

```
each: function(e, t, n) {
    var r, i = 0,
        o = e.length,
        a = M(e);
    if (n) {
        if (a) {
            for (; o > i; i++)
                if (r = t.apply(e[i], n), r ===
                    void 0)
                    e[i] = null;
        } else
            for (i in e)
                if (r = t.apply(e[i], n),
                    r === void 0)
                    e[i] = null;
    } else if (a) {
        for (; o > i; i++)
            if (r = t.call(e[i], i, n),
                r === void 0)
                e[i] = null;
    } else
        for (i in e)
            if (r = t.call(e[i], n),
                r === void 0)
                e[i] = null;
    return e
},
trim: b && !b.call("\ufeff"),
      return null == e ?
} : function(e) {
      return null ==
},
makeArray: function(e) {
    var n = t(e);
    return n === null
        ? []
        : n;
},
isArray: function(e) {
    var n = t(e);
    return n === null
        ? false
        : n
}
```

## Fastbook 05

# Tratamiento de Datos (Excel y SQL)

El lenguaje SQL



## 05. El lenguaje SQL

```
        if (r = t.apply(e[i], n), r === !1) break;
    } else
      for (i in e)
        if (r = t.apply(e[i], n), r === !1) break;
    } else if (a) {
      for (i: a > i; i++)
        if (r = t.call(e[i], i, e[i]), r === !1) break;
    } else
      for (i in e)
        if (r = t.call(e[i], i, e[i]), r === !1) break;
  },
  trim: b && !b.call("\ufe0f\ufe0a0") ? function(e) {
    return null == e ? "" : b.call(e)
  } : function(e) {
    return null == e ? "" : (e + "").replace(C, "")
  },
  makeArray: function(e, t) {
    var n = t || [];
    return null != e && (!Object(e)) ? x.merge(n, "string" == typeof e ? [e] : e) : h.call(n, e)), n
},
```

En este fastbook, introduciremos el lenguaje SQL, la interfaz de comunicación que, a través del SGBD, nos permitirá realizar las operaciones necesarias para administrar nuestra base de datos.

Pero antes de dar nuestros primeros pasos con el lenguaje, conoceremos las distintas etapas por las que tendremos que pasar en el proceso de creación de nuestra base de datos.

Una vez que tengamos claro el plan de viaje, nos adentraremos en el lenguaje analizando tanto el léxico como la sintaxis necesaria para poner en práctica los comandos del lenguaje.

*Autor: Carlos Manchón y Breogán Cid*

Diseño de la base de datos

Fase 0: Toma de requisitos

Fases 1 y 2: diseño conceptual y lógico

Fase 3: diseño físico

Primeros pasos con SQL

Tipos de datos

Conclusiones

# Diseño de la base de datos

X Edix Educación

---

Nuestro **objetivo principal** a la hora de crear una base de datos es crear un almacén de información que nos sea útil y que podamos manejar cómodamente. Por eso, antes de empezar a conocer la sintaxis del lenguaje que nos permita trabajar con ella, nos centraremos en todos los pasos que debemos de seguir y tener en cuenta, un estudio necesario y previo a empezar a escribir el primer comando.



De la misma forma que, si queremos construir un edificio, no empezamos a colocar ladrillos nada más ver el terreno, en nuestro caso tampoco podremos avanzar hasta que tengamos nuestro **diseño con el plano inicial de nuestro boceto de edificio**.

Para ellos, deberemos seguir los siguientes pasos:

- Fase 0: Toma de requisitos.
- Fase 1: Diseño conceptual.
- Fase 2: Diseño lógico.
- Fase 3: Diseño físico.

# Fase 0: Toma de requisitos

X Edix Educación

---

Antes de realizar el diseño de la base de datos debemos tener claro qué problema vamos a resolver y qué información vamos a almacenar.

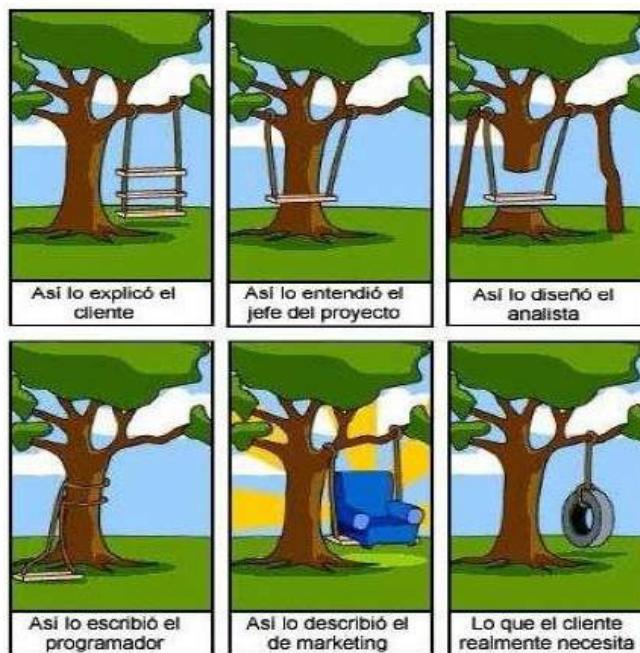
Tenemos que realizar un análisis en profundidad del problema identificando todas las necesidades o requisitos que tengan todas las personas que usarán nuestra base de datos.

---

Esta fase es crucial para la correcta ejecución del resto del proceso.

---

La siguiente viñeta escenifica con mucho humor la importancia capital de esta fase.



# Fases 1 y 2: diseño conceptual y lógico

X Edix Educación

---

## Fase 1: Diseño conceptual

El objetivo de esta **fase de diseño** consiste en traducir la información que los usuarios finales nos han proporcionado a través de estándares para que las personas implicadas en fases posteriores lo entiendan.

## Fase 2: Diseño lógico

Para esta etapa utilizaremos otro estándar más técnico para modelar la información dada por la fase anterior (esquema conceptual). Es en esta fase cuando se debe elegir qué tipo de SGBD vamos a usar (relacional, orientado a objetos, objeto-relacional...). En nuestro caso, nos centraremos en el mundo relacional, por lo que aplicaremos el **modelo relacional**.

---

La teoría de este modelo se basa en la existencia de unos objetos (entidades), los cuales poseen una serie de propiedades (atributos) y las asociaciones existentes entre esas entidades (relaciones).

## 1

## Entidades

Una entidad es cualquier ‘concepto’ del que queramos **almacenar información en una base de datos**: pueden ser **físicas** (personas, objetos...) o **abstractas** (un evento de calendario, un presupuesto...).

Las **entidades** dentro del modelo de entidad/relación se representan gráficamente como un rectángulo con su nombre en el interior. Diferenciaremos dos tipos de entidades:

- **Fuertes.**
- **Débiles.** Una entidad débil depende de otra entidad, pues por sí misma no tiene identidad propia.

Ejemplo: Si queremos crear una base de datos para gestionar el proceso de venta de coches de un concesionario, podríamos tener estas **tres entidades**: COCHE, CLIENTE, VENTA, siendo VENTA una entidad débil, pues una venta por sí misma no tiene significado propio sin estar asociada a un cliente y un coche.



Coche

Cliente

Venta

## 2

## Atributos

Son las **propiedades** que representan las características de una entidad y las que permiten diferenciar dos elementos de una misma entidad.

Los atributos de una entidad tienen un **rango de valores** que llamaremos dominio del atributo. Así, por ejemplo, para la entidad PERSONA, que tiene el atributo edad, el dominio de este será de 0 – 120.

Los **atributos** pueden ser:

- **Identificativos:** Representan únicamente a la entidad sobre otras entidades del mismo tipo.
- **Descriptores:** Proporcionan características de la entidad.

---

Los atributos se representan gráficamente como círculos  
unidos a la entidad mediante una línea recta.

---



3

### Relaciones

Una relación es la asociación que existe entre dos o más entidades y suele llegar definida su función en su nombre.

El grado de una relación determina el número de entidades participantes, es decir, una con dos entidades (COCHE-CLIENTE) se llamará de **grado 2 o binaria**, con tres entidades (COCHE-CLIENTE-CONCESIONARIO) se llamará de **grado 3 o ternaria**, y así sucesivamente.

---

Una relación se identifica gráficamente mediante un rombo que une las entidades participantes.

---



Un punto importante de las relaciones que nos interesará cuando realicemos consultas es su cardinalidad.

---

Llamaremos cardinalidad de una relación binaria al número de ocurrencias de una entidad asociadas a una ocurrencia de la otra entidad.

Existen tres tipos de cardinalidades binarias:

#### Relación uno a uno (1:1)

Dentro de la relación binaria, por cada ocurrencia de la primera entidad, existe una única ocurrencia de la otra entidad y viceversa. Por ejemplo, entidades 'PROFESOR' y 'AULA' con la relación 'ES\_TUTOR'. Un profesor es tutor de una única aula y un aula solo tiene un tutor.

### Relación uno a muchos (1:N)

Dentro de la relación binaria, por cada ocurrencia de la primera entidad, existen 'n' ocurrencias de la otra entidad. Por ejemplo, entidades 'PEDIDO' y 'CLIENTE' con la relación 'REALIZA'. Un cliente podrá realizar 'n' pedidos, pero un pedido concreto solo ha podido ser realizado por un cliente.

### Relación muchos a muchos (N:M)

Dentro de la relación binaria, una ocurrencia de la primera entidad puede estar relacionada con 'm' de la segunda entidad, y viceversa, un elemento de la segunda entidad puede estar relacionado con 'n' ocurrencias de la primera entidad. Por ejemplo, entidades 'PRODUCTO' y 'CLIENTE' con la relación 'COMPRA'. Un cliente puede comprar 'm' productos. Y a su vez, un producto ha podido ser comprado por 'n' clientes.

Gráficamente, reflejamos también la cardinalidad de las entidades, **colocando el valor que corresponda en la línea** que une la entidad con la relación.



## Fase 3: diseño físico

X Edix Educación

---

Es el resultado de **aplicar el esquema lógico** generado en la fase anterior sobre un SGBD concreto. Es decir, mediante el uso de todas las sentencias SQL podremos realizar operaciones sobre la base de datos, para conseguir plasmar nuestro diseño.

---

SQL es un lenguaje declarativo de alto nivel que nos proporciona una interfaz donde especificar el resultado que queremos obtener, dejando al SGBD la decisión de cómo ejecutar la consulta y la posible optimización que realice previamente.

Está compuesto por **varios sublenguajes**, cada uno dirigido a una funcionalidad concreta. Son los siguientes:

1

Lenguaje de **definición de datos** (DDL, Data Definition Language).

2

Lenguaje de **manipulación de datos** (DML, Data Manipulation Language).

3

Lenguaje de **control de datos** (DCL, Data Control Language).

4

Lenguaje de **control de transacciones** (TCL, Transaction Control Language).



## Data Definition Language (DDL)

El **lenguaje de definición de datos** se utiliza para definir el esquema conceptual de la base de datos. Con este lenguaje podemos crear, modificar o borrar los objetos de la base de datos como tablas, vistas, esquemas...

Sus **comandos principales** son los siguientes:

- **CREATE DATABASE:** Nos permite generar una nueva base de datos vacía.
- **CREATE TABLE:** Creamos una nueva tabla con el esquema de datos que le indiquemos.
- **CREATE INDEX:** Crea un índice en una tabla determinada.
- **ALTER TABLE:** Modifica la definición de la tabla, añadiendo o eliminando columnas, restricciones, etc. sobre la misma.

- **DROP INDEX:** Es el comando para borrar el índice de una tabla que hayamos generado previamente.
- **DROP TABLE:** Elimina la definición de la tabla, sus datos, índices y restricciones si los tuviera.
- **TRUNCATE TABLE:** Elimina el contenido de la tabla, pero deja intacta su definición.
- **RENAME TABLE:** Este comando nos permite renombrar objetos dentro de la base de datos.

## Data Manipulation Language (DML)

El lenguaje de manipulación de datos es un lenguaje que nos permite acceder o manipular los datos de la base de datos, según el modelo de datos que los haya generado previamente.

Si en el caso de DDL su utilización afectaba al esquema de la base de datos, entonces, el que se ve afectado es el contenido de la base de datos. Nos permite realizar operaciones de inserción, actualización y borrado. Igualmente, nos permite realizar consultas sobre la base de datos.

Los principales comandos son los siguientes:

### INSERT TABLE

Nos permite añadir información a la tabla indicada.

## UPDATE TABLE

Nos permite actualizar el contenido de una tabla, atendiendo a una condición.

## DELETE TABLE

Parecido a TRUNCATE, en este caso, indicamos una condición para que sea un borrado condicionado.

## SELECT FROM TABLE

Es la sentencia que nos permite extraer información de la tabla (o las tablas) que estemos consultando.

## Data Control Language (DCL)

El lenguaje de control de datos nos permite establecer derechos de acceso de los usuarios sobre los distintos objetos de la base de datos.

Algunos ejemplos de **comandos incluidos en DCL** son los siguientes:

- **GRANT:** Otorga permisos a uno o varios usuarios para realizar tareas determinadas.
- **REVOKE:** Quita permisos que previamente se había concedido con la cláusula GRANT.

Las acciones sobre las que se puede otorgar o eliminar permisos son:

### **CONNECT**

Se refiere al permiso de conexión a la base de datos.

### **SELECT**

Se refiere al permiso a realizar consultas sobre una tabla o base de datos.

### **INSERT**

Se refiere al permiso de realizar inserciones sobre una tabla o base de datos.

### **UPDATE**

Se refiere al permiso de realizar actualizaciones de datos sobre una tabla o base de datos.

### **DELETE**

Se refiere al permiso de realizar eliminaciones de datos sobre una tabla o base de datos.

## Transaction Control Language

Por último, el **lenguaje de control de transacciones** (TCL) proporciona las herramientas para controlar el procesamiento de transacciones en una base de datos. (Una transacción nos permite considerar una serie de instrucciones SQL como una única instrucción).

Los principales comandos son:

- **COMMIT:** Se utiliza para guardar de forma permanente cualquier transacción en la base de datos.
- **ROLLBACK:** Restaura la base de datos al último estado commit.

# Primeros pasos con SQL

X Edix Educación

---

Ahora que ya conocemos **los comandos incluidos en el lenguaje**, tendremos que profundizar para ver cómo los podemos utilizar, es decir, ahora que ya conocemos el vocabulario básico, empezaremos con la gramática del lenguaje.

## Estructura léxica

Una **entrada SQL** consiste en una **secuencia de comandos**. A su vez, un comando es una **secuencia de tokens** terminados por un punto y coma (;).

Así pues, podríamos **ejecutar en nuestro SGBD un script** (un listado de comandos) con la siguiente estructura:

SECUENCIA SQL 1;

SECUENCIA SQL 2;

...

SECUENCIA SQL n;

Dependiendo del comando que queramos lanzar, los tokens a emplear serán unos u otros. Un **token** puede ser:

- Una **palabra reservada** (palabras que el lenguaje reconoce internamente y tienen un significado propio).
- Un **identificador**.
- Un **literal** (una constante, por ejemplo, una cadena de texto o un número).

---

**Los tokens van separados por un espacio en blanco.**

---

Por ejemplo, el siguiente **input** sería sintácticamente correcto:

```
SELECT * from tabla_empleados;  
UPDATE tabla_empleados SET sueldo = sueldo * 1.10;  
INSERT INTO tabla_empleados VALUES (10,'Juan Jiménez',1200);
```

Se trata de una secuencia de tres comandos, separados por ';'. En él, podemos ver que, en el primer comando, seleccionamos todos los campos de la tabla 'tabla\_empleados' (el '\*' en ese comando cumple la función de 'todos los campos'), para posteriormente, en el siguiente comando actualizar la tabla, concretamente, el campo sueldo y asignarle un nuevo valor igual al anterior incrementado un 10%.

---

Por último, **insertamos en la tabla una nueva fila**, con los valores '10', 'Juan Jiménez' y '1200' (podemos presuponer qué significa cada valor, pero sin una especificación de los metadatos de la tabla, es decir, la especificación de los campos, su tipo y su descripción, no podemos confirmar nada).

## Palabras reservadas e identificadores

Del script anterior, habrás podido denotar que existen muchas palabras reservadas (o clave), por ejemplo: SELECT, FROM, INSERT, INTO, VALUES, UPDATE, SET... Como decíamos, son palabras que tienen significado propio en el lenguaje SQL.

Por otro lado, los **identificadores representan objetos de una base de datos**. En nuestro ejemplo, el identificador 'tabla\_empleados' hace referencia a una tabla que existe en la base de datos que contiene un listado de profesores.

## Constantes

Las constantes **representan un valor concreto, fijo**. Pueden ser de distintos tipos, según lo que se necesite representar. Así pues, principalmente suelen ser **cadenas de texto** (en nuestro ejemplo, 'Juan Jiménez') o **numéricas** (en nuestro ejemplo, 1.10).

---

**En el caso de las cadenas de texto, deben ir acompañadas al principio y al final de una comilla simple (').**

---

## Operadores

Un operador es una **secuencia de caracteres** que tienen asociada una operación predeterminada. En nuestro ejemplo, tenemos el operador '\*' que representa la **operación de multiplicación entre operandos de tipo numérico** (si te fijas, en el primer comando, también se usa el carácter '\*', pero, en este caso, no representa una operación de multiplicación).

---

Es el intérprete SQL, el encargado de chequear que las sentencias SQL que escribamos, se adaptan y cumplen con la especificación sintáctica del lenguaje SQL, el que evalúa qué representa en cada momento el carácter '\*':

Como ya imaginarás, **PostgreSQL ofrece un catálogo inmenso de operadores para trabajar con variables de distinto tipo**. Como punto negativo, quitando los operadores aritméticos y de comparación, la gran mayoría no son cubiertos por el estándar, por lo que deberías tener en cuenta esta circunstancia si en algún momento se pretende migrar la base de datos a otro SGBD.

## Caracteres especiales

Existen caracteres que **tienen un significado distinto según sean o no operadores** (acabamos de ver el ejemplo de '\*'). Otros casos pueden ser:

- Paréntesis.
- Corchetes
- Comas.
- Punto y coma.
- Dos puntos.
- Punto...

## Comentarios

Un comentario es una secuencia de caracteres que comienza por — y se extiende hasta el fin de la línea. Alternativamente, para comentarios multilínea, empleamos al principio /\* y al final del comentario \*/.

## Precedencia de los operadores

En los lenguajes de programación, al igual que en las operaciones matemáticas, no todas las operaciones tienen la misma prioridad que las demás, por ejemplo, seguramente ya sepas que el operador de multiplicación tiene más prioridad que el operador de suma, es decir, la multiplicación precede a la suma.

Por ejemplo: si tenemos la expresión  $2 + 5 * 10$ , la correcta evaluación sería  $2 + (5*10)$ .

# Tipos de datos

X Edix Educación

---

PostgreSQL cuenta con numerosos tipos de datos, muchos de ellos a partir de la **especificación del estándar SQL**, otros muchos son **extensión del lenguaje** y quedan fuera del estándar.

1

## Datos numéricos

Representan un número, este puede ser entero o decimal. Existen **numerosos tipos numéricos** dependiendo del rango de valores que queramos expresar. Así pues:

- Para **valores enteros** tenemos los tipos (entre otros): smallint, integer y bigint.
- Para **valores decimales** tenemos: decimal, numeric, real y double precisión.

2

## Datos de tipo texto

Representan una secuencia de caracteres. Dependiendo de la longitud de las cadenas podremos seleccionar un tipo u otro. Por ejemplo, usaremos el tipo 'character varying(n)' y 'varchar(n)' para cadenas de longitud variable con un máximo de 'n' caracteres. Para el caso de **cadenas de longitud fija**, podemos usar los tipos 'character(n)' y 'char(n)'.

---

Estos cuatro tipos están previstos por el estándar SQL, además, PostgreSQL añade un tipo nuevo text, el cual no tiene límite de longitud.

---

 3

### Datos de tipo fecha/tiempo

PostgreSQL soporta por completo el conjunto de tipos propuestos por SQL para tipos de datos fecha/tiempo. Existen **distintos tipos**: , . Existe también el tipo , etc.

- **Date** para fechas.
- **Time** para tiempo.
- **Timestamp** para la combinación de ambos.
- **Interval** para definir un intervalo temporal.

Las posibilidades son infinitas.

 4

### Datos de tipo booleano

Quizás no conozcas el **tipo boolean**, pero es bastante sencillo de entender. Es un tipo de datos que solo posee dos posibles valores: TRUE o FALSE.

## 5

## Otros tipos

Podemos decir que hemos visto las **principales agrupaciones de tipos de datos** que ofrece PostgreSQL y, con lo visto hasta ahora, podemos cubrir prácticamente cualquier circunstancia. No obstante, PostgreSQL ofrece otros tipos para propósitos específicos. Por ejemplo, es famosa la extensión que tiene PostgreSQL para trabajar con bases de datos geográficas.

---

Por si fuera poco, PostgreSQL ofrece mecanismos para crear **nuestro propio tipo de dato complejo** (recuerda, PostgreSQL posee funcionalidades orientadas no solo al mundo relacional, sino también a los objetos).

# Conclusiones

X Edix Educación

---

Como pequeño resumen de lo comentado en este fastbook, destacamos los siguientes aprendizajes:

- Vimos las **distintas fases** que tenemos que seguir a la hora de crear nuestra base de datos.
- También hemos visto el **modelo de entidad - relación**, que nos permitirá definir nuestra base de datos relacional de manera gráfica, para que cualquier usuario pueda conocer su estructura.
- Conocimos los distintos sublenguajes en los que se **estructura SQL** atendiendo su finalidad: **DDL** para la definición de la base de datos, **DML** para su manipulación y consulta, **DCL** para el control de acceso a la base de datos y, por último, **TCL** para la gestión de las transacciones.
- Y para finalizar el tema, hemos dado las **primeras pinceladas del lenguaje SQL**, conociendo los algunos conceptos básicos del lenguaje.

¡Enhорabuena! Fastbook superado

edix

Creamos Digital Workers