SQL: Lenguaje de acceso a bases de datos

Israel Herraiz, Igor Arambasic

2 de abril de 2016

Contenidos

Introducción a las bases de datos

Sobrecarga de información en el mundo actual

- Demasiada información, solo puede tratarse por medios automáticos e informáticos
- Los datos pueden ser en cualquier formato y de cualquier tipo
  - Texto en diferentes idiomas
  - Imágenes
  - Sonidos
  - Vídeos
- No podemos permitirnos perder información
- La información tiene que estar almacenada de manera segura y permanente
- Accesible por múltiples usuarios
- Fácilmente actualizable

Vamos a intentar hacer nuestro propio Facebook

¿Qué información y datos son la base del funcionamiento de Facebook?

¿Qué propiedades tienen que cumplir esos datos y su almacenamiento?

Sistemas de gestión de base de datos

El sistema de gestión de base de datos (SGBD) controla la base de datos. La mejor opción dependerá de las características de las que se encarga el SGBD.

- Almacenamiento
  La base de datos está en uno o varios ficheros en el disco.
- Número de usuarios
  En la mayoría de las aplicaciones varios usuarios accederán a la vez a la base de datos.
- Seguridad
  No todos los usuarios pueden acceder a los mismos datos, ni con los mismos permisos. Los datos están protegidos de accesos no autorizados.

Sistemas de gestión de base de datos

(cont.)

- Rendimiento
  La cantidad de datos, el tipo de datos, el número de usuarios, influyen en el rendimiento (la velocidad y los recursos que necesitará el SGBD para dar una respuesta).
- Escalabilidad
  ¿Es fácil migrar a una plataforma diferente? ¿Y añadir más capacidad a la base de datos?
- Coste
  Algunos SGBD son muy caros, otros literalmente se pueden obtener de manera gratuita.

¿Cuáles son los SGBD más habituales?

SGBD	Coste	Entorno
IBM DB2	Medio	Profesionales
Oracle	Alto	Profesionales, admón. pública
MS SQL Server	Bajo	Web con servidores MS
MySQL	Gratis	Web con servidores LAMP
PostgreSQL	Gratis	Profesionales, web
MS Access	Bajo	Doméstico
LibreOffice Base	Gratis	Doméstico

¿Cómo organizamos los datos en la base de datos?

La actividad de decidir y organizar los datos que se almacenarán en la BD se llama modelado de datos.

Se emplea el lenguaje SQL (*Structrured Query Language*).

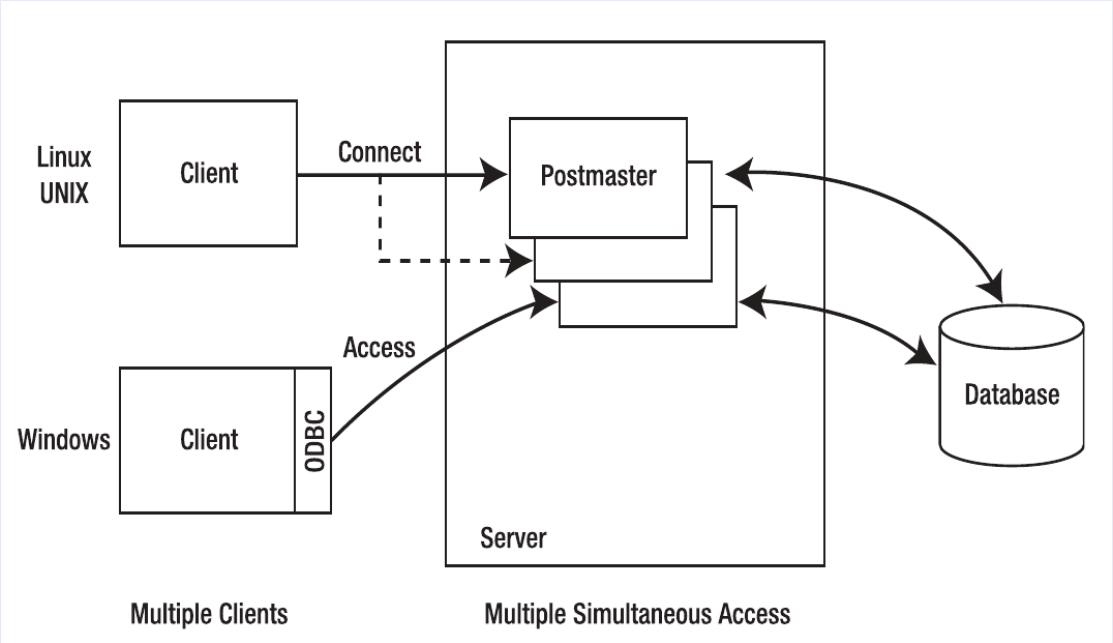
SQL

- Es un estándar ISO, aunque hay diferencias entre SGBD.
- Es un lenguaje declarativo (qué se hace, no cómo se hace).
- Insensible a mayúsculas.
- Para modelar, definimos tablas y esquemas de datos, relaciones entre los datos, y propiedades de manipulación de los datos.

Documentación de referencia
`http://postgresql.org/docs/9.1/static/sql-commands.html`
`http://postgresql.org/docs/9.1/static/sql.html`

¿Cómo funciona un SGBD?

Arquitectura cliente-servidor



PostgreSQL

El servidor escucha normalmente en el puerto 5432 TCP. En Ubuntu, la versión 9.1 está configurada para escuchar en el puerto 5434 TCP.

Primeros pasos en SQL

Estas operaciones se pueden hacer usando el gestor `pgAdmin`, o directamente en SQL.

Crear una BD

```
CREATE DATABASE facebook;
```

Eliminar una BD

```
DROP DATABASE facebook;
```

Tablas

Los datos se organizan en tablas. Las tablas tienen columnas, que pueden ser de diferentes tipos (texto, números, fechas).

Crear y borrar una tabla

```
CREATE TABLE amigos (nombre VARCHAR, edad INT, email VARCHAR);

DROP TABLE amigos;
```

Añadir registros a una tabla

```
INSERT INTO amigos VALUES ('Lionel Messi',24,'messi@fcbarcelona.es');
```

Extraer información

Los datos se extraen el comando SELECT

Leer todas las columnas

```
SELECT * FROM amigos;
```

Leer solo algunas columnas

```
SELECT nombre, edad FROM amigos;
```

Filtrar información

Igual que seleccionamos columnas, podemos seleccionar solo las filas que cumplan determinadas condiciones.

Ejemplo de filtro

SELECT $*$ **FROM** amigos **WHERE** edad $<$ 25;

Distinto que

SELECT nombre, edad **FROM** amigos **WHERE** edad <> 24;

Filtrar información

Ejercicio

Crear una tabla de amigos para la base de datos facebook.

- Borrar previamente cualquier otra tabla que pudiera existir en la base de datos.
- Genera la BD de Facebook con una tabla donde cada persona estará identificada por su nombre, edad, lugar de residencia y dirección de correo electrónico
- Tiene que contener al menos 10 personas (anota las consultas usadas para añadir la información).
- Escribe una consulta para encontrar a los menores de edad o los mayores de 65 años.
- Escribe una consulta para encontrar a las personas que no viven en Madrid.

Borrar información

Del mismo modo que seleccionamos información que cumple unos determinados criterios, podemos borrar de manera selectiva registros.

Borrar todos los registros

```
DELETE FROM amigos;
```

Borrar todos los menores de edad

```
DELETE FROM amigos WHERE edad < 18;
```

Diferencias entre DROP y DELETE

Pregunta

Hemos visto los comandos DROP y DELETE, que sirven para quitar información de la base de datos.
¿Cuál es la diferencia entre ambos comandos?
¿En qué situaciones usarías uno y otro?

Diferencias entre DROP y DELETE

Pregunta

Hemos visto los comandos DROP y DELETE, que sirven para quitar información de la base de datos.
¿Cuál es la diferencia entre ambos comandos?
¿En qué situaciones usarías uno y otro?

Respuesta

DROP borra por completo una tabla de la base de datos, y no puede eliminar de manera selectiva. Se usa si ya no necesitamos una tabla, si estamos haciendo pruebas o si hay que volver a crear una tabla.

DELETE borra filas (registros) de una tabla, de manera selectiva, y deja el resto de la tabla intacta. Se usa por ejemplo para eliminar un usuario de nuestro sistema.

Cambiar una tabla sin borrarla

Es habitual que necesitemos cambiar el esquema de una tabla para añadir columnas, o eliminar columnas que ya no necesitemos. El resto de datos no se toca en absoluto.

Añadir una columna

```
ALTER TABLE amigos ADD COLUMN telefono VARCHAR;
```

Eliminar una columna

```
ALTER TABLE amigos DROP COLUMN telefono;
```

Actualizar información

Podemos cambiar datos específicos de una tabla sin tocar el resto, usando el comando UPDATE.

Cambiar el lugar de residencia

```
UPDATE amigos SET residencia = 'Getafe';
```

Bloquear menores de edad

```
UPDATE amigos SET estado = 'bloqueado' WHERE edad < 18;
```

Ejemplo

Ejercicio

Añade una columna para el número de teléfono y otra para que indique su nombre de usuaro

- ¿Cuál es el contenido de esas columnas justo tras haberlas creado?
- Actualiza el número de teléfono de cada una de las personas de la tabla amigos, usando cualquier número que se te ocurra.
- Todos los amigos de la tabla usan su dirección de email como nombre de usuario, así que ambos datos coinciden. ¿Cómo puedes aprovechar los datos que ya están en la tabla para actualizar las columna con la dirección del chat?

Test resumen de introducción a SQL

Responde a las siguientes preguntas

- ¿Qué comando se usa para crear una base de datos?
- ¿Qué comando se usa para crear una tabla? ¿Qué necesitamos especificar?
- ¿Para qué sirve el comando DROP?
- ¿Cómo se extrae información de una base de datos?
- ¿Cuál es la diferencia entre UPDATE y ALTER?

Introducción al diseño de bases de datos

¿Qué es una base de datos relacional?

Abstracción lógica que describe una colección de objetos interrelacionados. Contiene tablas, y dentro de cada tabla, registros. Las tablas pueden establecer relaciones entre sí.

¿Cómo creamos una base de datos?

Primero es necesario crear un modelo de datos. El modelo de datos transforma la información que queremos manejar en un sistema relacional, con tablas, columnas y relaciones entre tablas.
Hay tres niveles de modelado:

- Conceptual
- Lógico
- Físico

Pasos en el modelado

Conceptual → Lógico → Físico

Pasos	Conceptual	Lógico	Físico
Entidades	X		
Relaciones	X	X	
Atributos		X	
Claves primarias		X	X
Claves ajenas		X	X
Tablas / Vistas			X
Columnas			X
Tipos de datos			X

Modelado conceptual

En el modelado conceptual se definen entidades y relaciones.

Entidades

Objetos físicos o conceptos abstractos de la aplicación.

Por ejemplo, en una biblioteca: libros, autores, usuarios de la biblioteca, editoriales.

Relaciones

Las relaciones definen cómo interactúan las entidades.

Por ejemplo, un libro puede tener varios autores, y un usuario puede prestar varios libros.

Las relaciones se definen mediante claves primarias y ajenas.

Modelado lógico

Atributos

Se definen qué atributos tienen las diferentes entidades.

Por ejemplo, un autor tiene un nombre, fecha de nacimiento. Un libro tiene título, autor, ISBN.

La clave está en elegir qué atributos son importantes y cuáles irrelevantes, para no añadir información innecesaria a la base de datos. Por ejemplo, es irrelevante el color de ojos de un autor.

Modelado físico

Detalles relacionados con el SGBD elegido

Implementación de la información anterior en SQL, usando los tipos de datos proporcionados por el SGDB elegido. Es necesario especificar también los nombres de las columnas.

Modelado físico

Detalles relacionados con el SGBD elegido

Implementación de la información anterior en SQL, usando los tipos de datos proporcionados por el SGDB elegido. Es necesario especificar también los nombres de las columnas.

¿Por qué son necesarios tipos de datos?

- ¿Por qué son necesarios? ¿Por qué no simplemente poner todos los campos de todas las tablas como texto?

¿Cómo hacemos un buen modelo de datos?

- El requisito fundamental es entender el problema que estamos intentando modelar.
- Para entender el problema es necesario realizar una ingeniería de requisitos, preguntando a las personas que van a usar el sistema.
- Una vez que tengamos la información necesaria para saber cómo funciona el sistema, podemos empezar con las tres etapas del modelado.
- El modelado es iterativo. Conforme vamos obteniendo un modelo más detallado será necesario volver atrás para cambiar algo decidido en una etapa anterior.

Claves primarias

Una clave primaria identifica de manera única a cada registro de una tabla. Por ejemplo, el DNI podría ser una buena clave primaria para una base de datos de personas.

Buena práctica

Indicar siempre una clave primaria para todas las tablas que creemos.

- La clave primaria no puede estar vacía
- Hay que asegurarse que no hay dos registros con la misma clave primaria (o el SGBD dará error).
- Una vez creada, la clave primaria no debería cambiarse nunca

Ejemplo

Clave primaria de la tabla amigos

```
CREATE TABLE amigos (nombre VARCHAR,
                     email  VARCHAR,
                     edad   INT,
                     PRIMARY KEY (email));
```

Añadir una clave primaria a una tabla ya creada

```
ALTER TABLE amigos ADD PRIMARY KEY (email);
```

Tipos de datos

Los tipos de datos definen qué clase de información podemos almacenar en un campo. SQL define algunos tipos básicos (textos, números), pero la mayoría de los SGBD extiende estos tipos con algunos propios.

Vamos a ver cómo manejar información de cuatro tipos diferentes

- Valores lógicos (*booleanos*)
- Texto
- Números
- Temporales (fechas, horas)

Valores lógicos (*booleanos*)

Es el tipo más sencillo. Solo puede tomar dos valores:

- verdadero (TRUE)
- falso (FALSE)

Se puede escribir de varias maneras (insensible a mayúsculas):

Verdadero	**Falso**
'1'	'0'
'yes'	'no'
'y'	'n'
'true'	'false'
't'	'f'

Texto

Se puede representar un solo carácter, cadenas de texto de longitud fija y cadenas de texto de longitud variable.

Comando	Explicación
`char`	Un único carácter
`char(n)`	Cadena de longitud fija
`varchar(n)`	Cadena de longitud variable, máximo `n` caracteres
`varchar`	Cadena de longitud arbitraria. Específico de PostgreSQL
`text`	Igual que `varchar`. Específico de PostgreSQL

Números

Números enteros:

- `smallint` (entre -32768 y 32767)
- `int`

Números reales:

- `float`

Valores temporales

- `date` para almacenar una fecha
- `time` para almacenar una hora
- `timestamp` para almacenar fecha y hora
- `interval` para almacenar diferencias entre fechas y horas
- `timestamptz` para almacenar fecha y hora con información de la zona horaria. Específico de PostgreSQL

Un tipo especial: NULL

NULL es un tipo especial que representa la ausencia de valor. Cuando un campo no contenga datos, debe contener NULL. Ningún campo debería aceptar NULL como un valor posible, desde el punto de vista de cómo funciona nuestra aplicación.

¿Cómo se introduce un valor NULL?

Cuando creamos campos (añadiendo filas o columnas) sin especificar su contenido, automáticamente el valor que tiene es NULL.

¿Cómo se identifica un valor NULL?

No se puede usar una comparación normal. Hay que usar `IS NULL` o `IS NOT NULL`. Por ejemplo:

```
SELECT * FROM amigos WHERE edad IS NULL;
```

Consultas de información en SQL

Hasta ahora, hemos visto cómo crear la estructura de la base de datos, expresándola en código SQL.
Hemos usado SQL como un lenguaje de definición de datos (DDL por sus siglas en inglés).

Pero SQL es también:

- lenguaje de manipulación de datos (DML)
- lenguaje de consulta de datos (DQL)

Consultas de información en SQL

Hasta ahora, hemos visto cómo crear la estructura de la base de datos, expresándola en código SQL.
Hemos usado SQL como un lenguaje de definición de datos (DDL por sus siglas en inglés).

Pero SQL es también:

- lenguaje de manipulación de datos (DML)
- lenguaje de consulta de datos (DQL)

Alias

Un alias sirve para cambiar momentáneamente el nombre a algo, durante una consulta, para hacer la escritura de la consulta más sencilla, o para visualizar el resultado de una manera más inteligible.

Consulta con un alias

Se especifica con `AS`, y se aplica a columnas, tablas y/o consultas enteras.

```
SELECT *, (edad*2) as doub_edad from amigos where
  (edad*2)>49;

SELECT * FROM amigos AS t WHERE t.residencia = 'Valencia';
```

Eliminar duplicados

Con el comando DISTINCT podemos seleccionar solo resultados que no están duplicados.

Ciudades en nuestra base de datos

SELECT DISTINCT residencia **FROM** amigos;

Eliminar duplicados

Con el comando DISTINCT podemos seleccionar solo resultados que no están duplicados.

Ciudades en nuestra base de datos

SELECT DISTINCT residencia **FROM** amigos;

¿Cuál es la diferencia con esta consulta?

SELECT residencia **FROM** amigos;

Ordenar los resultados

Cuando hacemos una consulta, el orden de los resultados no está garantizado. Normalmente se devuelven en el mismo orden en que se escribieron. Podemos decidir el orden usando el comando ORDER BY.

De menor a mayor

```
SELECT * FROM amigos ORDER BY edad;
```

De mayor a menor

```
SELECT * FROM amigos ORDER BY edad DESC;
```

Ordenar por criterios múltiples

Se puede ordenar por varias columnas

```
SELECT * FROM amigos ORDER BY edad, num_amigos;
```

Ordenar en diferentes órdenes

```
SELECT * FROM amigos ORDER BY edad DESC, num_amigos
  ASC;
```

Limitar la cantidad de registros

Los cinco más viejos del lugar

Específico de PostgreSQL

```
SELECT * FROM amigos ORDER BY edad DESC LIMIT 5;
```

Mismo ejemplo, en SQL estándar

```
SELECT * FROM amigos ORDER BY edad DESC FETCH
FIRST 5 ROWS ONLY;
```

Limitar la cantidad de registros

Con OFFSET podemos empezar la cuenta en la fila que queramos.
Es un comando SQL estándar.

Los (siguientes) tres más viejos del lugar

```
SELECT * FROM amigos ORDER BY edad DESC OFFSET 5
LIMIT 3;
```

Mismo ejemplo, en SQL estándar

```
SELECT * FROM amigos ORDER BY edad DESC FETCH
OFFSET 5 FIRST 3 ROWS ONLY;
```

Operadores en consultas SQL

Operadores lógicos

- ALL
- AND
- ANY
- BETWEEN
- IN
- NOT
- OR

Ejemplos con operadores

SELECT * **FROM** amigos **WHERE** edad **BETWEEN** 18 **AND** 30;

SELECT * **FROM** amigos **WHERE** edad **NOT IN** (15, 16, 17);

Agregar y agrupar registros

Funciones que usan información compuesta

- `AVG`
- `COUNT`
- `MAX`
- `MIN`
- `SUM`

Ejemplos

SELECT COUNT(∗) **FROM** amigos;

SELECT AVG(edad), **MIN**(edad), **MAX**(edad) **FROM** amigos;

Consultas por grupos

Podemos usar GROUP BY para agrupar los resultados por uno o varios campos, y aplicar las funciones anteriores solo a los campos dentro de cada grupo.

Ejemplo

```
SELECT residencia, COUNT(*), AVG(edad) FROM amigos
  GROUP BY residencia;
```

Consultas por grupos

Podemos usar GROUP BY para agrupar los resultados por uno o varios campos, y aplicar las funciones anteriores solo a los campos dentro de cada grupo.

Ejemplo

```
SELECT residencia, COUNT(*), AVG(edad) FROM amigos
  GROUP BY residencia;
```

Pregunta

¿Qué calcula la consulta anterior?

Más sobre GROUP BY

Agrupar por varios criterios

```
SELECT residencia, edad, COUNT(*) FROM amigos GROUP
BY residencia, edad;
```

Agrupar si cumplen una condición

```
SELECT residencia, COUNT(*), AVG(edad) FROM amigos
GROUP BY residencia HAVING AVG(edad)>20;
```

Consultas múltiples

Cuando hacemos una consulta, el resultado no es más que una tabla temporal, que tiene las mismas propiedades que el resto de tablas. Por tanto, podemos aplicar a su vez consultas sobre los resultados de consultas.
Es obligatorio asignar un alias a cada una de las consultas que vamos a consultar a su vez.

Dentro de la cláusula `WHERE` podemos hacer consultas y comprobarlas usando los comandos IN, EXISTS, ANY y ALL.

Seleccionar nombre de los paises con mas de 150 aeropuertos

```
select country_code, country_name from country_names where
  country_code IN (select country_code from airports where
  nairports>150);
```

JOINS

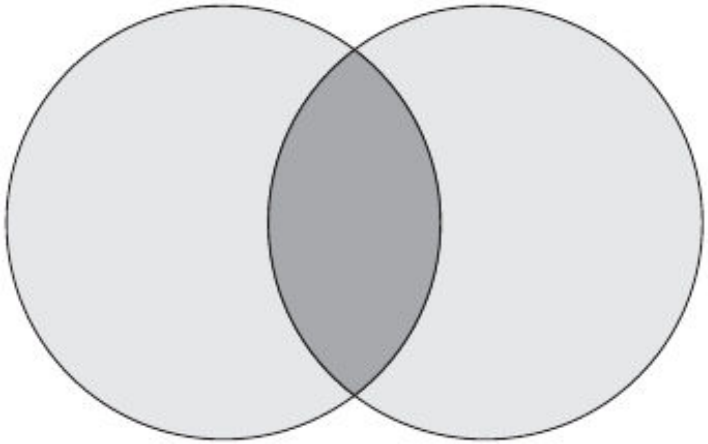
Cuando la consulta múltiple extrae información de varias tablas, también se pueden usar JOIN.
La combinación puede ser de varias formas: INNER JOIN (solapamiento), LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL JOIN

Redundancia y rendimiento

El modelo de datos debe evitar la redundancia de datos. Esto facilita el mantenimiento y la integridad de los datos.
Las consultas múltiples son más lentas y pueden consumir más memoria. El modelo de datos debe facilitar las consultas múltiples que necesitemos, pero siempre sin caer en la duplicidad de datos.

INNER JOIN

Combina datos de varias tablas, y devuelve el solapamiento entre todas las tablas.



Codigos de paises con mas de 100 aeropuertos y mas de 20M personas

```
SELECT i.country_code FROM population AS i INNER JOIN
  airports AS d ON i.country_code= d.country_code;
```

Sintaxis alternativa para INNER JOIN

El estándar SQL prefiere la forma con INNER JOIN a la alternativa, pero ambas son válidas.

Otra manera de escribirlo

```
SELECT i.country_code FROM population AS i, airports AS d
WHERE i.country_code= d.country_code;
```

Sintaxis alternativa para INNER JOIN

El estándar SQL prefiere la forma con INNER JOIN a la alternativa, pero ambas son válidas.

Otra manera de escribirlo

```
SELECT i.country_code FROM population AS i, airports AS d
WHERE i.country_code= d.country_code;
```

Ejercicio

Escribe una consulta para extraer codigo, nombre de pais y numero de habitanes utilizando la tabla de los nombres y tabla de populacion. Ordena el resultado por numero de habitantes de mayor a menor

JOIN con varias tablas

Un INNER JOIN puede hacerse con tantas tablas como queramos.

Pais, Aeropuertos y populacion

```
    SELECT a.country_code, a.country_name, b.npop, c.nairports
FROM population AS b
INNER JOIN country_names AS a ON a.country_code=
    b.country_code
INNER JOIN airports AS c ON a.country_code= c.country_code
 ORDER by b.npop DESC;
```

LEFT OUTER JOIN

Todos los registros de la tabla izquierda, junto con los datos adicionales que puedan tener en la tabla derecha.

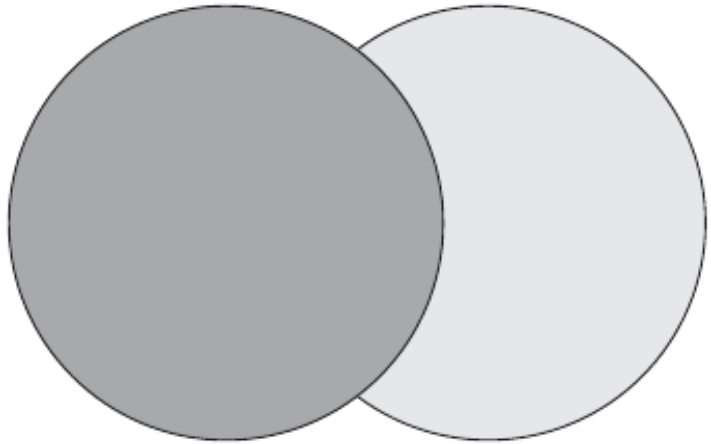


Tabla de populacion m̈ejoradac̈on datos de tabla de aeropuertos

```
SELECT i.country_code, i.npop, d.nairports
FROM population AS i
LEFT OUTER JOIN airports AS d
ON i.country_code= d.country_code;
```

Ejemplo

¿El resultado es el mismo? ¿Cuál devolverá más registros?

```
SELECT i.country_code, i.npop, d.nairports
FROM population AS i
LEFT OUTER JOIN airports AS d
ON i.country_code= d.country_code;
```

```
SELECT i.country_code, i.npop, d.nairports
FROM population AS i
INNER JOIN airports AS d
ON i.country_code= d.country_code;
```

RIGHT OUTER JOIN

Todos los registros de la tabla derecha, junto con los datos adicionales que puedan tener en la tabla izquierda.

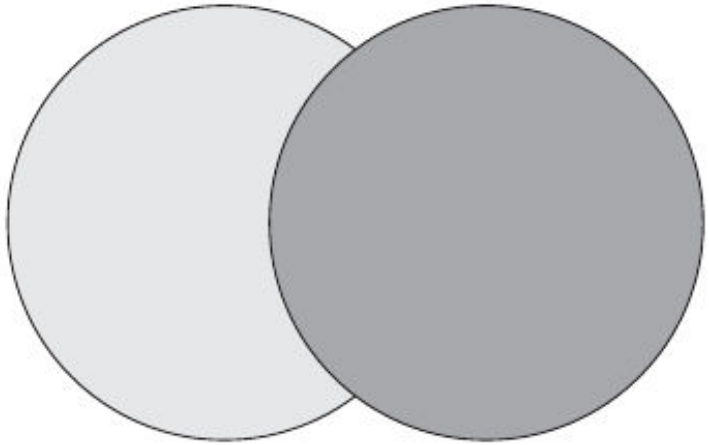
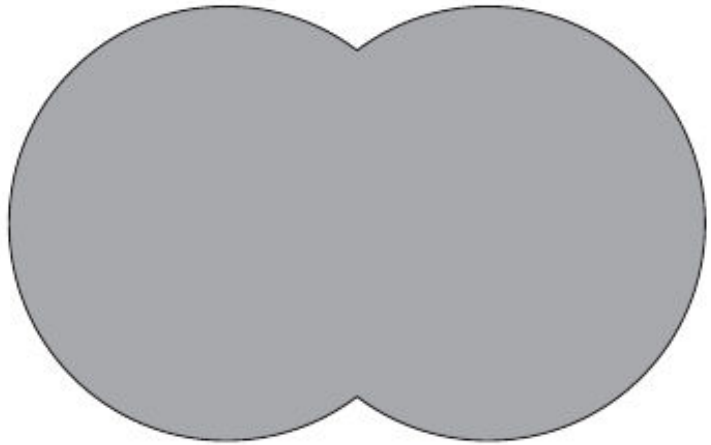


Tabla de aeropuertos m̈ejoradaẗon datos de tabla de populacion

```
SELECT d.country_code, i.npop, d.nairports
FROM population AS i
RIGHT OUTER JOIN airports AS d
ON i.country_code= d.country_code;
```

FULL JOIN

Combinación del LEFT OUTER JOIN y del RIGHT OUTER JOIN



Todos los aeroupertos con populacion

```
SELECT d.country_code, i.npop, d.nairports
FROM population AS i
FULL JOIN airports AS d
ON i.country_code= d.country_code;
```

Ejemplo

Ejercicio

Usando las tablas aeroupertos y populacion y las consultas que hemos visto en los ejemplos con los JOIN, ordena los siguientes de mayor a menor número de resultados:

- INNER JOIN
- LEFT OUTER JOIN
- FULL JOIN
- RIGHT OUTER JOIN

Ejercicio

Ejercicio

- Importa los datos de trafico a Postgres (/Data/us_dot/traffic/).
- Encuentra numero de aerolineas que vuelan a MAD.
- Encuentra promedio de factor de ocupación de las aerolíneas en los vuelos a MAD.
- Para las aerolineas que vuelan desde MAD presenta en una columna promedio de factor de ocupacion para vuelos desde MAD y en otra columna promedio de factor de ocupacion de la aerolinea en general.
- Encuentra aerolinea que tiene mayor numero de vuelos de largo recorrido (donde origen y destino estan en continentes distintos). (utiliza los datos de trafico junto con optd_por_public)