MANIPULACIÓN DE DATOS

Ricardo Pérez López

IES Doñana, curso 2017-18

ESQUEMAS

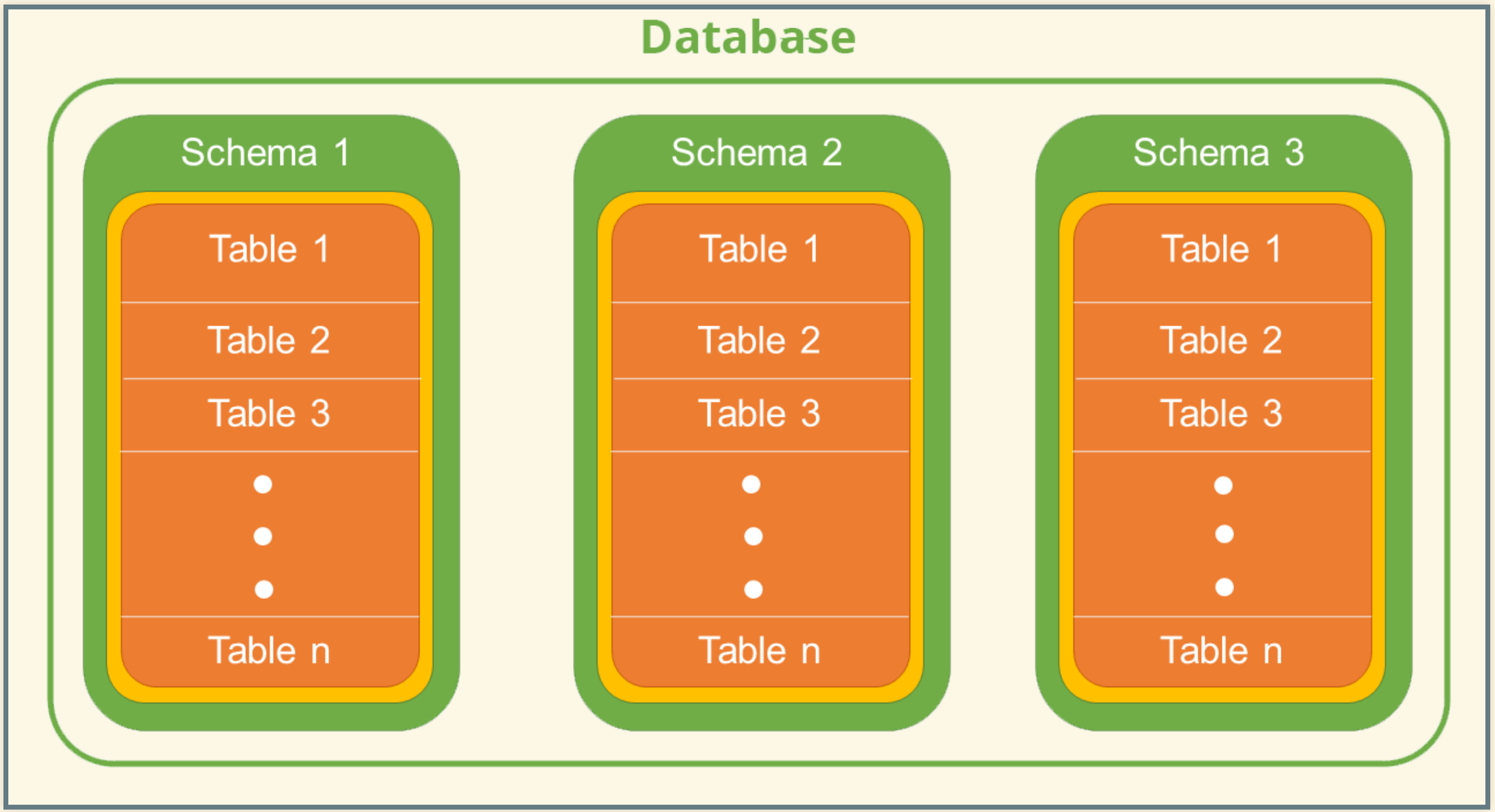
ESQUEMAS

- Una base de datos contiene uno o más **esquemas**, los cuales a su vez contienen tablas y otros objetos.
- Dos objetos pueden tener el mismo nombre siempre que se encuentren en esquemas diferentes.
- Puede haber dos tablas distintas que se llamen igual (por ejemplo, `mitabla`) siempre y cuando se encuentren almacenadas en esquemas diferentes.

Es lo mismo que ocurre en un sistema de archivos:

- Las tablas (y demás objetos de bases de datos) son como los archivos.
- Los esquemas son como las carpetas, ya que son los que contienen los objetos de bases de datos de la misma forma que una carpeta contiene archivos.

La diferencia es que **los esquemas no pueden anidarse** (no puede haber esquemas dentro de esquemas).


Database
Schema 1
Table 1
Table 2
Table 3
Table n
Schema 2
Table 1
Table 2
Table 3
Table n
Schema 3
Table 1
Table 2
Table 3
Table n

Un usuario puede acceder a los objetos de un determinado esquema si dispone de los privilegios suficientes.

USO DE LOS ESQUEMAS

- Para acceder a un objeto (una tabla, por ejemplo) situado dentro de un esquema, se escribe su **nombre cualificado**, que consta del nombre del esquema y el nombre del objeto separados por un punto:

esquema.tabla

- Esto funciona en cualquier sitio donde se pueda escribir el nombre de una tabla (quien dice tabla, dice cualquier otro objeto de base de datos). Por ejemplo:

```
SELECT *
  FROM facturacion.pedidos;
```

LA RUTA DE BÚSQUEDA DE ESQUEMAS

- Hasta ahora hemos estado utilizando tablas sin especificar el nombre de ningún esquema.
- Cuando usamos el nombre de una tabla sin indicar el esquema, se dice que estamos usando su **nombre no cualificado**.
- El sistema determina a qué tabla nos estamos refiriendo siguiendo una *ruta de búsqueda*, que no es más que una lista de esquemas en los que mirar.
- Se va mirando en dicha ruta esquema por esquema hasta encontrar una tabla con el nombre que se está buscando.
- Si no se encuentra ninguna en toda la ruta se emitirá un mensaje de error, incluso si hay tablas con ese nombre en otros esquemas de la base de datos.

- El primer esquema que aparece en la ruta de búsqueda se denomina el **esquema actual**.
- Además de ser el primer esquema en el que se busca, también es el esquema en el que se crearán las nuevas tablas si no se especifica ningún esquema durante su creación.

Para mostrar la ruta de búsqueda actual, se usa el comando:

```
SHOW search_path;
```

Por defecto vale:

```
   search_path
-----------------
 "$user", public
```

El primer elemento representa el esquema que tenga el mismo nombre que el usuario actual. Si no existe tal esquema, se ignora.

El segundo elemento se refiere al **esquema público**, que trataremos en breve.

EJEMPLO

- Dada la situación predeterminada, en la que `search_path='"$user", public'`, y siendo `manolo` el usuario actual, ¿qué ocurre cuando `manolo` introduce la siguiente orden?

```
SELECT *
  FROM emple;
```

- *(SOLUCIÓN)* El sistema busca la tabla `emple` primero en el esquema `manolo`, y si no lo encuentra, lo busca en el esquema público (`public`). Si tampoco la encuentra ahí, emitirá un error.

EL ESQUEMA PÚBLICO

- Se representa como `public`.
- Es el esquema en el que están actualmente todas las tablas de nuestra base de datos.
- Se puede decir que es el esquema "*por defecto*", mientras no se usen otros esquemas.

FUNCIÓN `copiar_tabla()`

- Un usuario sólo puede modificar los datos de una tabla si:
  - Es su propietario, o
  - Su propietario le ha concedido los privilegios adecuados.
- Como no tenéis privilegios para poder modificar las tablas, tenéis que crear vuestras propias tablas en vuestro propio esquema.
- A partir de ahora, cada uno de vosotros cuenta con un esquema que se llama igual que vuestro nombre de usuario, y cada usuario sólo puede entrar en su esquema (no en el de los demás).
- Lo que haréis será crear *copias* de las tablas del esquema `public` en vuestros propios esquemas, usando la función **`copiar_tabla()`**.

EJEMPLO

```sql
SELECT copiar_tabla('public.emple', 'ricardo.emple');
```

- Si en algún momento hay que reconstruir la copia, primero se debe **borrar** con la orden:

```
DROP TABLE ricardo.emple CASCADE;
```

- Evidentemente, sólo podemos borrar nuestra copia, no la que hay en `public`.

BORRADO DE FILAS

DELETE FROM

```
DELETE FROM tabla
    WHERE condición;
```

EJEMPLO

```
DELETE FROM ricardo.emple
    WHERE dept_no = 30;
```

Pregunta: ¿Da lo mismo poner `emple` que `ricardo.emple`?

El WHERE puede ser muy complicado:

```
DELETE FROM emple
    WHERE salario > (SELECT avg(salario)
                       FROM emple
                      WHERE dept_no = 20);
```

Incluso con correlacionadas:

```
DELETE FROM emple e
    WHERE salario > (SELECT salario
                       FROM emple j
                      WHERE e.dir = j.emp_no);
```

Un `DELETE` sin `WHERE`:

```sql
DELETE FROM tabla;
```

Es lo mismo que poner:

```sql
DELETE FROM tabla
    WHERE true;
```

¡Borra todas las filas de la tabla!

```
<iframe width="620" height="315"
    src="https://www.youtube.com/embed/OhlVBpEnjig">
</iframe>
```

INSERCIÓN DE FILAS

INSERT INTO

```
INSERT INTO tabla [ ( columna [, ...] ) ]
    { VALUES ( { expresión | DEFAULT } [, ...] ) [, ...]
    | DEFAULT VALUES
    | consulta }
```

EJEMPLOS

Con todas las columnas:

```
INSERT INTO depart (dept_no, dnombre, loc)
    VALUES (50, 'INFORMÁTICA', 'SANLÚCAR');
```

Con algunas columnas:

```
INSERT INTO depart (dept_no, dnombre)
    VALUES (60, 'PREVENCIÓN');
```

Sin columnas:

```
INSERT INTO depart
    VALUES (70, 'LABORATORIO', 'JEREZ');
```

DEFAULT

```
INSERT INTO depart (dept_no, dnombre)
    VALUES (60, 'PREVENCIÓN');
```

Es lo mismo que:

```
INSERT INTO depart (dept_no, dnombre, loc)
    VALUES (60, 'PREVENCIÓN', DEFAULT);
```

DEFAULT / NULL

¿Es lo mismo `DEFAULT` que `NULL`?

```
INSERT INTO depart (dept_no, dnombre, loc)
    VALUES (60, 'PREVENCIÓN', DEFAULT);
```

```
INSERT INTO depart (dept_no, dnombre, loc)
    VALUES (60, 'PREVENCIÓN', NULL);
```

VARIAS FILAS

```
INSERT INTO depart (dept_no, dnombre, loc)
    VALUES (50, 'INFORMÁTICA', 'SANLÚCAR'),
           (60, 'LABORATORIO', 'JEREZ');
```

Con la coma delante de la línea siguiente:

```
INSERT INTO depart (dept_no, dnombre, loc)
    VALUES (50, 'INFORMÁTICA', 'SANLÚCAR')
         , (60, 'LABORATORIO', 'JEREZ');
```

DEFAULT VALUES

```
INSERT INTO depart
    DEFAULT VALUES;
```

Es lo mismo que:

```
INSERT INTO depart (dept_no, dnombre, loc)
    VALUES (DEFAULT, DEFAULT, DEFAULT);
```

INSERT ... SELECT

```
SELECT copiar_tabla('public.emple', 'emple30');

DELETE FROM emple30
    WHERE true;

INSERT INTO emple30
    SELECT *
      FROM emple
     WHERE dept_no = 30;
```

MODIFICACIÓN DE FILAS

UPDATE

```
UPDATE tabla
    SET { columna = { expresión | DEFAULT }
        | ( columna [, ...] ) = ( { expresión | DEFAULT } [, ...] )
        | ( columna [, ...] ) = ( subconsulta )
        } [, ...]
  [ FROM tabla ]
  [ WHERE condición ]
```

EJEMPLOS

```sql
UPDATE emple
   SET oficio = 'VENDEDOR'
 WHERE dept_no = 30;
```

```sql
UPDATE emple
   SET salario = salario * 2.0
 WHERE salario > 1000;
```

¿Importa el orden de las asignaciones?

```
UPDATE emple
   SET salario = salario + comision
     , comision = NULL
 WHERE comision IS NOT NULL;
```

No. La orden anterior se podría haber escrito así:

```
UPDATE emple
   SET comision = NULL
     , salario = salario + comision
 WHERE comision IS NOT NULL;
```

Intercambiar el valor de dos columnas:

```
UPDATE emple
   SET salario = comision
     , comision = salario
WHERE oficio = 'VENDEDOR';
```

Un `UPDATE` sin `WHERE`:

```
UPDATE emple
   SET salario = salario / 2;
```

Es lo mismo que poner:

```
UPDATE emple
   SET salario = salario / 2
 WHERE true;
```

¡Modifica todas las filas de la tabla!