SQL, no permiten crear vistas con ORDER BY*

Consultas en SQL: Vistas

- En SQLite, las vistas sólo tienen permisos de consulta.

- Esto significa que se pueden hacer SELECTs sobre ellas, como hemos visto, pero en principio, no se pueden realizar INSERTs, UPDATEs ni DELETEs.

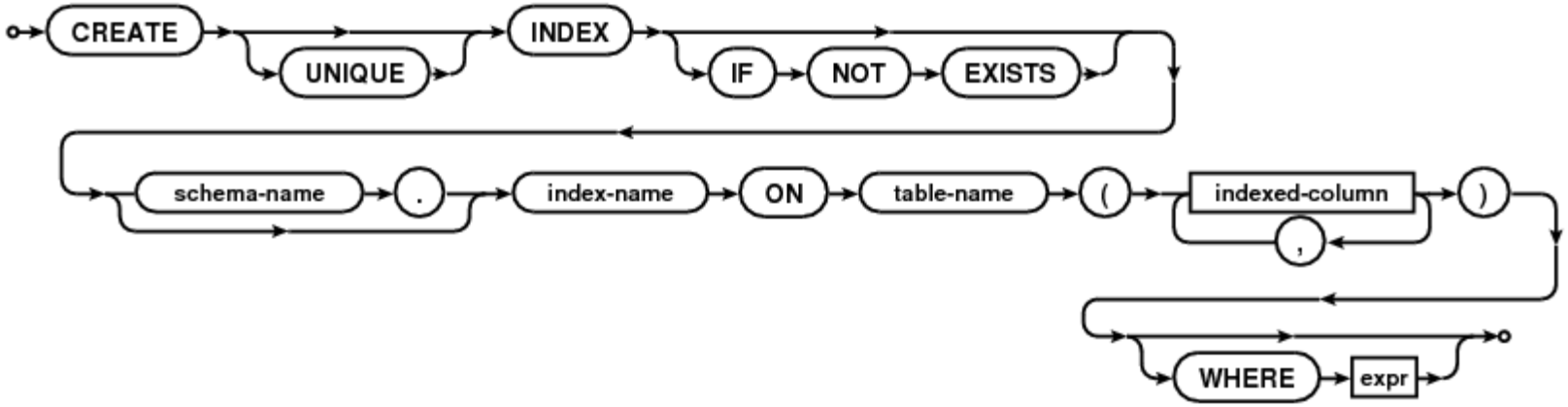
  - No obstante, se puede utilizar un mecanismo conocido como disparador (TRIGGER) de tipo INSTEAD OF para habilitar los UPDATEs sobre las vistas.
  - En otros gestores, si se permiten estas operaciones. En ese caso, lo que se actualiza a través de la vista son los datos de las tablas en las que se basa.

AHORA TOCA HACER EL EJERCICIO 10

SECCIÓN 2: ÍNDICES

Consultas en SQL: Índices

- Los índices son estructuras de datos que nos permiten agilizar las consultas sobre una tabla.
- Se crean sobre un conjunto de campos, por lo que se agilizan las consultas que tienen condiciones que involucran a esos campos.
- ¡OJO!, al agilizar las consultas con los índices, los insertados, borrados y actualizados se vuelven más lentos, ya que cada una de estas operaciones puede requerir la reconstrucción del índice.



Consultas en SQL: Índices

- Imaginemos que queremos agilizar las consultas que se realizan en la tabla Proveedores cuando filtran por su procedencia:

```
SELECT codigpro, cifpro, nombrpro, procepro
    FROM proveedores
    WHERE procepro = 'UE';
```

- La anterior consulta se puede agilizar creando un índice sobre el campo ‘procepro’:

```
CREATE INDEX idx_procepro ON proveedores (procepro);
```

Consultas en SQL: Índices

- Los índices almacenan punteros a los datos con un ordenamiento, que puede ser ascendente o descendente. Imaginemos que queremos retornar a los proveedores ordenados de forma descendente por nombre:

```
SELECT codigpro, cifpro, nombrpro, procepro
    FROM proveedores
    ORDER BY nombrpro DESC;
```

- La anterior consulta se puede agilizar creando un índice sobre el campo ‘nombrpro’ ordenado de forma descendente:

```
CREATE INDEX idx_nombrpro ON proveedores (nombrpro DESC);
```

- Si no decimos nada, por defecto el ordenamiento es ascendente.

Consultas en SQL: Índices

- Los índices pueden ser creados sobre varios campos. Por ejemplo, imaginemos que la siguiente consulta es frecuente sobre la base de datos:

```
SELECT codigpro, cifpro, nombrpro, procepro
    FROM proveedores
    ORDER BY nombrpro DESC, cifpro ASC;
```

- La anterior consulta se puede agilizar creando un índice doble sobre los campos ‘nombrpro’ y ‘cifpro’:

```
CREATE INDEX idx_nombrcifpro ON proveedores (nombrpro DESC, cifpro ASC);
```

Consultas en SQL: Índices

- Los índices pueden ser únicos. El siguiente índice fuerza a que en la tabla de proveedores no pueda haber dos filas con el mismo valor en el campo ‘nombrpro’

```
CREATE UNIQUE INDEX idx_nombrUNIQUE ON proveedores (nombrpro);
```

- Tiene el mismo efecto que el incluir la restricción UNIQUE(nombrpro) al crear la tabla.
  - De hecho, al crear esta restricción, lo que el sistema hace de forma implícita es crear un índice.
  - Cuando creamos una PRIMARY KEY, también se crea un índice, que en este caso define el orden físico de almacenamiento de las filas. Por ello, no se pueden crear varias PRIMARY KEY.

Consultas en SQL: Índices

- Los índices pueden ser únicos. El siguiente índice fuerza a que en la tabla de proveedores no pueda haber dos filas con el mismo valor en el campo ‘nombrpro’

```
CREATE UNIQUE INDEX idx_nombrUNIQUE ON proveedores (nombrpro);
```

- Tiene el mismo efecto que el incluir la restricción UNIQUE(nombrpro) al crear la tabla.
  - De hecho, al crear esta restricción, lo que el sistema hace de forma implícita es crear un índice.
  - Cuando creamos una PRIMARY KEY, también se crea un índice, que en este caso define el orden físico de almacenamiento de las filas. Por ello, no se pueden crear varias PRIMARY KEY.

Consultas en SQL: Índices y planes de ejecución

- El efecto de los índices lo podemos ver con el comando ‘EXPLAIN QUERY PLAN’.
- Por ejemplo, ejecutemos la siguiente línea:

```
EXPLAIN QUERY PLAN
SELECT *
    FROM proveedores
    ORDER BY cpostpro DESC;
```

- Obtenemos el siguiente resultado. ‘SCAN TABLE’ significa que, al no haber índices sobre los campos involucrados en la consulta, la tabla es escaneada de forma secuencial:

	id	parent	notuse(	detail
1	3	0	0	SCAN TABLE proveedores
2	18	0	0	USE TEMP B-TREE FOR ORDER BY

Consultas en SQL: Índices y planes de ejecución

- El efecto de los índices lo podemos ver con el comando ‘EXPLAIN QUERY PLAN’.
- Creemos ahora un índice sobre el campo del ORDER BY, y ejecutemos de nuevo la consulta:

```
CREATE INDEX idx_cpostpro ON proveedores (cpostpro);
EXPLAIN QUERY PLAN
SELECT *
    FROM proveedores
    ORDER BY cpostpro DESC;
```

- Obtenemos el siguiente resultado. Esta vez ‘SCAN TABLE’ se acompaña del uso de un índice:

	id	parent	notused	detail
1	4	0	0	SCAN TABLE proveedores USING INDEX idx_cpostpro

Consultas en SQL: Índices y planes de ejecución

- El efecto de los índices lo podemos ver con el comando ‘EXPLAIN QUERY PLAN’.
- Probemos ahora a usar la Primary Key (PK) en los criterios de búsqueda:

```
EXPLAIN QUERY PLAN
SELECT *
    FROM proveedores
    WHERE codigpro = ‘P001’;
```

- Al existir un índice predefinido sobre la PK, este es utilizado para agilizar la búsqueda:

	id	parent	notused	detail
1	3	0	0	SEARCH TABLE proveedores USING INDEX sqlite_autoindex_Proveedores_1 (codigpro=?)

AHORA TOCA HACER EL EJERCICIO 11