

Temario

- ♦ Introducción y fundamentos

- ♦ Introducción a SQL

Elmasri cap. 8

Postgres (<http://www.postgresql.org/docs/12>)

- ♦ Modelo Entidad / Relación

- ♦ Modelo relacional

- ♦ Diseño relacional: formas normales

- ♦ Consultas

- Cálculo relacional
- Álgebra relacional

- ♦ Implementación de bases de datos

- Estructura física: campos y registros
- Indexación
 - Índices simples
 - Árboles B, B* y B+

Structured Query Language – SQL

- ♦ Lenguaje de “programación” para SGBDs
 - Data Definition Language: creación del modelo de datos (diseño de tablas)
 - Data Manipulation Language: inserción, modificación, eliminación de datos
 - Data Query Language: consultas
 - Data Control Language: control
- ♦ Se ejecuta sobre un SGBD
- ♦ El estándar más utilizado
 - Creado en 1974 (D. D. Chamberlin & R. F. Boyce, IBM)
 - ANSI en 1986, ISO en 1987
 - Core (todos los SGBD) + packages (módulos opcionales)
- ♦ Versiones
 - SQL1 – SQL 86
 - SQL2 – SQL 92
 - SQL3 – SQL 1999, no plenamente soportado por la industria (recursión, programación, objetos...)
- ♦ Limitaciones
 - Importantes divergencias entre implementaciones (no es directamente portable en general, incompletitudes, extensiones) –uno termina aprendiendo variantes de SQL



Donald D.
Chamberlin

Elementos de una base de datos SQL

- ♦ Base de datos = conjunto de tablas
- ♦ Tabla (relación, entidad, esquema...) =
 - Estructura fija de campos (esquema)
 - Conjunto de registros con valores de campos
- ♦ Campo (atributo, propiedad, “columna”), tiene un tipo de dato
- ♦ Registro (tupla, “fila”)
- ♦ Clave primaria
- ♦ Claves externas

Estructura léxica del lenguaje

Operaciones SQL

- ♦ DDL – Creación, diseño, modificación y eliminación de tablas (CREATE, ALTER, DROP)
- ♦ DML – Inserción, modificación, eliminación de registros (INSERT, UPDATE, DELETE)
- ♦ DQL – Consulta (SELECT)
- ♦ DCL – Control (GRANT, REVOKE)

Estructura léxica de SQL

- ♦ Case-insensitive, insignificant whitespace
- ♦ Sentencias, expresiones, valores, tipos de datos

Ejemplo: BD para aplicación de música

The screenshot displays the Spotify Premium desktop application. The interface is dark-themed with a green accent color. At the top, the Spotify logo and 'Spotify Premium' are visible. Below the menu bar (File, Edit, View, Playback, Help), there is a search bar and navigation icons. The left sidebar contains 'Stations', 'Local Files', and a 'PLAYLISTS' section with various playlist thumbnails. The main content area shows a playlist titled 'chosen just for you' with a description 'chosen just for you. Updated every Monday, so save your favourites!' and 'Created by: spotifydiscover • 30 songs, 1 hr 57 min'. Below the title are buttons for 'PLAY', 'FOLLOWING', and 'CREATE SIMILAR PLAYLIST'. A 'Filter' search bar and an 'Available Offline' toggle are also present. The playlist table lists songs with their titles, artists, and release dates. At the bottom, a playback bar shows the current song 'Piano S' by Ludwig v, with a progress bar and volume controls. A green bar at the very bottom states 'You are listening on'.

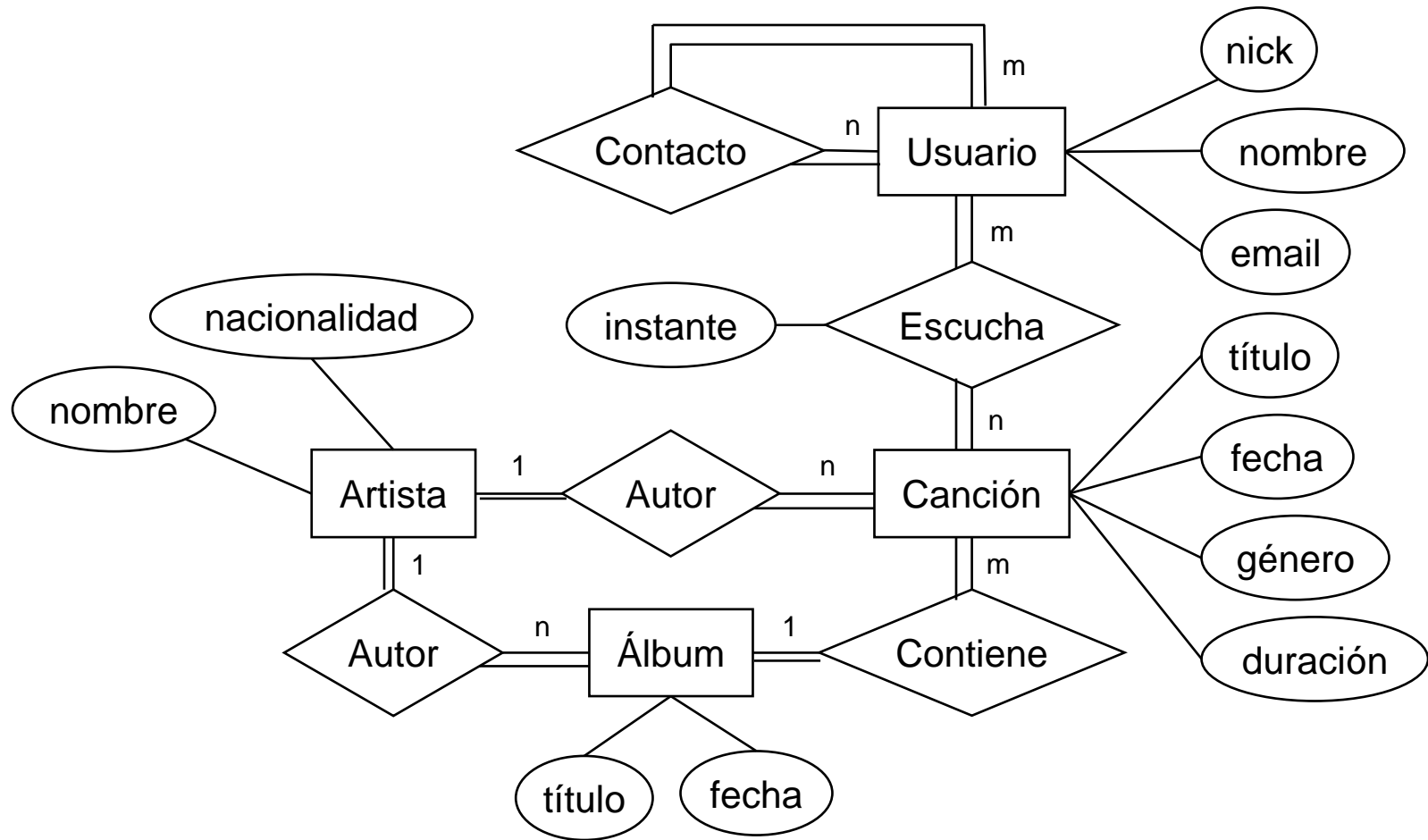
| | SONG | ARTIST | |
|---|-------------------------|---------------------------|------------|
| + | Same To You | Melody Gardot | 2 days ago |
| + | Holding Up - Radio Edit | Jabberwocky, Na Kyung Lee | 2 days ago |
| + | Palmar | Caloncho | 2 days ago |
| + | Sunny | Bryan Adams | 2 days ago |
| + | Jour 1 - Gostan Remix | Louane | 2 days ago |
| + | Quicksand | Caro Emerald | 2 days ago |

Ejemplo: descripción informal

Aplicación de música online con red social

- ◆ Tipos de datos: usuarios, canciones, discos, autores...
- ◆ Estructuras:
 - Los usuarios tienen nick, nombre, email...
 - Las canciones tienen título, género, duración, fecha...
 - Los artistas tienen nombre, nacionalidad...
- ◆ Relaciones:
 - Las canciones tienen autores, los discos tienen canciones, los usuarios tienen amigos, discos favoritos, escuchan canciones...
- ◆ Funcionalidades:
 - Buscar una canción, escucharla, ver sus datos...
 - Ver / añadir amigos...

Ejemplo: diagrama ER



Artista

| id | nombre | nacionalidad |
|----|--------------------|--------------|
| 0 | The Beatles | UK |
| 1 | The Rolling Stones | UK |
| 2 | David Bowie | UK |

Ejemplo: vista tablas

Canción

| id | título | genero | duración | fecha | autor | álbum |
|----|----------------------------|--------|----------|------------|-------|-------|
| 0 | Norwegian Wood | Pop | 125 | 1965-03-12 | 0 | 0 |
| 1 | Here, there and everywhere | Pop | 145 | 1969-08-05 | 0 | 1 |
| 2 | Jumping jack flash | Pop | 225 | 1968-04-20 | 1 | 2 |

Usuario

| nick | nombre | email |
|-------|------------|-------------------|
| lola | Dolores | lola@gmail.com |
| pepe | José | jose@gmail.com |
| chema | José María | chema@gmail.com |
| charo | Rosario | rosario@gmail.com |

Contacto

| usuario1 | usuario2 |
|----------|----------|
| pepe | lola |
| charo | pepe |
| chema | charo |
| pepe | chema |

Escucha

| usuario | canción | instante |
|---------|---------|---------------------|
| charo | 1 | 2011-09-09 16:57:54 |
| pepe | 2 | 2011-09-12 21:15:30 |

Álbum

| id | autor | nombre | fecha |
|----|-------|-----------------|-------|
| 0 | 0 | Rubber Soul | 1965 |
| 1 | 0 | Revolver | 1966 |
| 2 | 2 | Beggars Banquet | 1968 |

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones del 69 (mostrar título y género)

Conjunto de nacionalidades de los artistas en la BD

Canciones de artistas del Reino Unido

Título de las canciones escuchadas por un usuario

Todos los contactos de un usuario dado

Usuarios que se llaman igual

Usuarios a distancia 2 de un usuario dado en la red social

Contactos comunes a dos usuarios

Cuántas veces ha sido escuchada una canción

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Data Definition Language

```
CREATE TABLE nombre (  
    campo1 tipo1 [restricciones1],  
    campo2 tipo2 [restricciones2],  
    ...,  
    [restricciones]  
);
```

Comandos ALTER útiles cuando
ya hay una tabla creada y con datos



```
ALTER TABLE nombre ADD COLUMN campo tipo [restricciones];
```

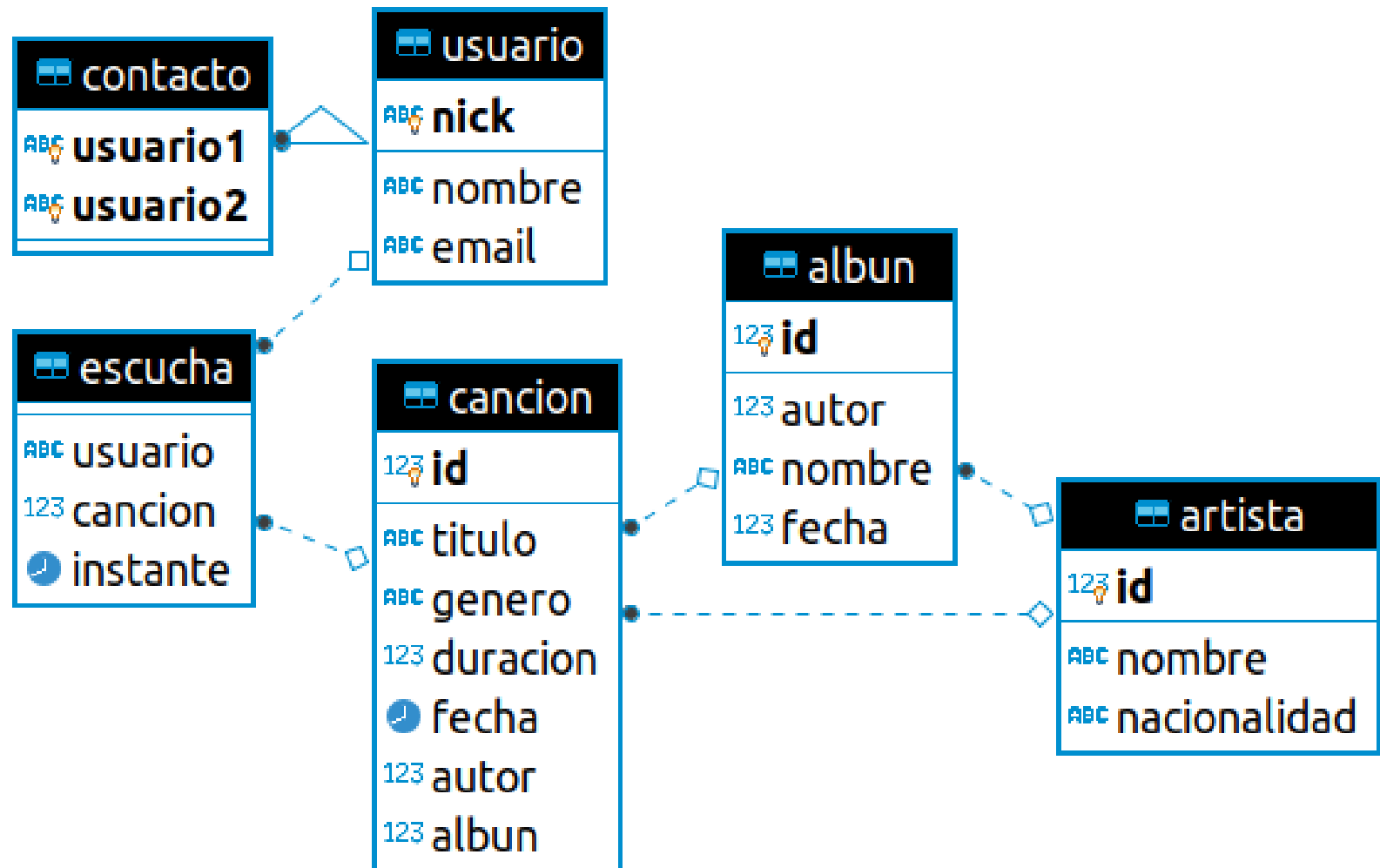
```
ALTER TABLE nombre ADD restricción;
```

```
ALTER TABLE nombre DROP COLUMN campo;
```

```
DROP TABLE nombre;
```

```
DROP CONSTRAINT nombre-restricción;
```

Ejemplo: 2-musica.sql



Restricciones

En un campo

NOT NULL

UNIQUE

PRIMARY KEY

REFERENCES *tabla* (*clave*) [(ON DELETE | ON UPDATE)

(NO ACTION | RESTRICT | CASCADE
| SET NULL | SET DEFAULT)]

DEFAULT *valor*

Con nombre

CONSTRAINT *nombre* *restricción*

Si se omite, tomará
la clave primaria



En una tabla

PRIMARY KEY (*campo1*, *campo2*, ...)

FOREIGN KEY (*campo1*, *campo2*, ...) REFERENCES *tabla* (*clave1*, *clave2*, ...)

UNIQUE (*campo1*, *campo2*, ...)

CHECK (*expresión*)

Claves primarias

- ♦ Designan un identificador único de las filas de una tabla
- ♦ Sólo puede haber una clave primaria por tabla, aunque puede incluir varios campos
- ♦ Es muy aconsejable que toda tabla tenga su clave primaria
- ♦ Técnicamente equivalen a UNIQUE más NOT NULL
- ♦ Importante en la indexación (lo veremos más adelante)
- ♦ Opción de diseño: selección de clave primaria entre varias posibles
 - Clave primaria natural (email, dominio web, DNI, ISBN, etc.)
 - Clave primaria artificial: p.e. un ID entero (típicamente autoincremental), una cadena de caracteres (códigos), etc.

Claves externas

- ◆ Conceptualmente son comparables a punteros
- ◆ Referencian campos únicos de otra tabla
- ◆ Normalmente el campo referenciado es una clave primaria
 - O bien como mínimo tiene que ser declarado 'unique'
- ◆ Técnicamente no es imprescindible usarlas
- ◆ Ayudan a asegurar la consistencia en las referencias
 1. Generan un error cuando se intenta insertar o cambiar una clave externa por un valor que no existe en la tabla referenciada
 2. Permiten establecer qué se debe hacer cuando desaparece una clave referenciada
- ◆ En general es preferible (más eficiente) que sean de tipo entero

Claves: en resumen...

- ♦ La **clave primaria** de una tabla actúa como **identificador** de las filas
 - Juega un papel similar a la dirección de memoria RAM de un dato en C
 - Sólo hay una por tabla (aunque no es obligatorio, definir siempre una)
 - Puede ser un campo natural o (mejor) artificial (un entero en este caso)
 - Podría estar formada por varios campos
- ♦ Las **claves externas** juegan el papel de **punteros** entre filas de distintas (o las mismas) tablas
 - Deben apuntar a una clave primaria
 - Podrían apuntar también a un campo unique not null
 - Una clave externa puede estar formada por varios campos
- ♦ ¿Qué **ventaja** tiene declarar una **clave primaria**?
 - Obligar a que no se repitan valores en distintas filas
 - Diferencia con unique not null: cuestión de implementación (índices)
- ♦ ¿Qué **ventaja** tiene declarar **claves externas**?
 - Obligar a que su valor aparezca en alguna fila de la tabla apuntada
 - Reaccionar automáticamente a cambios en la clave primaria apuntada

Ejemplo: claves primarias y externas

Artista

| <u>id</u> | nombre | nacionalidad |
|-----------|--------------------|--------------|
| 0 | The Beatles | UK |
| 1 | The Rolling Stones | UK |
| 2 | David Bowie | UK |

Álbum

| <u>id</u> | autor | nombre | fecha |
|-----------|-------|-----------------|-------|
| 0 | 0 | Rubber Soul | 1965 |
| 1 | 0 | Revolver | 1966 |
| 2 | 2 | Beggars Banquet | 1968 |

Escucha

| usuario | canción | instante |
|---------|---------|---------------------|
| charo | 1 | 2011-09-09 16:57:54 |
| pepe | 2 | 2011-09-12 21:15:30 |

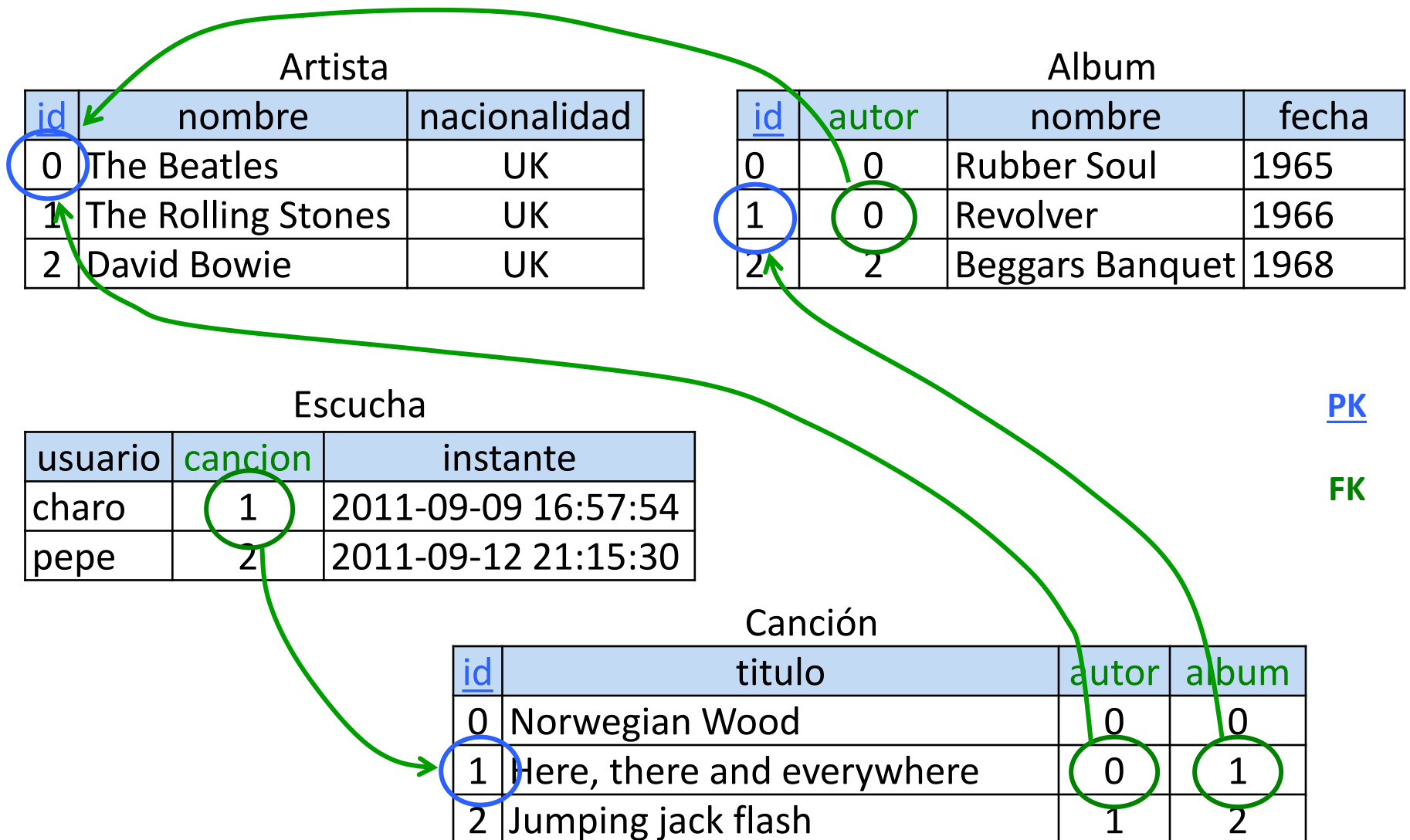
PK

FK

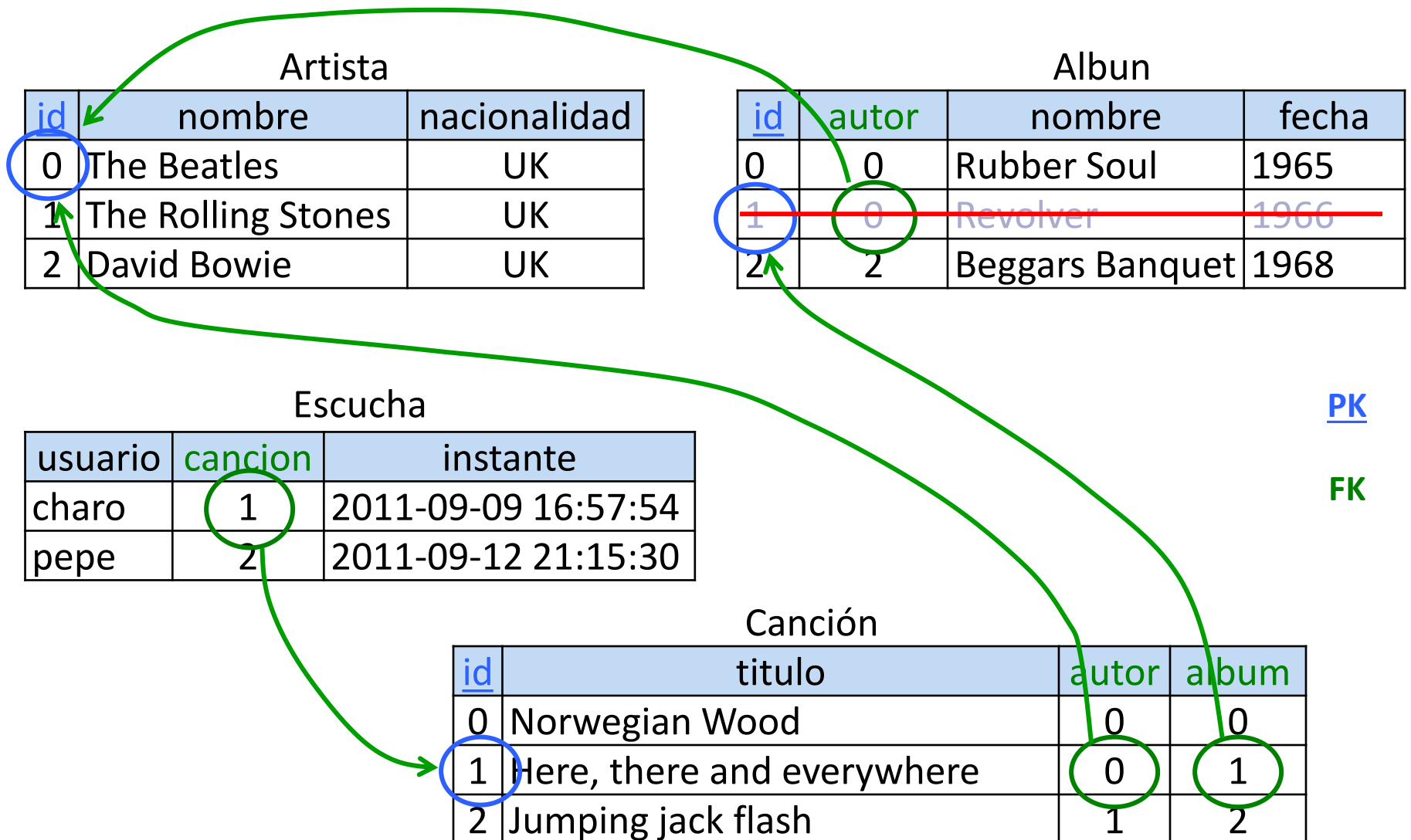
Canción

| <u>id</u> | título | autor | álbum |
|-----------|----------------------------|-------|-------|
| 0 | Norwegian Wood | 0 | 0 |
| 1 | Here, there and everywhere | 0 | 1 |
| 2 | Jumping jack flash | 1 | 2 |

Ejemplo: claves primarias y externas



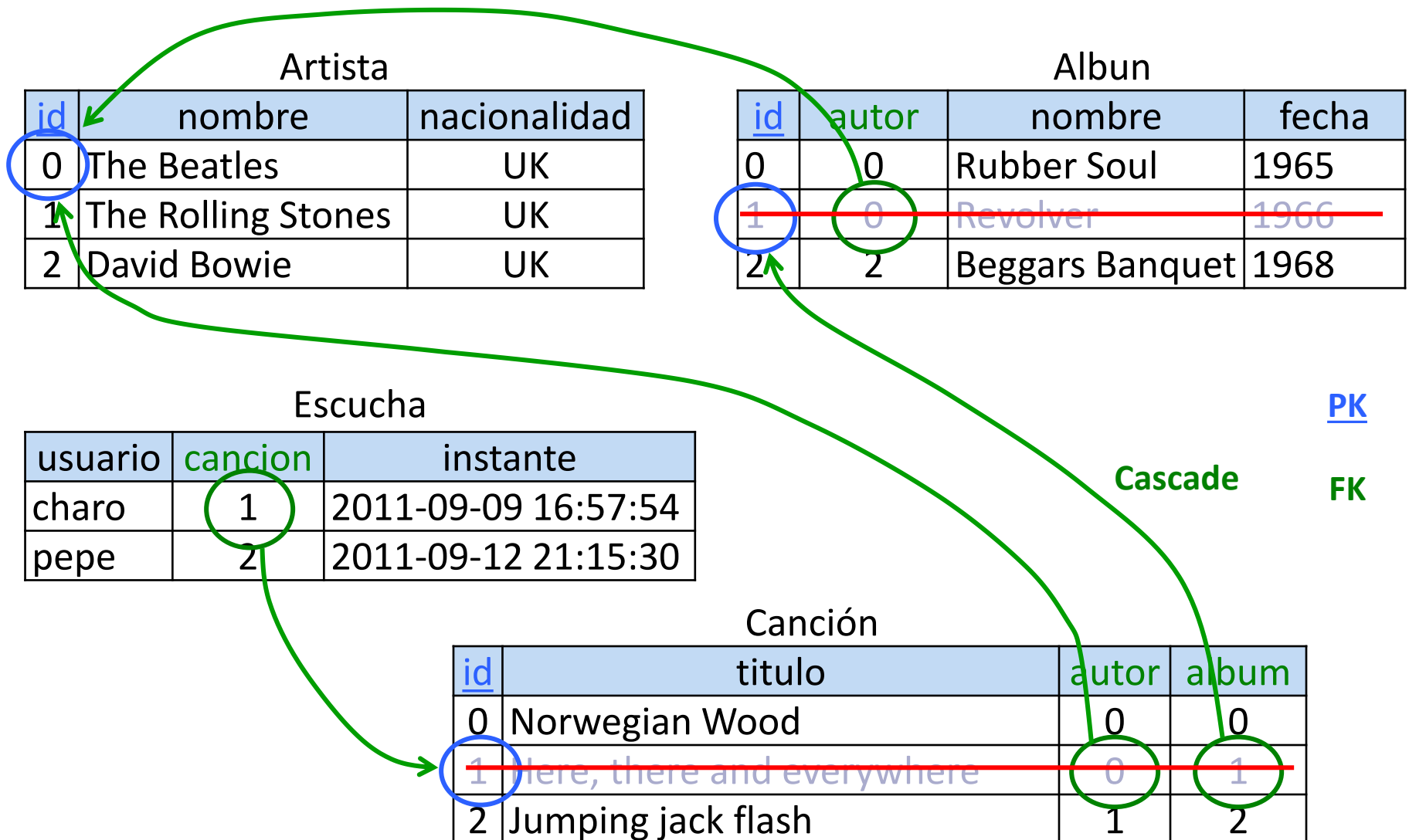
Ejemplo: on delete



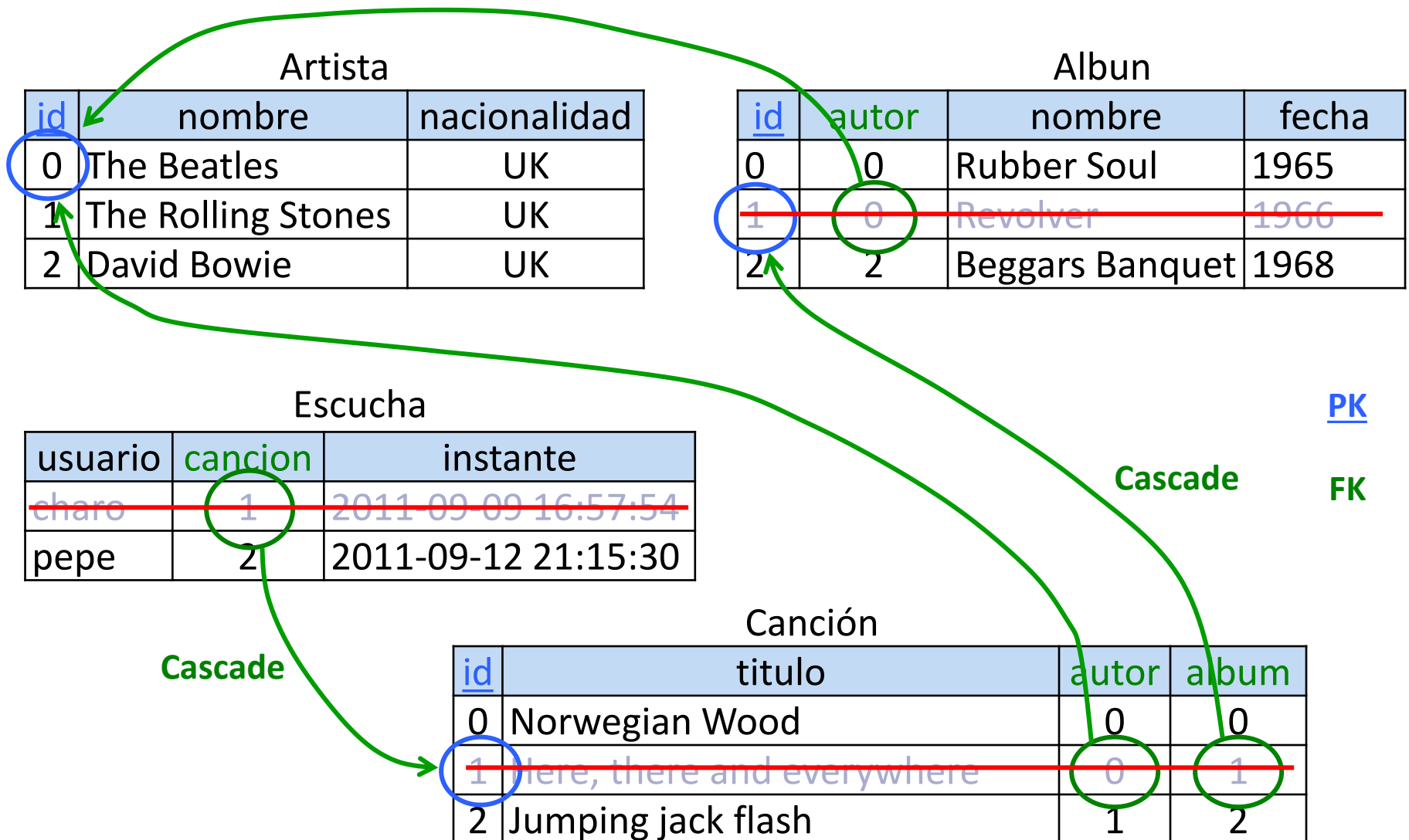
PK

FK

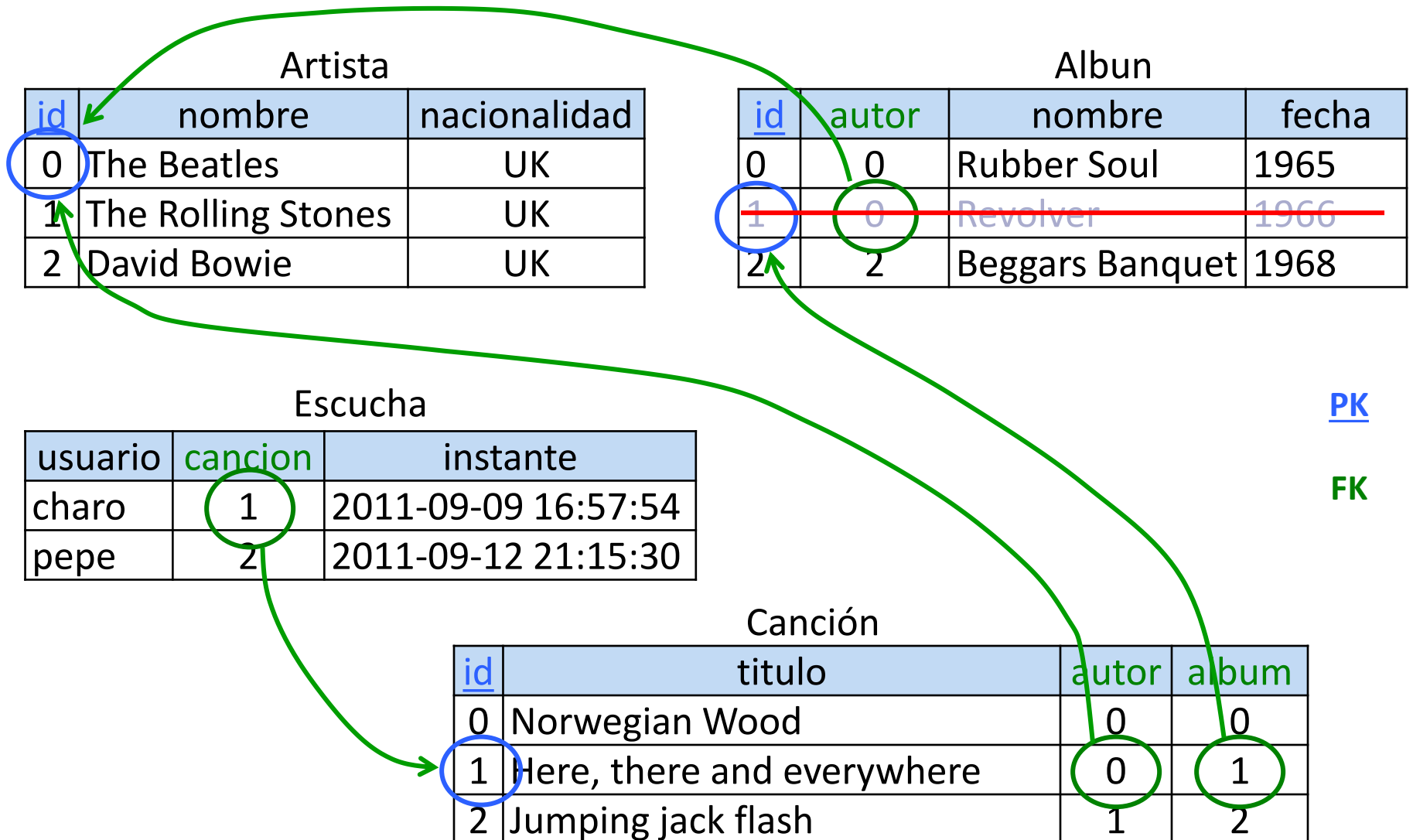
Ejemplo: on delete cascade



Ejemplo: on delete cascade



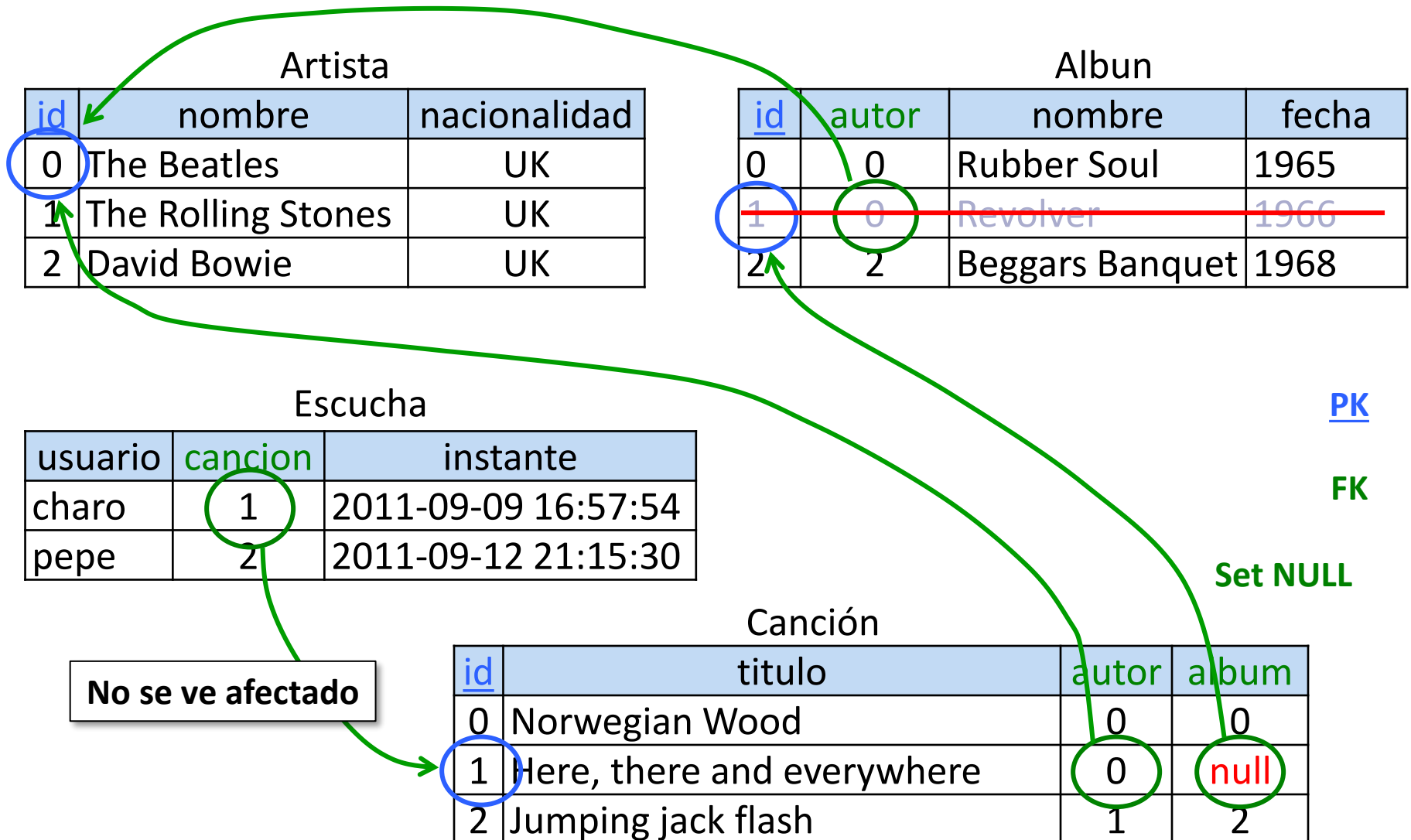
Ejemplo: on delete



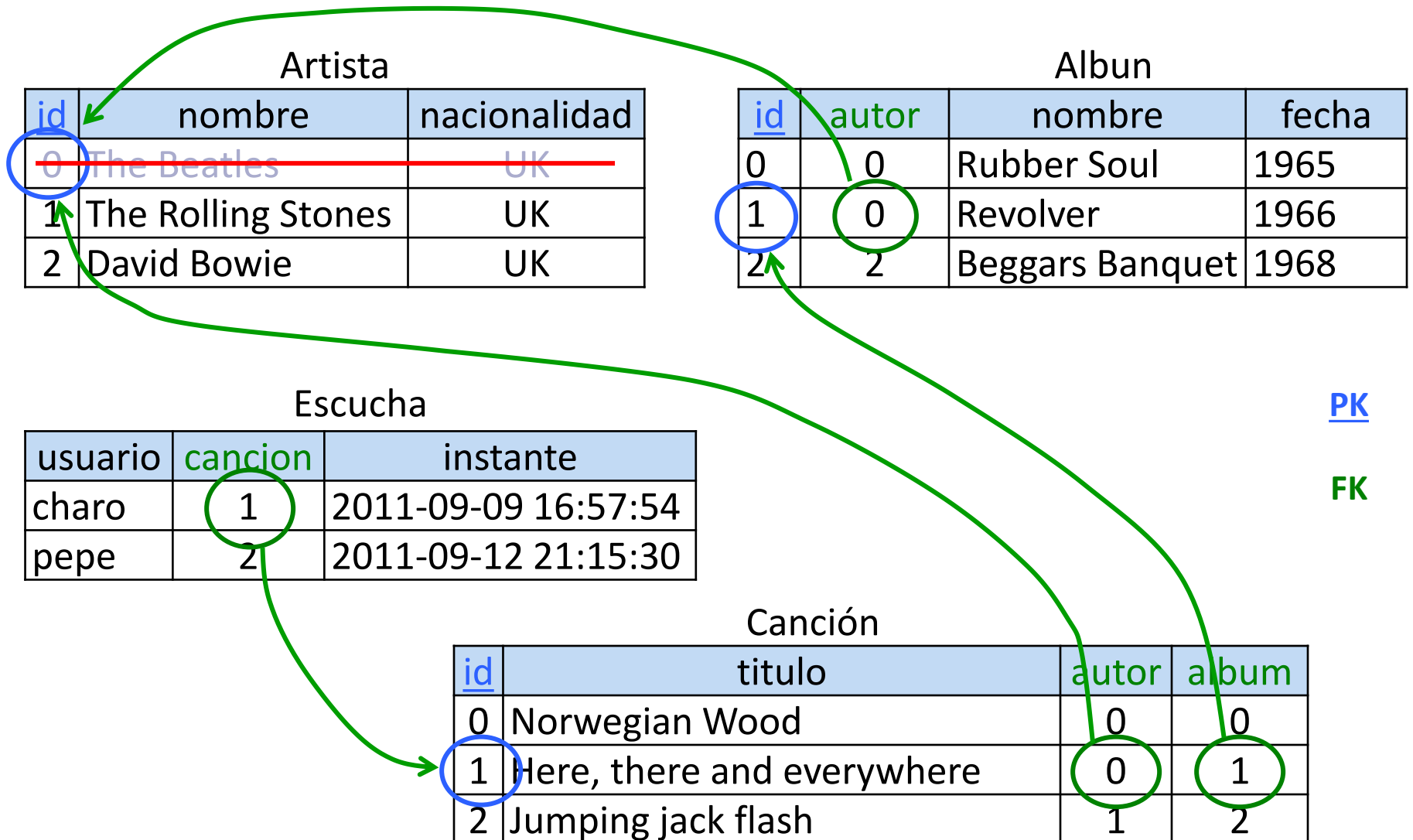
PK

FK

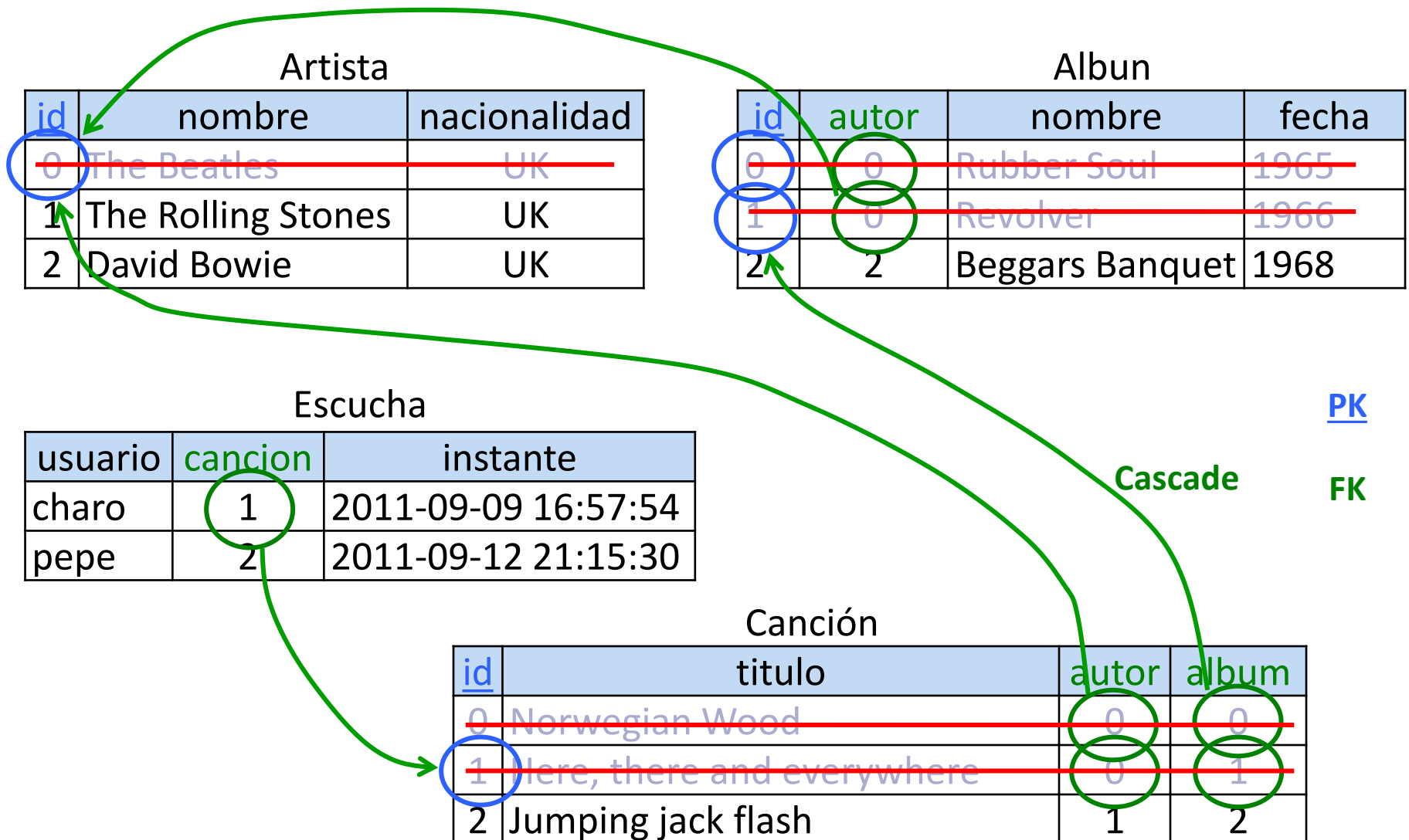
Ejemplo: on delete set null



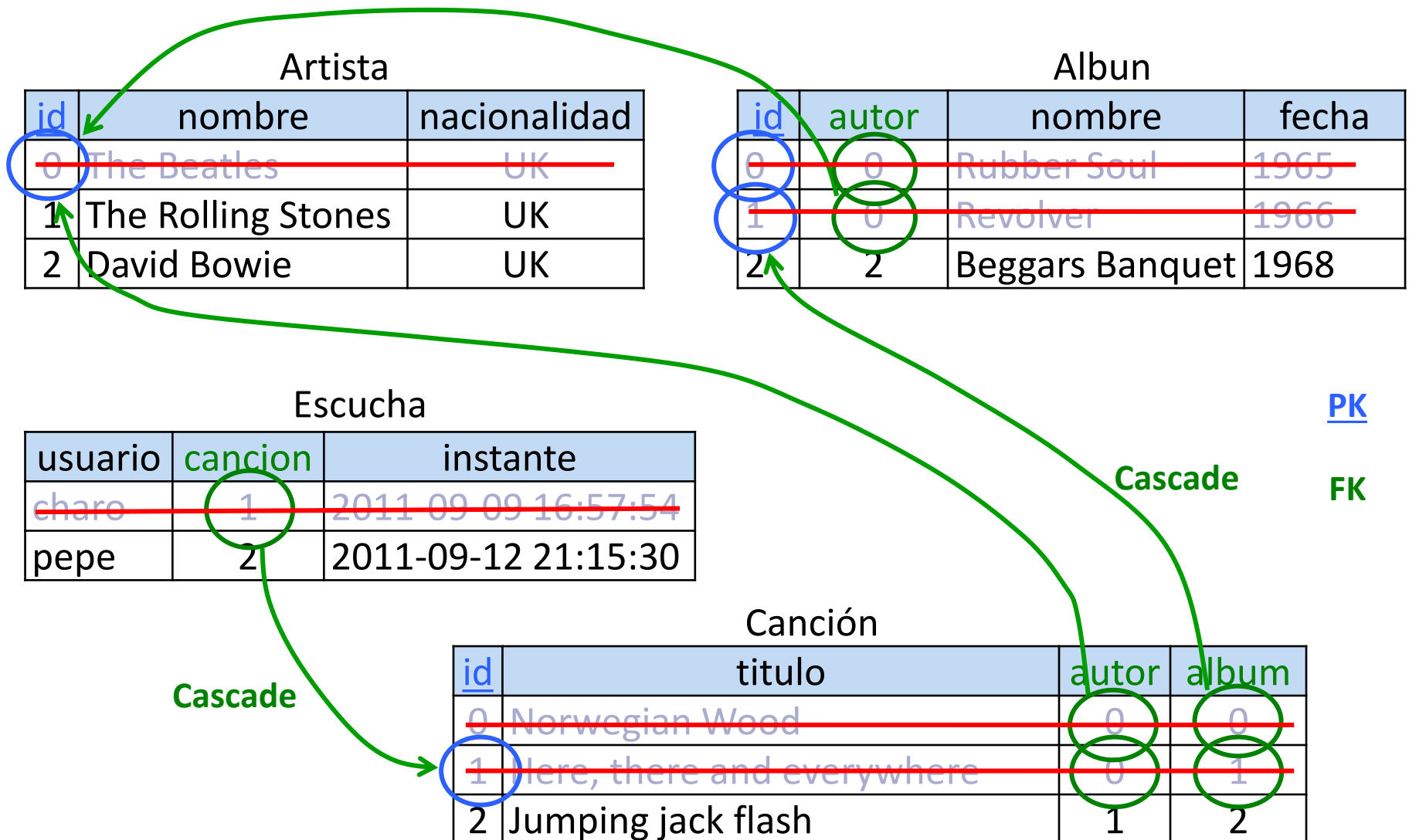
Ejemplo: on delete



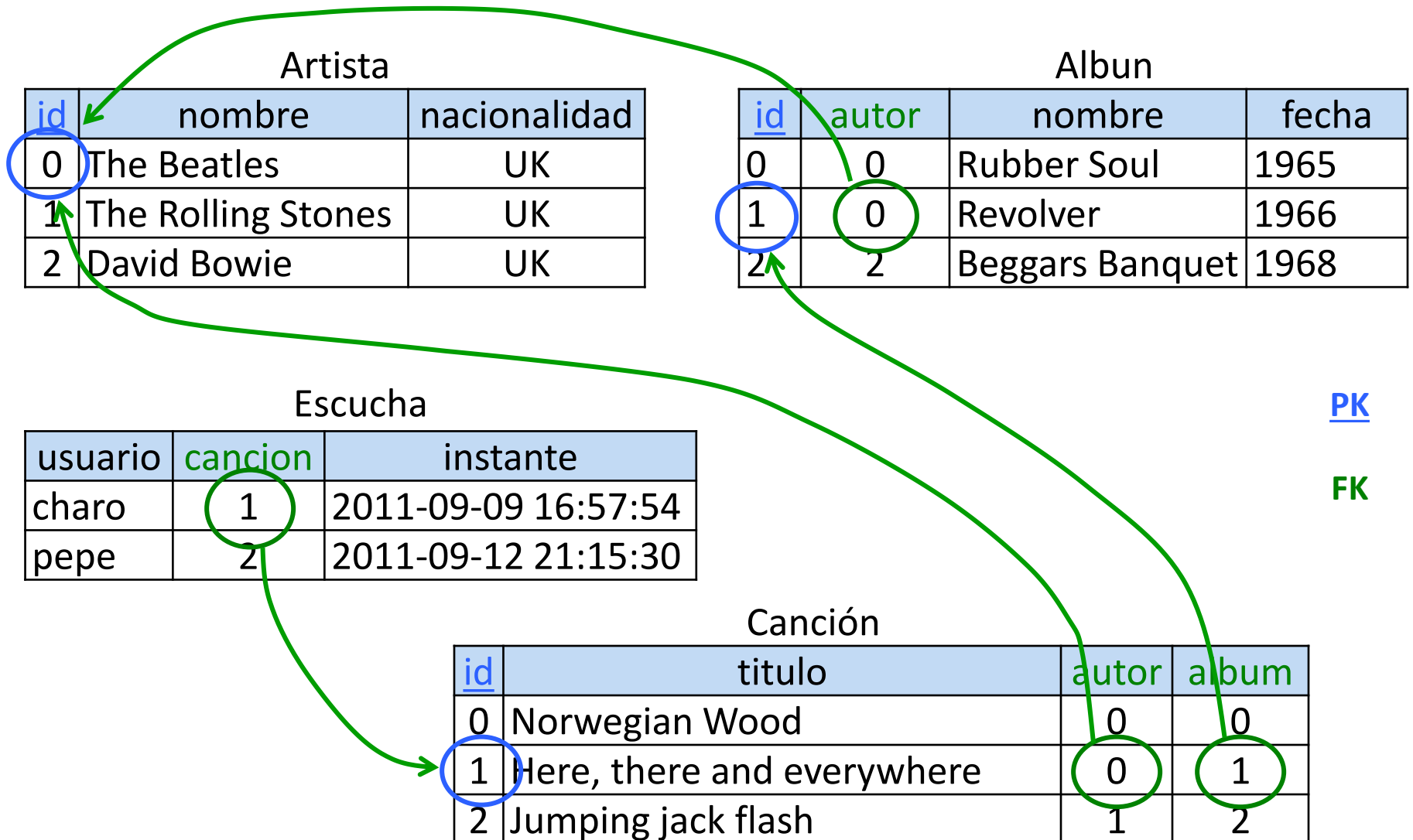
Ejemplo: on delete cascade



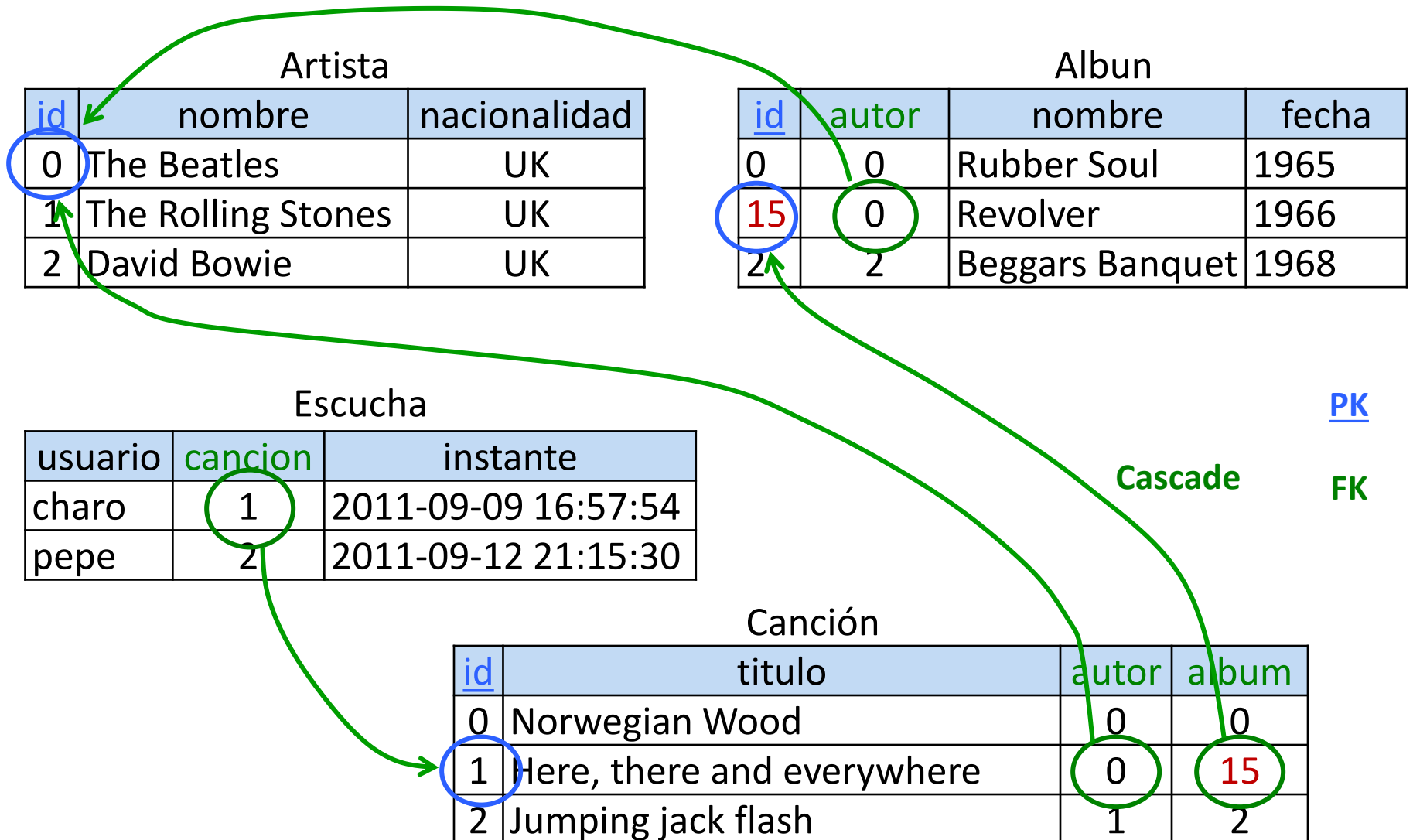
Ejemplo: on delete cascade



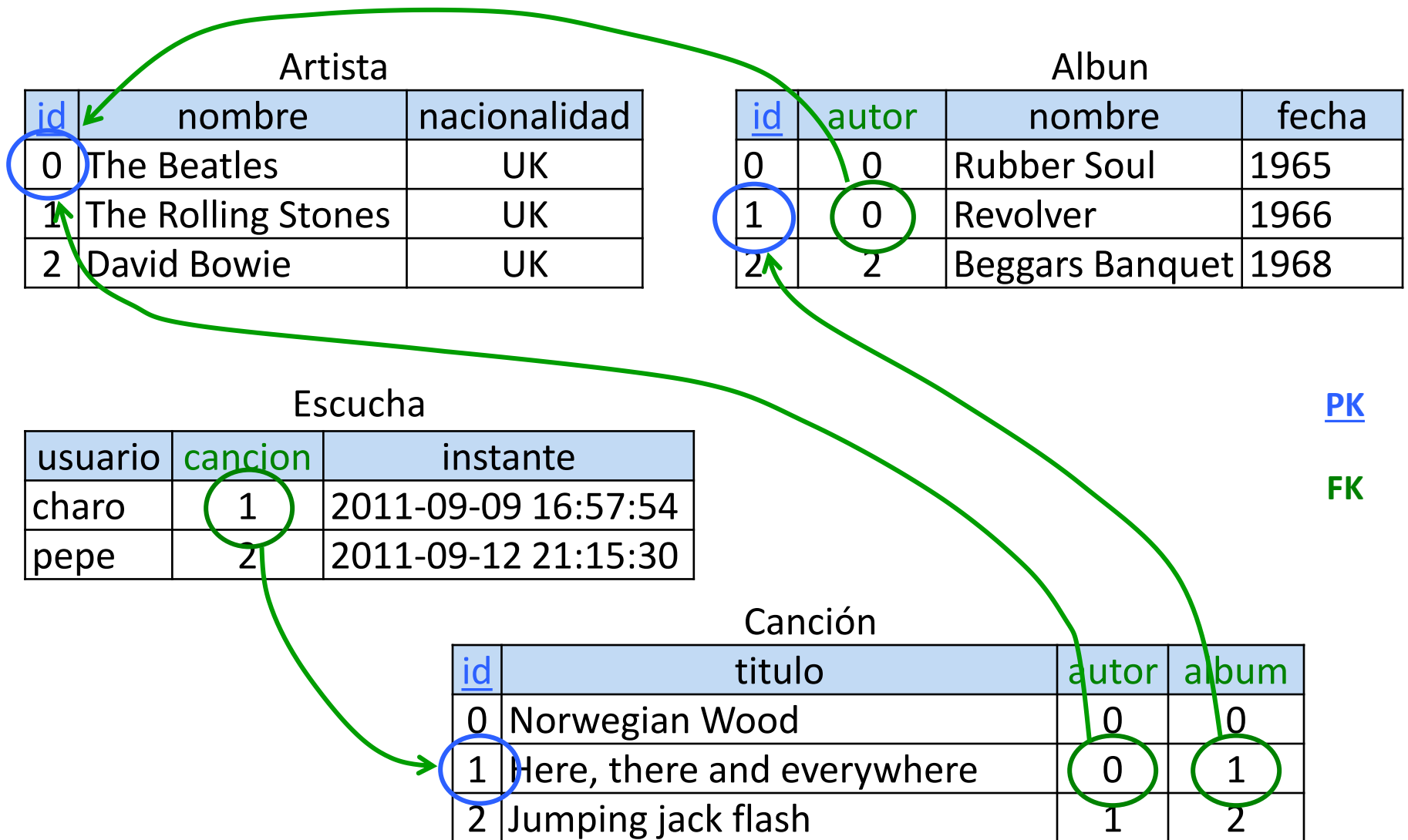
Ejemplo: on update



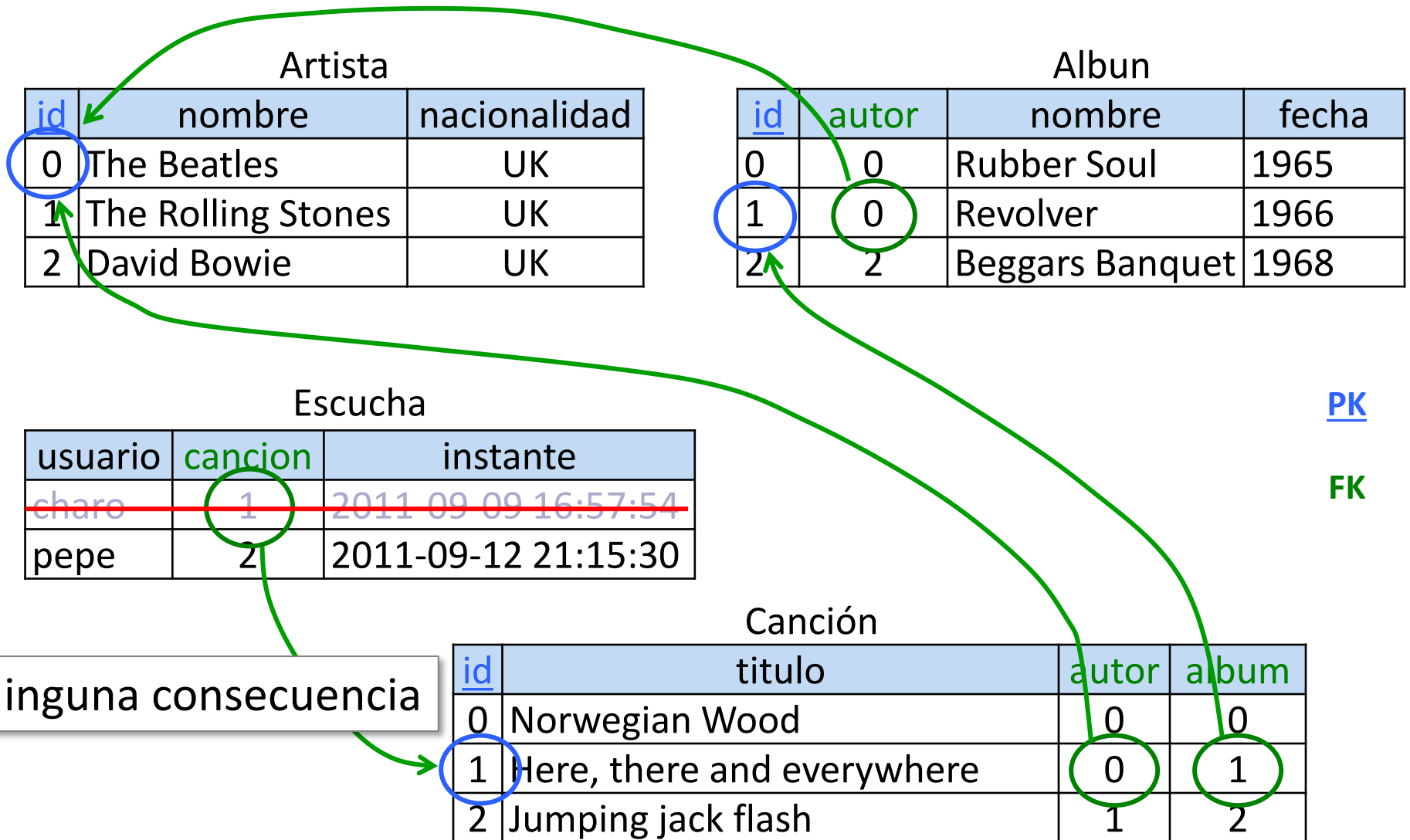
Ejemplo: on update



¿Y si se borran referencias?



¿Y si se borran referencias?



Data manipulation

INSERT INTO *tabla* [(*campo1*, *campo2*, ...)]

VALUES

(*valor11*, *valor12*, ...),

(*valor21*, *valor22*, ...),

...

También se puede anidar aquí una consulta:
se insertan las tuplas que devuelva

;

UPDATE *tabla* SET *campo1* = *valor1*, *campo2* = *valor2*, ...

/* PostgreSQL: [FROM ...] para formar condiciones con otras tablas */

[WHERE ...];

DELETE FROM *tabla*

/* PostgreSQL: [USING *tabla1*, *tabla2*, ...] para condiciones con otras tablas */

[WHERE ...];

TRUNCATE TABLE *tabla*;

Ejemplo

```
INSERT INTO Artista VALUES (0, 'The Beatles', 'UK');
```

```
INSERT INTO Artista VALUES (1, 'The Rolling Stones', 'UK');
```

```
INSERT INTO Artista (id, nombre) VALUES (2, 'David Bowie');
```

```
INSERT INTO Cancion VALUES
```

```
    (0, 'Norwegian wood', 'Pop', 125, '1965-03-12', 0),
```

```
    (1, 'Here, there and everywhere', 'Pop', 145, '1966-08-05', 0),
```

```
    (2, 'Jumping jack flash', 'Pop', 225, '1968-04-20', 1);
```

```
INSERT INTO Usuario VALUES
```

```
    ('lola', 'Dolores', 'lola@gmail.com'),
```

```
    ('pepe', 'José', 'jose@gmail.com'),
```

```
    ('chema', 'José María', 'chema@gmail.com'),
```

```
    ('charo', 'Rosario', 'rosario@gmail.com');
```

```
INSERT INTO Contacto VALUES
```

```
    ('pepe', 'lola'),
```

```
    ('charo', 'pepe'),
```

```
    ('chema', 'charo');
```

```
INSERT INTO Escucha VALUES
```

```
    ('charo', 2, '2011-09-09 16:57:54'),
```

```
    ('pepe', 3, '2011-09-12 21:15:30');
```

```
UPDATE Artista SET nacionalidad = 'UK' WHERE nombre = 'David Bowie';
```

```
UPDATE Album SET precio = precio * 1.2;
```

```
DELETE FROM Escucha WHERE instante < '2000-01-01 00:00:00';
```


Tipos y expresiones

Tipos SQL (Data Types PostgreSQL, chapter 8)


character(*n*) \equiv char(*n*), varchar(*n*), text

integer \equiv int, smallint

float, real, double precision

numeric (*precisión*, *escala*) \equiv decimal (*precisión*, *escala*)

date, time, timestamp


dígitos decimales
(por defecto 0)

Valores literales

Cadenas de caracteres entre '...'

Valores numéricos similar p.e. a C

date 'YYYY-MM-DD', time 'HH:MM:SS'

Operadores

+ - * / % ^

AND OR NOT

= < > <= >= LIKE ISNULL

operaciones con strings: concatenación, like, expresiones regulares ('%' '_')

Comentarios

--

/* ... */

Artista

| id | nombre | nacionalidad |
|----|--------------------|--------------|
| 0 | The Beatles | UK |
| 1 | The Rolling Stones | UK |
| 2 | David Bowie | UK |

Ejemplo: vista tablas

Canción

| id | título | genero | duración | fecha | autor | álbum |
|----|----------------------------|--------|----------|------------|-------|-------|
| 0 | Norwegian Wood | Pop | 125 | 1965-03-12 | 0 | 0 |
| 1 | Here, there and everywhere | Pop | 145 | 1969-08-05 | 0 | 1 |
| 2 | Jumping jack flash | Pop | 225 | 1968-04-20 | 1 | 2 |

Usuario

| nick | nombre | email |
|-------|------------|-------------------|
| lola | Dolores | lola@gmail.com |
| pepe | José | jose@gmail.com |
| chema | José María | chema@gmail.com |
| charo | Rosario | rosario@gmail.com |

Contacto

| usuario1 | usuario2 |
|----------|----------|
| pepe | lola |
| charo | pepe |
| chema | charo |
| pepe | chema |

Escucha

| usuario | canción | instante |
|---------|---------|---------------------|
| charo | 1 | 2011-09-09 16:57:54 |
| pepe | 2 | 2011-09-12 21:15:30 |

Álbum

| id | autor | nombre | fecha |
|----|-------|-----------------|-------|
| 0 | 0 | Rubber Soul | 1965 |
| 1 | 0 | Revolver | 1966 |
| 2 | 2 | Beggars Banquet | 1968 |

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de 1969 (mostrar título y género)

Conjunto de nacionalidades de los artistas en la BD

Canciones de artistas del Reino Unido

Título de las canciones escuchadas por un usuario

Todos los contactos de un usuario dado

Usuarios que se llaman igual

Usuarios a distancia 2 de un usuario dado en la red social

Usuarios sin contactos

Contactos comunes a dos usuarios

Cuántas veces ha sido escuchada una canción

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Data Query Language

SELECT [**DISTINCT**] *campos* **FROM** *tablas*
[**WHERE** *condición*];

Pueden ser expresiones
sobre campos

Ejemplos:

SELECT cancion.titulo, cancion.genero **FROM** cancion
WHERE cancion.fecha >='1969-01-01' **AND** cancion.fecha <= '1969-12-31';

SELECT DISTINCT artista.nacionalidad **FROM** Artista;

/ Varias tablas */*

SELECT * FROM Cancion, Artista

WHERE Cancion.autor = Artista.id **AND** Artista.nacionalidad = 'UK';

/ Expresiones */*

SELECT dni, teoria * 0.6 + practicas * 0.4 **FROM** Notas;

Artista

| id | nombre | nacionalidad |
|----|--------------------|--------------|
| 0 | The Beatles | UK |
| 1 | The Rolling Stones | UK |
| 2 | David Bowie | UK |

Ejemplo: vista tablas

Canción

| id | título | genero | duración | fecha | autor | álbum |
|----|----------------------------|--------|----------|------------|-------|-------|
| 0 | Norwegian Wood | Pop | 125 | 1965-03-12 | 0 | 0 |
| 1 | Here, there and everywhere | Pop | 145 | 1969-08-05 | 0 | 1 |
| 2 | Jumping jack flash | Pop | 225 | 1968-04-20 | 1 | 2 |

Usuario

| nick | nombre | email |
|-------|------------|-------------------|
| lola | Dolores | lola@gmail.com |
| pepe | José | jose@gmail.com |
| chema | José María | chema@gmail.com |
| charo | Rosario | rosario@gmail.com |

Contacto

| usuario1 | usuario2 |
|----------|----------|
| pepe | lola |
| charo | pepe |
| chema | charo |
| pepe | chema |

Escucha

| usuario | canción | instante |
|---------|---------|---------------------|
| charo | 1 | 2011-09-09 16:57:54 |
| pepe | 2 | 2011-09-12 21:15:30 |

Álbum

| id | autor | nombre | fecha |
|----|-------|-----------------|-------|
| 0 | 0 | Rubber Soul | 1965 |
| 1 | 0 | Revolver | 1966 |
| 2 | 2 | Beggars Banquet | 1968 |

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas en la BD

Canciones de artistas del Reino Unido

Título de las canciones escuchadas por un usuario

Todos los contactos de un usuario dado el nombre de éste

Usuarios que se llaman igual

Usuarios a distancia 2 de un usuario dado en la red social

Usuarios sin contactos

Contactos comunes a dos usuarios

Cuántas veces ha sido escuchada una canción

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Join

SELECT *campos*

FROM *tabla1* **JOIN** *tabla2* **ON** *condición*

[**WHERE** *condición*];

Ejemplos: Título de las canciones escuchadas por un usuario

Semánticamente
equivalentes

SELECT

c.titulo

FROM

cancion c, escucha e , usuario u

WHERE c.id=e.cancion **AND** e.usuario=u.nik **AND** u.nombre='Jose'

Producto cartesiano

SELECT

c.titulo

FROM

escucha e **JOIN** usuario u **ON** e.usuario=u.Nick

JOIN cancion c **ON** e.cancion=c.id

WHERE

u.nombre='Jose'

join

Join

Ejemplo: Todos los contactos de un usuario dado su nombre

Semánticamente
equivalentes

SELECT

u2.nombre

FROM

contacto c1 **JOIN** usuario u1 **ON** c1.usuario1=u1.Nick

JOIN usuario u2 **ON** c1.usuario2=u2.Nick

WHERE

u1.nombre='Jose'

join

SELECT

u.nombre

FROM

usuario u

WHERE u.nick **IN**

(**SELECT** contacto.usuario2 **FROM** usuario **JOIN** contacto **ON**
contacto.usuario1 = usuario.nick **AND** usuario.nombre = 'Jose');

SELECT anidado

Tipos de join

- ♦ **INNER.** Una fila de salida por cada par de filas de entrada que cumplan la condición del join. (Por defecto no hace falta poner INNER).
- ♦ **LEFT.** Igual que el inner excepto que si hay cualquier fila de la tabla de la izquierda para la que no se encuentra una coincidencia con una fila de la tabla de la derecha, las columnas de la tabla de la derecha se sustituyen por NULL. Al menos cada fila de la tabla de la izquierda aparece una vez.
- ♦ **RIGHT.** Lo mismo que el “left join” pero con las tablas invertidas.
- ♦ **FULL.** Combinación de “left” y “right join”. Todas las filas de ambas tablas aparecen al menos una vez.
- ♦ **NATURAL.** La condición consiste en igualdad entre los campos que se llamen igual.

Ejemplo: Join

estudiante

| <u>id</u> | nombre | taquilla |
|-----------|--------|----------|
| 0 | Carlos | 0 |
| 1 | Sara | 0 |
| 2 | Laura | 1 |
| 3 | Pedro | NULL |
| 4 | Jose | NULL |

taquilla

| <u>id</u> | estudiante | loc |
|-----------|------------|-------|
| 0 | 0 | Loc A |
| 1 | 2 | Loc B |
| 2 | NULL | Loc C |

- Hay estudiantes sin taquilla y algunos comparten taquilla
- Hay taquillas sin estudiantes
- estudiante es la tabla LEFT y taquilla la tabla RIGHT

Producto cartesiano

-- Producto cartesiano

SELECT * FROM estudiante, taquilla;

Estudiante, taquilla

| <u>id</u> | nombre | taquilla | id | estudiante | loc |
|-----------|--------|----------|----|------------|-------|
| 0 | Carlos | 0 | 0 | 0 | Loc A |
| 0 | Carlos | 0 | 1 | 2 | Loc B |
| 0 | Carlos | 0 | 2 | NULL | Loc C |
| 1 | Sara | 0 | 0 | 0 | Loc A |
| 1 | Sara | 0 | 1 | 2 | Loc B |
| 1 | Sara | 0 | 2 | NULL | Loc C |
| 2 | Laura | 1 | 0 | 0 | Loc A |
| 2 | Laura | 1 | 1 | 2 | Loc B |
| 2 | Laura | 1 | 2 | NULL | Loc C |
| 3 | Pedro | NULL | 0 | 0 | Loc A |
| 3 | Pedro | NULL | 1 | 2 | Loc B |
| 3 | Pedro | NULL | 2 | NULL | Loc C |
| 4 | Jose | NULL | 0 | 0 | Loc A |
| 4 | Jose | NULL | 1 | 2 | Loc B |
| 4 | Jose | NULL | 2 | NULL | Loc C |

Ejemplo: Join (inner join)

SELECT

*

FROM

estudiante e **JOIN** taquilla t **ON** e.id=t.estudiante;

| <u>id</u> | nombre | taquilla | id | estudiante | loc |
|-----------|--------|----------|----|------------|-------|
| 0 | Carlos | 0 | 0 | 0 | Loc A |
| 2 | Laura | 1 | 1 | 2 | Loc B |

sólo donde id de estudiante coincide con estudiante en taquilla

Ejemplo: left join

SELECT

*

FROM

estudiante e **LEFT JOIN** taquilla t **ON** e.id=t.estudiante;

| <u>id</u> | nombre | taquilla | id | estudiante | loc |
|-----------|--------|----------|------|------------|-------|
| 0 | Carlos | 0 | 0 | 0 | Loc A |
| 1 | Sara | 0 | NULL | NULL | NULL |
| 2 | Laura | 1 | 1 | 2 | Loc B |
| 3 | Pedro | NULL | NULL | NULL | NULL |
| 4 | Jose | NULL | NULL | NULL | NULL |

Inner join + todas las filas de la tabla de la izquierda que no coincidieron donde se sustituye las columnas de la tabla de la derecha por NULL

Ejemplo: right join

SELECT

*

FROM

estudiante e **RIGHT JOIN** taquilla t **ON** e.id=t.estudiante;

| <u>id</u> | nombre | taquilla | id | estudiante | loc |
|-----------|--------|----------|----|------------|-------|
| 0 | Carlos | 0 | 0 | 0 | Loc A |
| 2 | Laura | 1 | 1 | 2 | Loc B |
| NULL | NULL | NULL | 2 | NULL | Loc C |

Inner join + todas las filas de la tabla de la derecha que no coincidieron donde se sustituye las columnas de la tabla de la izquierda por NULL

Ejemplo: full join

SELECT

*

FROM

estudiante e **FULL JOIN** taquilla t **ON** e.id=t.estudiante;

| <u>id</u> | nombre | taquilla | id | estudiante | loc |
|-----------|--------|----------|------|------------|-------|
| 0 | Carlos | 0 | 0 | 0 | Loc A |
| 1 | Sara | 0 | NULL | NULL | NULL |
| 2 | Laura | 1 | 1 | 2 | Loc B |
| 3 | Pedro | NULL | NULL | NULL | NULL |
| 4 | Jose | NULL | NULL | NULL | NULL |
| NULL | NULL | NULL | 2 | NULL | Loc C |

left join + right join

Ejemplo: natural join

SELECT

*

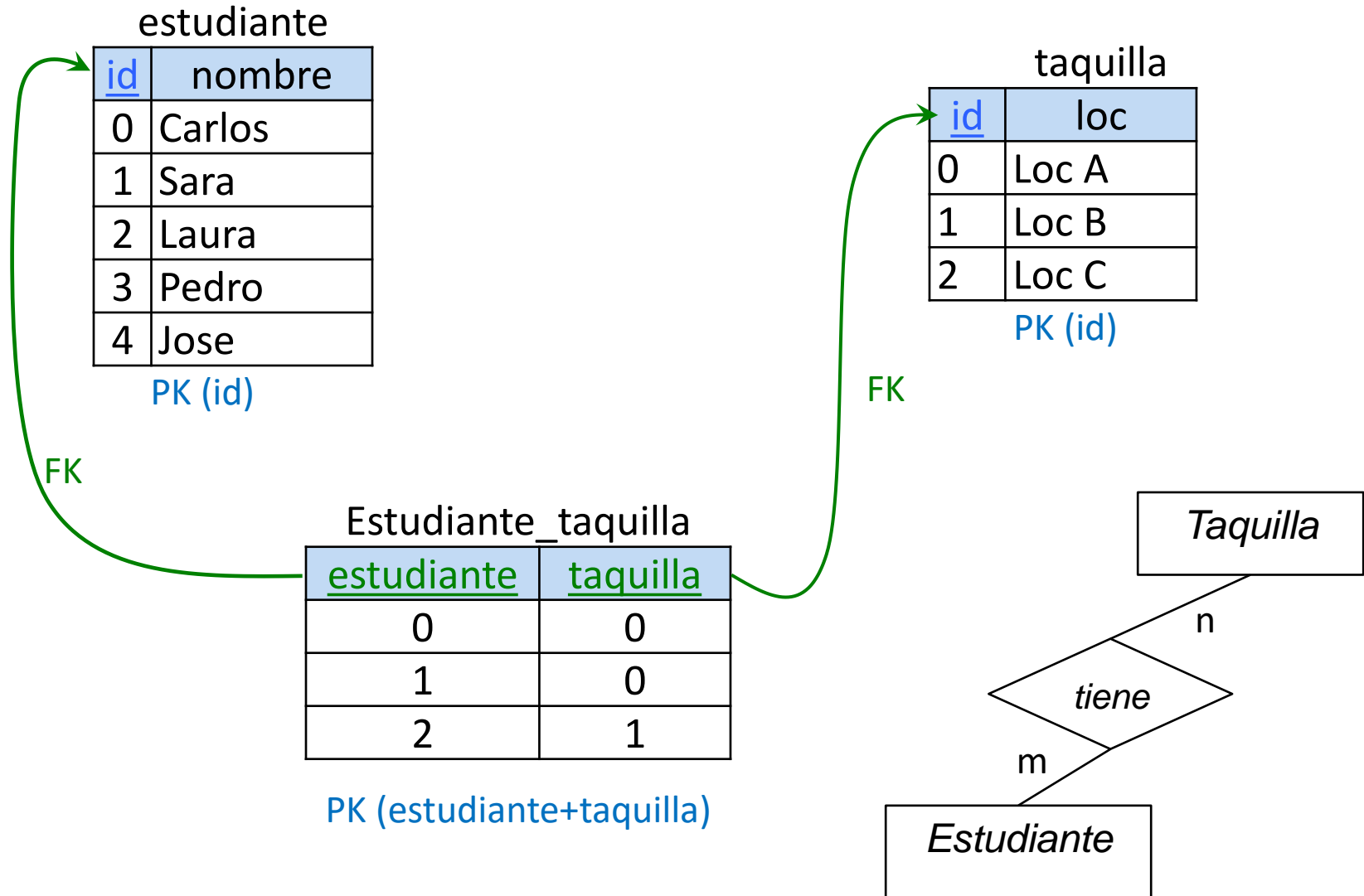
FROM

estudiante **NATURAL JOIN** taquilla;

| <u>id</u> | nombre | taquilla | estudiante | loc |
|-----------|--------|----------|------------|-------|
| 0 | Carlos | 0 | 0 | Loc A |
| 1 | Sara | 0 | 2 | Loc B |
| 2 | Laura | 1 | NULL | Loc C |

Resultados en donde las columnas con el mismo nombre coinciden (id)

Ejemplo: Join (diseño correcto)



Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas en la BD

Canciones de artistas del Reino Unido

Título de las canciones escuchadas por un usuario

Todos los contactos de un usuario dado el nombre de éste

Usuarios que se llaman igual

Usuarios a distancia 2 de un usuario dado en la red social

Usuarios sin contactos

Contactos comunes a dos usuarios

Cuántas veces ha sido escuchada una canción

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Artista

| id | nombre | nacionalidad |
|----|--------------------|--------------|
| 0 | The Beatles | UK |
| 1 | The Rolling Stones | UK |
| 2 | David Bowie | UK |

Ejemplo: vista tablas

Canción

| id | título | genero | duración | fecha | autor | álbum |
|----|----------------------------|--------|----------|------------|-------|-------|
| 0 | Norwegian Wood | Pop | 125 | 1965-03-12 | 0 | 0 |
| 1 | Here, there and everywhere | Pop | 145 | 1969-08-05 | 0 | 1 |
| 2 | Jumping jack flash | Pop | 225 | 1968-04-20 | 1 | 2 |

Usuario

| nick | nombre | email |
|-------|------------|-------------------|
| lola | Dolores | lola@gmail.com |
| pepe | José | jose@gmail.com |
| chema | José María | chema@gmail.com |
| charo | Rosario | rosario@gmail.com |

Contacto

| usuario1 | usuario2 |
|----------|----------|
| pepe | lola |
| charo | pepe |
| chema | charo |
| pepe | chema |

Escucha

| usuario | canción | instante |
|---------|---------|---------------------|
| charo | 1 | 2011-09-09 16:57:54 |
| pepe | 2 | 2011-09-12 21:15:30 |

Álbum

| id | autor | nombre | fecha |
|----|-------|-----------------|-------|
| 0 | 0 | Rubber Soul | 1965 |
| 1 | 0 | Revolver | 1966 |
| 2 | 2 | Beggars Banquet | 1968 |

Alias

SELECT *campos* FROM *tabla* **AS** *alias* [(*alias-campo1*, *alias-campo2*, ...)]
[WHERE *condición*];

SELECT *campo* **AS** *alias* FROM ...

Ejemplo: Usuarios que se llaman igual

SELECT

u1.nick **AS** NickName, u1.nombre **AS** name

FROM

usuario u1, usuario u2

WHERE

u1.nombre=u2.nombre **AND** u1.nick<>u2.nick

Ejemplo: Notas

SELECT n.dni, n.teoria * 0.6 + n.practicas * 0.4 **AS** media **FROM** Notas n;

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los en la BD

Canciones de artistas del Reino Unido

Título de las canciones escuchadas por un usuario

Todos los contactos de un usuario dado el nombre de éste

Usuarios que se llaman igual

Usuarios a distancia 2 de un usuario dado en la red social

Usuarios sin contactos

Contactos comunes a dos usuarios

Cuántas veces ha sido escuchada una canción

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Consultas anidadas

SELECT *campos* FROM (SELECT ...) AS *alias* WHERE ...;

SELECT *campos* FROM *tabla*
WHERE (*campo1*, *campo2*, ...) **IN** (SELECT *campo1*, *campo2*, ...);

SELECT *campos* FROM *tabla*
WHERE *campo comparación* (**SOME** | **ALL**) (SELECT ...);

SELECT *campos* FROM *tabla*
WHERE [NOT] **EXISTS** (SELECT ...);

SELECT *campos* FROM *tabla*
WHERE (SELECT ...) **CONTAINS** (SOME | ALL) (SELECT ...);

Artista

| id | nombre | nacionalidad |
|----|--------------------|--------------|
| 0 | The Beatles | UK |
| 1 | The Rolling Stones | UK |
| 2 | David Bowie | UK |

Ejemplo: vista tablas

Canción

| id | título | genero | duración | fecha | autor | álbum |
|----|----------------------------|--------|----------|------------|-------|-------|
| 0 | Norwegian Wood | Pop | 125 | 1965-03-12 | 0 | 0 |
| 1 | Here, there and everywhere | Pop | 145 | 1969-08-05 | 0 | 1 |
| 2 | Jumping jack flash | Pop | 225 | 1968-04-20 | 1 | 2 |

Usuario

| nick | nombre | email |
|-------|------------|-------------------|
| lola | Dolores | lola@gmail.com |
| pepe | José | jose@gmail.com |
| chema | José María | chema@gmail.com |
| charo | Rosario | rosario@gmail.com |

Contacto

| usuario1 | usuario2 |
|----------|----------|
| pepe | lola |
| charo | pepe |
| chema | charo |
| pepe | chema |

Escucha

| usuario | canción | instante |
|---------|---------|---------------------|
| charo | 1 | 2011-09-09 16:57:54 |
| pepe | 2 | 2011-09-12 21:15:30 |

Álbum

| id | autor | nombre | fecha |
|----|-------|-----------------|-------|
| 0 | 0 | Rubber Soul | 1965 |
| 1 | 0 | Revolver | 1966 |
| 2 | 2 | Beggars Banquet | 1968 |

Consultas anidadas (SELECT * FROM (SELECT))

1. join de “contacto” para ver segundo nivel

```
SELECT c1.usuario1 u11, c1.usuario2 u12, c2.usuario1 u21, c2.usuario2 u22  
FROM contacto c1 JOIN contacto c2 ON c1.usuario2=c2.usuario1
```

| u11 | u12 | u21 | u22 |
|-------|-------|-------|-------|
| charo | pepe | pepe | chema |
| charo | pepe | pepe | lola |
| chema | charo | charo | pepe |
| pepe | chema | chema | charo |

2. selección ultima columna cuando la 1ª es charo

```
SELECT u.u22 FROM (  
    SELECT c1.usuario1 u11, c1.usuario2 u12, c2.usuario1 u21, c2.usuario2 u22  
    FROM contacto c1 JOIN contacto c2 ON c1.usuario2=c2.usuario1  
) AS u WHERE u.u11='charo'
```

| u22 |
|-------|
| lola |
| chema |

Consultas anidadas (SELECT * FROM .. WHERE in (SELECT))

[Previous Slide](#)

Artista

| id | nombre | nacionalidad |
|----|--------------------|--------------|
| 0 | The Beatles | UK |
| 1 | The Rolling Stones | UK |
| 2 | David Bowie | UK |

Ejemplo: vista tablas

Canción

| id | título | genero | duración | fecha | autor | álbum |
|----|----------------------------|--------|----------|------------|-------|-------|
| 0 | Norwegian Wood | Pop | 125 | 1965-03-12 | 0 | 0 |
| 1 | Here, there and everywhere | Pop | 145 | 1969-08-05 | 0 | 1 |
| 2 | Jumping jack flash | Pop | 225 | 1968-04-20 | 1 | 2 |

Usuario

| nick | nombre | email |
|-------|------------|-------------------|
| lola | Dolores | lola@gmail.com |
| pepe | José | jose@gmail.com |
| chema | José María | chema@gmail.com |
| charo | Rosario | rosario@gmail.com |

Contacto

| usuario1 | usuario2 |
|----------|----------|
| pepe | lola |
| charo | pepe |
| chema | charo |
| pepe | chema |

Escucha

| usuario | canción | instante |
|---------|---------|---------------------|
| charo | 1 | 2011-09-09 16:57:54 |
| pepe | 2 | 2011-09-12 21:15:30 |

Álbum

| id | autor | nombre | fecha |
|----|-------|-----------------|-------|
| 0 | 0 | Rubber Soul | 1965 |
| 1 | 0 | Revolver | 1966 |
| 2 | 2 | Beggars Banquet | 1968 |

Consultas anidadas (cont)

Nombre de los contactos de 'pepe'

```
SELECT * FROM usuario u WHERE u.nick = ANY (SELECT c.usuario2  
FROM contacto c WHERE c.usuario1='pepe')
```

= ANY es equivalente a IN o = SOME. Devuelve TRUE si algún valor de la izquierda del operador (u.nick) coincide con alguno de los de la derecha (SELECT).

= ALL devuelve TRUE si todos coinciden

Además del operador = se pueden usar >, >=, <, <=, <>

```
SELECT * FROM usuario u WHERE EXISTS (SELECT * FROM contacto c  
WHERE u.nick=c.usuario2 AND c.usuario1='pepe')
```

Si el SELECT anidado existe se ejecuta el primer SELECT

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas cubiertos en la BD

Canciones de artistas del Reino Unido

Título de las canciones escuchadas por un usuario

Todos los contactos de un usuario dado el nombre de éste

Usuarios que se llaman igual

Usuarios a distancia 2 de un usuario dado en la red social

Usuarios sin contactos

Contactos comunes a dos usuarios

Cuántas veces ha sido escuchada una canción

Artistas por orden de más a menos escuchados

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Artista

| id | nombre | nacionalidad |
|----|--------------------|--------------|
| 0 | The Beatles | UK |
| 1 | The Rolling Stones | UK |
| 2 | David Bowie | UK |

Ejemplo: vista tablas

Canción

| id | título | genero | duración | fecha | autor | álbum |
|----|----------------------------|--------|----------|------------|-------|-------|
| 0 | Norwegian Wood | Pop | 125 | 1965-03-12 | 0 | 0 |
| 1 | Here, there and everywhere | Pop | 145 | 1969-08-05 | 0 | 1 |
| 2 | Jumping jack flash | Pop | 225 | 1968-04-20 | 1 | 2 |

Usuario

| nick | nombre | email |
|-------|------------|-------------------|
| lola | Dolores | lola@gmail.com |
| pepe | José | jose@gmail.com |
| chema | José María | chema@gmail.com |
| charo | Rosario | rosario@gmail.com |

Contacto

| usuario1 | usuario2 |
|----------|----------|
| pepe | lola |
| charo | pepe |
| chema | charo |
| pepe | chema |

Escucha

| usuario | canción | instante |
|---------|---------|---------------------|
| charo | 1 | 2011-09-09 16:57:54 |
| pepe | 2 | 2011-09-12 21:15:30 |

Álbum

| id | autor | nombre | fecha |
|----|-------|-----------------|-------|
| 0 | 0 | Rubber Soul | 1965 |
| 1 | 0 | Revolver | 1966 |
| 2 | 2 | Beggars Banquet | 1968 |

Álgebra de conjuntos

consulta1 UNION *consulta2*

consulta1 INTERSECT *consulta2*

consulta1 EXCEPT *consulta2*

- ♦ **Tuplas homogéneas**: los conjuntos de tuplas tienen que tener los mismos campos
- ♦ Aplica un **DISTINCT implícito** (a menos que indiquemos ALL)
- ♦ Normalmente se pueden reexpresar como una consulta simple con AND, OR, NOT en la condición WHERE → la ventaja es de legibilidad

Ejemplo: contactos comunes a 'charo' o 'lola'

```
((SELECT c.usuario2 FROM contacto c WHERE c.usuario1 = 'charo'
UNION
SELECT c.usuario1 FROM contacto c WHERE c.usuario2 = 'charo'))
INTERSECT
((SELECT c.usuario2 FROM contacto c WHERE c.usuario1 = 'lola'
UNION
SELECT c.usuario1 FROM contacto c WHERE c.usuario2 = 'lola'))
```

Álgebra de conjuntos (cont)

Ejemplo: Contactos a distancia 2 de 'charo' (en ambos sentidos)

(**SELECT** c1.usuario1 **FROM** contacto c1 **JOIN** contacto c2 **ON**
c1.usuario2=c2.usuario1 **WHERE** c2.usuario2='charo')

UNION

(**SELECT** c2.usuario2 **FROM** contacto c1 **JOIN** contacto c2 **ON**
c1.usuario2=c2.usuario1 **WHERE** c1.usuario1='charo')

chema

lola

pepe

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas cubiertos en la BD

Canciones de artistas del Reino Unido

Título de las canciones escuchadas por un usuario

Todos los contactos de un usuario dado el nombre de éste

Usuarios que se llaman igual

Usuarios a distancia 2 de un usuario dado en la red social

Usuarios sin contactos

Contactos comunes a dos usuarios

Cuántas veces ha sido escuchada una canción

Cuántas veces ha sido escuchado cada artista

Usuarios con más de dos contactos

Usuarios ordenados por nº de contactos

Usuario con más contactos

Orden, agregación: ejemplos

Cuántas veces ha sido escuchada una canción

```
SELECT c.titulo FROM escucha e JOIN cancion c ON e.cancion = c.id
```

```
SELECT COUNT(*) FROM escucha e JOIN cancion c ON e.cancion = c.id  
WHERE c.titulo='Jumping jack flash'
```

Cuántas veces ha sido escuchado un artista determinado

```
SELECT  
    COUNT(*)  
FROM  
    escucha e JOIN cancion c ON e.cancion = c.id  
    JOIN artista a ON c.autor=a.id  
WHERE g.nombre='The Beatles'
```

Motivación

Base de datos de músicos, canciones, usuarios, escuchas, red social

Ejemplos de consultas

Canciones de los años 60 (mostrar título y género)

Conjunto de nacionalidades de los artistas cubiertos en la BD

Canciones de artistas del Reino Unido

Título de las canciones escuchadas por un usuario

Todos los contactos de un usuario dado el nombre de éste

Usuarios que se llaman igual

Usuarios a distancia 2 de un usuario dado en la red social

Usuarios sin contactos

Contactos comunes a dos usuarios

Cuántas veces ha sido escuchada una canción

Cuántas veces ha sido escuchado cada artista

Usuarios con más de dos contactos

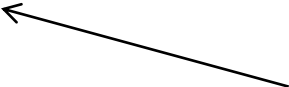
Usuarios ordenados por nº de contactos

Usuario con más contactos

Orden, agregación

SELECT COUNT ([DISTINCT] *campo*) FROM *tabla* ...
[**GROUP BY** *campo1*, *campo2*, ... [**HAVING** *condición*]];


Filtro post-
agregación



SELECT SUM | MAX | MIN | AVG (*campo*) FROM *tabla* ...
[**GROUP BY** *campo1*, *campo2*, ...];

SELECT ...

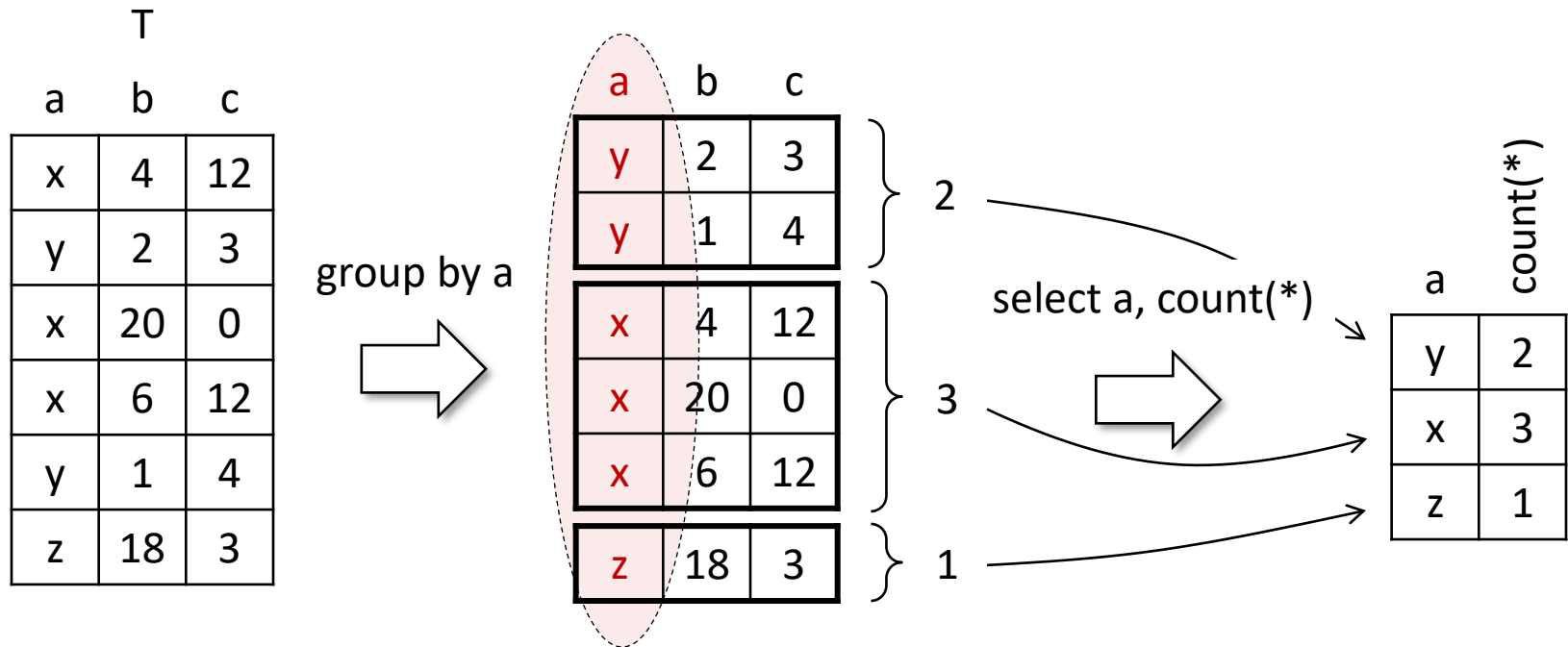
[**ORDER BY** *campo1*, *campo2*, ... [DESC]];



Útil combinar
con LIMIT *n*

Orden, agregación: ejemplo

SELECT a, count(*) FROM T GROUP BY a

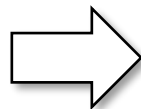


Orden, agregación: ejemplo

SELECT a, count(*), **sum(b)** FROM T GROUP BY a

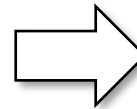
| T | | |
|---|----|----|
| a | b | c |
| x | 4 | 12 |
| y | 2 | 3 |
| x | 20 | 0 |
| x | 6 | 12 |
| y | 1 | 4 |
| z | 18 | 3 |

group by a



| a | b | c |
|---|----|----|
| y | 2 | 3 |
| y | 1 | 4 |
| x | 4 | 12 |
| x | 20 | 0 |
| x | 6 | 12 |
| z | 18 | 3 |

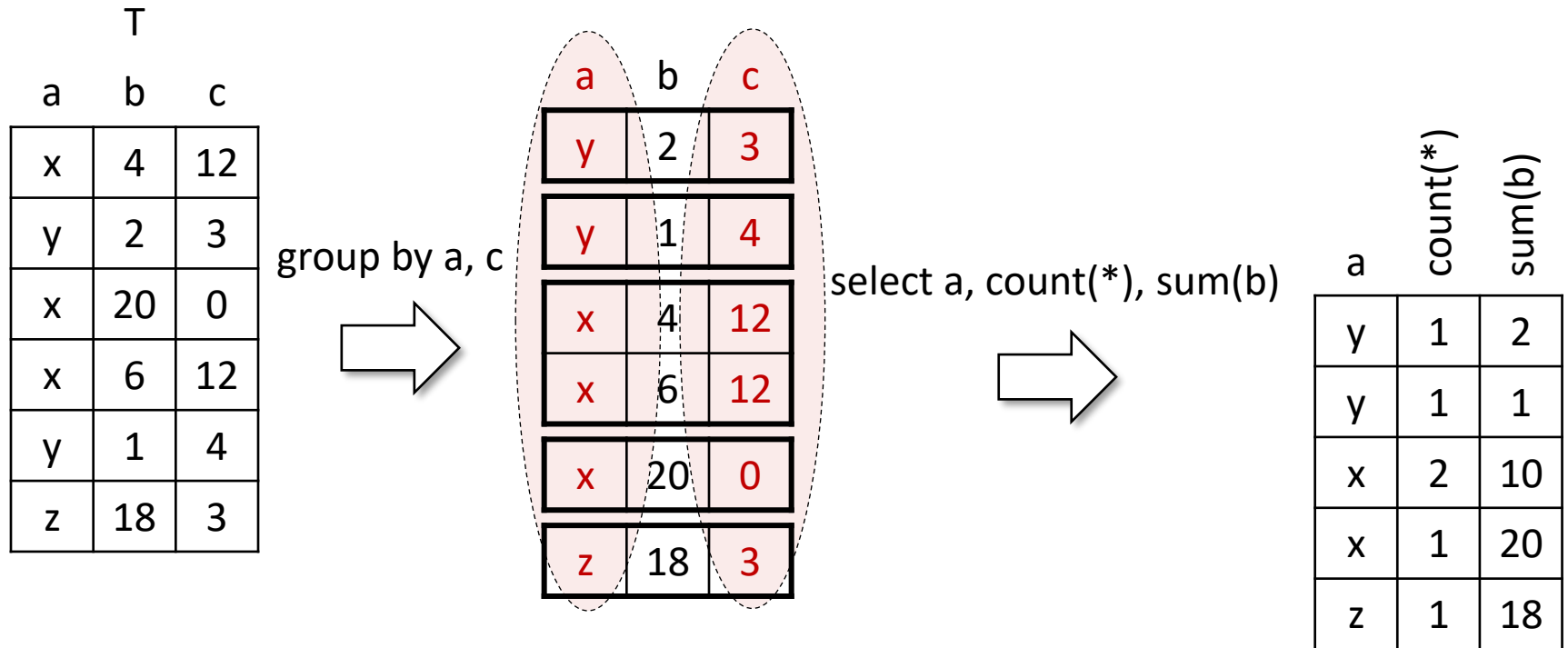
select a, count(*), sum(b)



| a | count(*) | sum(b) |
|---|----------|--------|
| y | 2 | 3 |
| x | 3 | 30 |
| z | 1 | 18 |

Orden, agregación: ejemplo

SELECT a, count(*), sum(b) FROM T GROUP BY a, c



Orden, agregación: ejemplo

SELECT a, count(*), sum(b) FROM T GROUP BY a, c
HAVING sum(b) < 15

| T | | |
|---|----|----|
| a | b | c |
| x | 4 | 12 |
| y | 2 | 3 |
| x | 20 | 0 |
| x | 6 | 12 |
| y | 1 | 4 |
| z | 18 | 3 |

group by a, c

| a | b | c |
|---|----|----|
| y | 2 | 3 |
| y | 1 | 4 |
| x | 4 | 12 |
| x | 6 | 12 |
| x | 20 | 0 |
| z | 18 | 3 |

select a, count(*), sum(b)

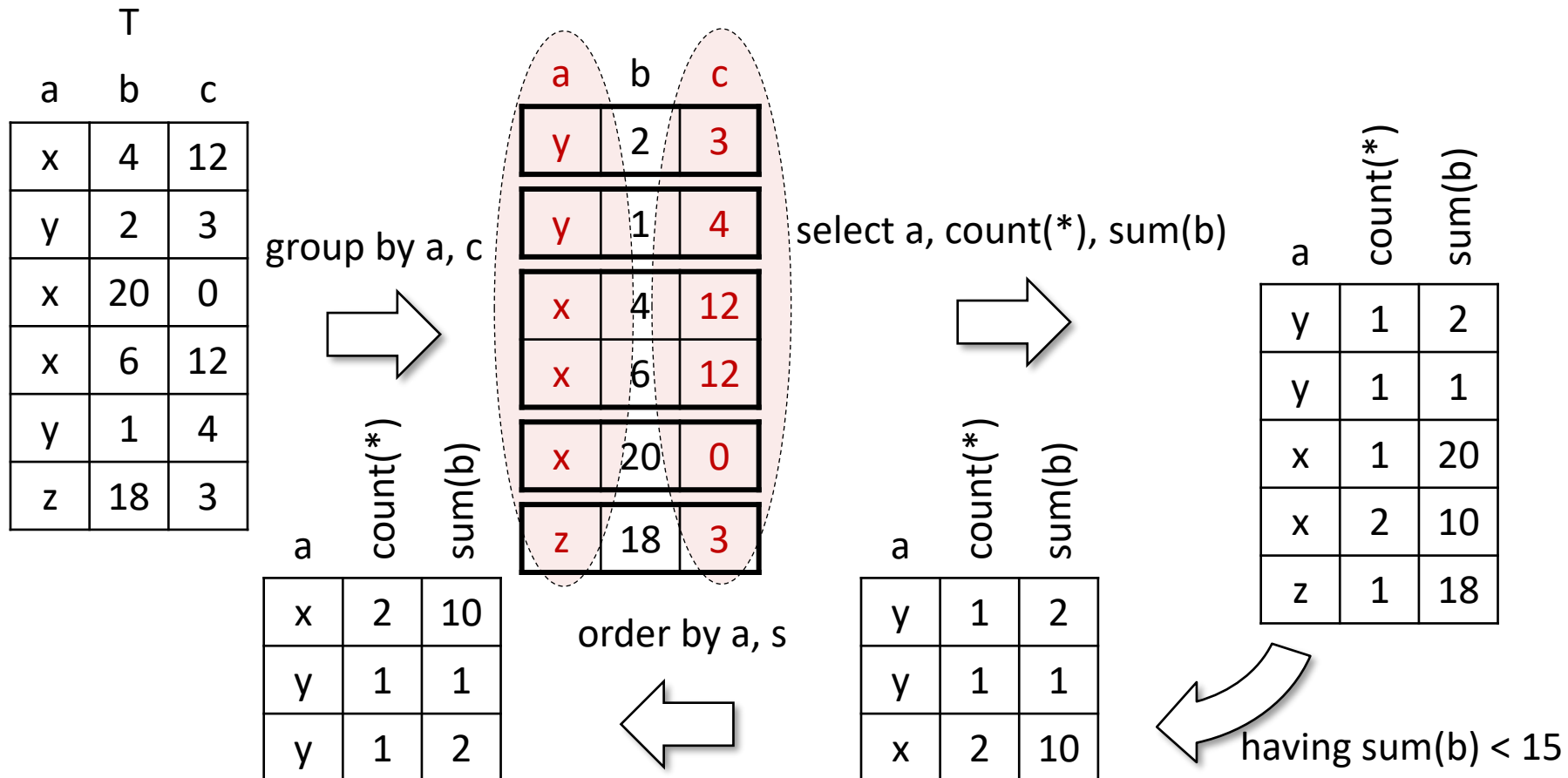
| a | count(*) | sum(b) |
|---|----------|--------|
| y | 1 | 2 |
| y | 1 | 1 |
| x | 2 | 10 |

| a | count(*) | sum(b) |
|---|----------|--------|
| y | 1 | 2 |
| y | 1 | 1 |
| x | 2 | 10 |
| x | 1 | 20 |
| z | 1 | 18 |

having sum(b) < 15

Orden, agregación: ejemplo

SELECT a, count(*), sum(b) s FROM T GROUP BY a, c
HAVING sum(b) < 15
ORDER BY a, s



Orden de ejecución de cláusulas en una query

1. Se construyen el producto de todas las tablas en la cláusula FROM
2. Se evalúa la cláusula WHERE para determinar las filas que no satisfacen las condiciones de búsqueda.
3. Se agrupan las columnas de la cláusula GROUP BY
4. Se eliminan los grupos que no satisfacen las condiciones de la cláusula HAVING
5. Se evalúa las expresiones en SELECT
6. Si existe DISTINCT en el SELECT se eliminan los duplicados
7. Las filas se ordenan conforme a la cláusula ORDER BY

Por esta razón, los alias en las columnas no funcionan en la cláusula HAVING pero si en ORDER BY

Artista

| id | nombre | nacionalidad |
|----|--------------------|--------------|
| 0 | The Beatles | UK |
| 1 | The Rolling Stones | UK |
| 2 | David Bowie | UK |

Ejemplo: vista tablas

Canción

| id | título | genero | duración | fecha | autor | álbum |
|----|----------------------------|--------|----------|------------|-------|-------|
| 0 | Norwegian Wood | Pop | 125 | 1965-03-12 | 0 | 0 |
| 1 | Here, there and everywhere | Pop | 145 | 1969-08-05 | 0 | 1 |
| 2 | Jumping jack flash | Pop | 225 | 1968-04-20 | 1 | 2 |

Usuario

| nick | nombre | email |
|-------|------------|-------------------|
| lola | Dolores | lola@gmail.com |
| pepe | José | jose@gmail.com |
| chema | José María | chema@gmail.com |
| charo | Rosario | rosario@gmail.com |

Contacto

| usuario1 | usuario2 |
|----------|----------|
| pepe | lola |
| charo | pepe |
| chema | charo |
| pepe | chema |

Escucha

| usuario | canción | instante |
|---------|---------|---------------------|
| charo | 1 | 2011-09-09 16:57:54 |
| pepe | 2 | 2011-09-12 21:15:30 |

Álbum

| id | autor | nombre | fecha |
|----|-------|-----------------|-------|
| 0 | 0 | Rubber Soul | 1965 |
| 1 | 0 | Revolver | 1966 |
| 2 | 2 | Beggars Banquet | 1968 |

Orden, agregación: ejemplos

Usuarios con mas de un contacto

SELECT

c.usuario1, **count(*)**

FROM contacto c **GROUP BY** c.usuario1 **HAVING COUNT(*) > 1**

Usuarios ordenados por numero de contactos

SELECT

c.usuario1, count(*)

FROM contacto c **GROUP BY** c.usuario1 **ORDER BY COUNT(*) DESC**

Usuario con mas contactos

SELECT

c.usuario1, count(*)

FROM contacto c **GROUP BY** c.usuario1 **ORDER BY COUNT(*) DESC LIMIT 1**

Vistas

CREATE VIEW *nombre* **AS** SELECT...;

Dan un nombre a una consulta

Equivalente a consulta anidada, pero...

- Útil para reutilizar consultas y simplificar la sintaxis
- Se mantienen siempre actualizadas
- Pueden configurarse para que se almacenen en disco

Ejemplos:

CREATE VIEW

vista **AS**

SELECT

a1.nombre album, a2.nombre grupo, a1.fecha

FROM album a1 **JOIN** artista a2 **ON** a1.autor=a2.id;

SELECT * FROM vista

Otros elementos de SQL...

- ◆ Esquemas
 - Para definir espacios de nombres de tablas y permisos de usuarios
- ◆ Dominios
 - Tipos de datos definidos por propiedades y condiciones sobre un tipo primitivo (p.e. una cadena de texto con un cierto formato en expresión regular)
- ◆ Triggers
 - Ejecutar un procedimiento cuando se producen acciones de actualización (insert, update, delete) de una tabla
- ◆ Assertion
 - Checks que pueden hacerse sobre varias filas y varias tablas
- ◆ Transacciones
 - Expresan secuencias de acciones y consultas que deben completarse o cancelarse en bloque
 - Permiten también sincronizar (bloquear) operaciones concurrentes
- ◆ Y muchas más funcionalidades básicas soportadas por cada SGBD extendiendo el estándar SQL...