

IV. Structured Query Language

Introduction to Data Science

Data Everywhere

June 12, 2020



DataOwl



Overview I

1. Conceptos básicos
2. Tipos de datos y estructuras de datos
3. Queries comunes
4. Funciones para el análisis de datos
5. Introducción a NoSQL



1. Conceptos básicos



1.1 ¿Qué es SQL?

SQL

SQL es acrónimo de **Structured Query Language** (en español Lenguaje Estructurado de Consultas), y es el lenguaje estandar para muchos sistemas de administración de bases de datos relacionales y para el manejo de datos. Se utiliza generalmente para consultar, insertar, actualizar y modificar datos. En términos simples, SQL es un método para la comunicación entre el programador y la base de datos.



La gracia de SQL es que casi todas sus declaraciones son palabras explícitas y descriptivas. En otras palabras, muchos de los comandos que seutilizan en SQL son bastante fáciles de interpretar en comparación con mu-chos otros lenguajes de programación



1.2 Bases de datos

Base de datos

Un concepto importante que debemos manejar antes que todo es la definición de **base de datos**, la cual es básicamente un contenedor con uno o más archivos que se utilizan para organizar y almacenar los datos.

Dentro de las bases de datos podemos encontrar **tablas**, las cuales son listas estructuradas de elementos. Ahora, si profundizamos un poco más en las tablas, nos encontramos con las **filas** y las **columnas**, en donde las filas representan los registros o datos de la tabla, y las columnas individuales son campos o variables.

Indices	Columna 1	Columna 2	Columna 3	Columna 4
Fila 1	a	b	c	d
Fila 2	a	b	c	d

Ejemplo de tabla



1.3 Modelo relacional

Algo importante que diferenciar, es que cuando hablamos de **modelo de datos**, no nos referimos a un modelo matemático; más bien estamos hablando de una manera de organizar nuestros datos y representarlo.

Si bien hay una historia interesante detrás de la evolución de los modelos, nos remitiremos a definir el modelo que utilizaremos para programar en SQL.

Modelo Relacional

Se utiliza un grupo de tablas para representar los datos y las relaciones entre ellos. Cada tabla está compuesta por varias columnas, y cada columna tiene un nombre único.

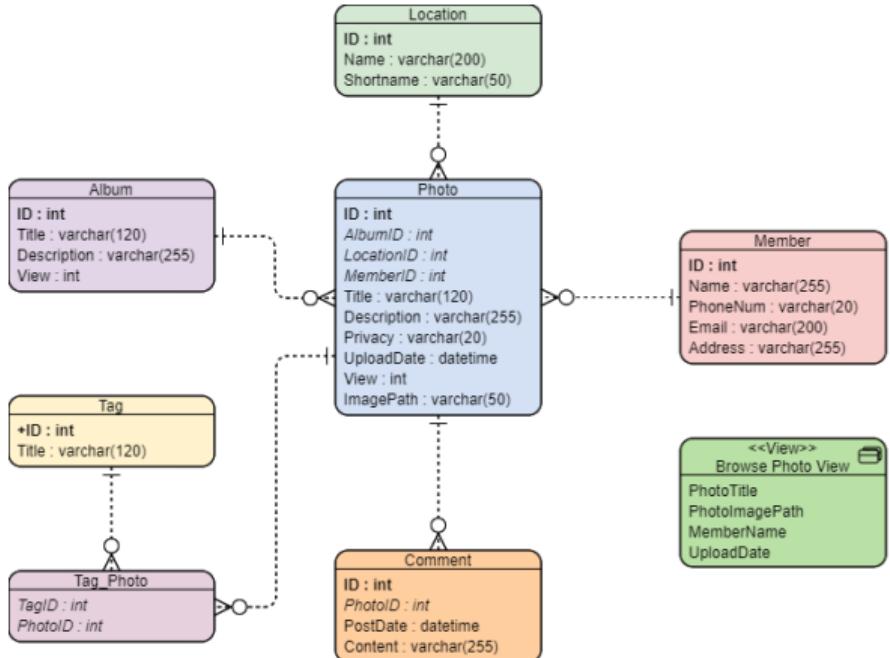
Los modelos relacionales están compuestos por más de una tabla, las cuales están relacionadas entre sí por lo que conocemos como **key** o llave.

En el modelo relacional podemos encontrar tres componentes para su estructura:

- Entidad
- Atributo
- Relación



1.3 Modelo relacional





1.4 RDBMS

RDBMS

Un **RDBMS** es un sistema gestor de bases de datos relacionales. Se trata de software capaz de crear, manipular y gestionar bases de datos de tipo relacional.

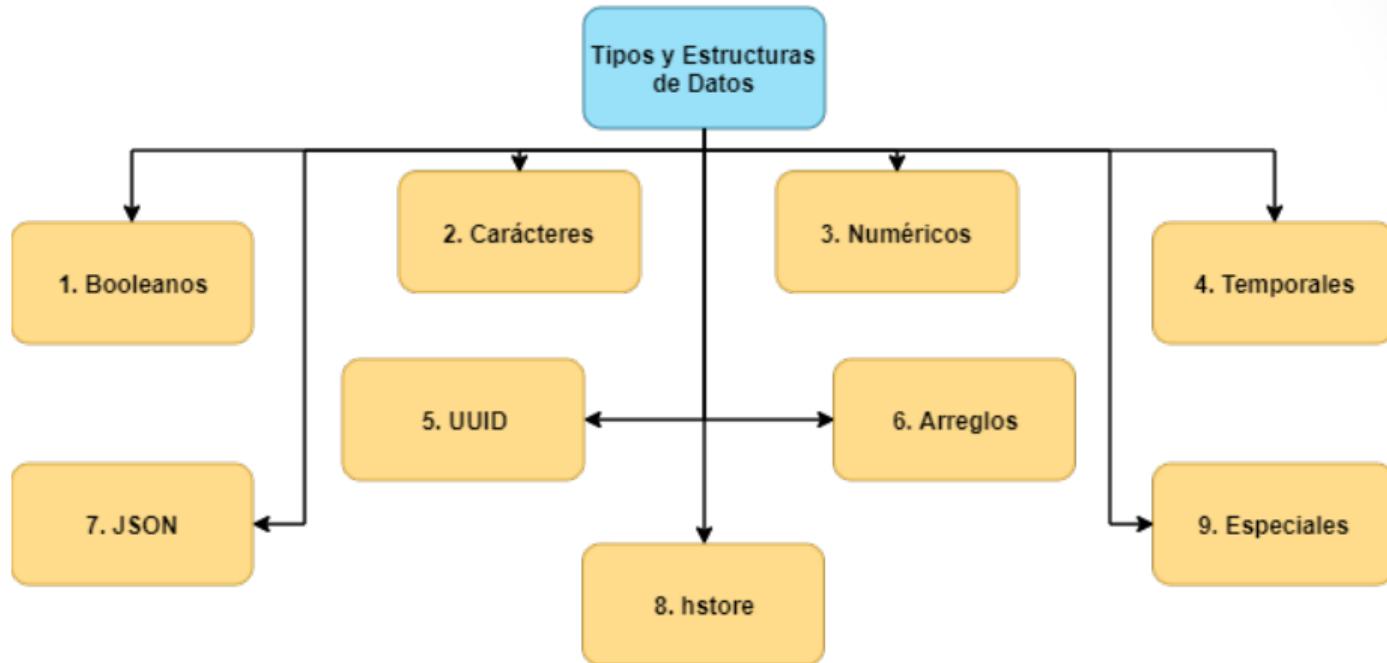




2. Tipos de datos



2. Tipos de datos y estructuras de datos





2.1 Booleanos



El tipo de datos booleano puede contener uno de tres valores posibles: Verdadero, falso o nulo.

Para definir una variable booleana en PostgreSQL podemos usar tanto **bool** o **boolean** para declararla.

Algo interesante, es que al insertar datos en una columna booleana, PostgreSQL los convierte en un valor booleano automáticamente:

- 1, sí, y, t, true: Se convierten en **true**.
- 0, no, n, f, false: Se convierten en **false**.



2.2 Carácteres

PostgreSQL nos proporciona tres tipos de datos de caracteres: **char(n)**, **varchar(n)** y **text**.

- **char(*n*)**: Es un tipo de cadena de caracteres con longitud fija. Si inserta una cadena de texto que es más corta que la longitud *n* de la columna, PostgreSQL rellena espacios. Si inserta una cadena que es más larga que la longitud de la columna, PostgreSQL emitirá un error.
- **varchar(*n*)**: Es un tipo cadena de caracteres de longitud variable, aunque tiene un máximo de hasta *n* caracteres. PostgreSQL no rellena espacios cuando la cadena almacenada es más corta que la longitud de la columna.
- **text(*n*)**: Es un tipo de cadena de caracteres de longitud variable. Teóricamente, los datos del tipo text son una cadena de caracteres con longitud ilimitada.



2.3 Números

En PostgreSQL tenemos dos opciones para los datos de tipo numérico:

Enteros

- **smallint**: Entero en 2 bytes.

(Desde: -32768 Hasta: 32767)

FLOATS

- **float(*n*)**: Número de tipo flotante con precisión *n*.

(Máximo de 8 bytes).

- **int**: Entero en 4 bytes.

(Desde: 2147483648 Hasta: 2147483647)

- **serial**: Generador automático de secuencias.

- **real**: Número de tipo flotante de 4 bytes de precisión.

- **numeric(*p,s*)**: Número de tipo flotante con *p* dígitos y *s* decimales.



2.4 Datos temporales

Los tipos de datos temporales le permiten almacenar datos de fecha y/o hora. PostgreSQL tiene cinco tipos principales de datos temporales:

- **date**: Guarda fechas.
- **time**: Guarda horas.
- **timestamp**: Guarda fecha y hora.
- **timestamptz**: Guarda fecha, hora y zona temporal.
- **interval**: Guarda periodos de tiempo.

DATA_CREA
January, 24 2013 12:33:02+0000
January, 24 2013 12:32:49+0000
January, 24 2013 12:32:29+0000
January, 24 2013 12:25:51+0000
January, 24 2013 12:25:38+0000
January, 24 2013 12:25:10+0000
January, 24 2013 12:24:37+0000
January, 24 2013 12:22:48+0000
January, 24 2013 12:22:12+0000
January, 24 2013 12:21:19+0000
January, 23 2013 14:17:22+0000
January, 22 2013 18:44:53+0000
January, 22 2013 18:44:10+0000
January, 22 2013 17:49:39+0000
January, 22 2013 17:49:21+0000
January, 22 2013 17:46:57+0000



2.X UUID, JSON, Arreglos y hstore

- 2.5 UUID:** El tipo de datos **UUID** permite almacenar identificadores únicos universales definidos como RFC 4122. Los valores de UUID garantizan un mejor identificador en términos de unicidad que serial y se pueden usar para ocultar datos confidenciales expuestos al público, como los valores de id en la URL.
- 2.6 JSON:** PostgreSQL proporciona dos tipos de datos para JSON, estos son **JSON** y **JSONB**. El tipo de datos JSON almacena datos JSON simples que requieren análisis para cada procesamiento, mientras que el tipo de datos JSONB almacena datos JSON en un formato binario que es más rápido de procesar pero más lento de insertar. Además, JSONB admite la indexación, lo que puede ser una ventaja
- 2.7 Arreglos:** En PostgreSQL, se pueden almacenar **arreglos**, ya sean de strings, de enteros, etc. Definir un arreglo puede ser útil en algunas situaciones, por ejemplo, almacenar días de la semana, meses del año, etc.
- 2.8 hstore:** El tipo de datos **hstore** se utiliza para almacenar pares key-value en un solo valor. El tipo de datos hstore es muy útil en muchos casos, como datos semiestructurados o filas con muchos atributos que rara vez se consultan. Tenga en cuenta que las claves y los valores son solo cadenas de texto.



2.9 Otros tipos de datos

Además de los tipos de datos primitivos, PostgreSQL también provee muchos otros tipos de datos especiales, relacionados a la geometría y a las redes.

- **box**: Guarda un rectángulo.
- **line**: Guarda un trío de ordenado de números (A, B, C) , que definen una ecuación de la recta $Ax + By + C = 0$.
- **point**: Guarda un par ordenado de números (x, y) .
- **lseg**: Guarda dos pares ordenados de puntos $[(x_1, y_1), (x_2, y_2)]$ y representa un segmento entre ellos.
- **polygon**: Guarda una lista de puntos $[(x_1, y_1), \dots, (x_n, y_n)]$ que representan los vértices de un polígono.
- **inet**: Guarda una dirección IP4.
- **macaddr**: Guarda una dirección MAC.



3. Queries comunes



3. Queries comunes: Sobrevivientes del Titanic

Una query, es una consulta que nosotros le hacemos a cierta base de datos para obtener ciertos datos de manera rápida y sencilla. Para repasar las queries comunes nos centraremos en un ejercicio en el que las utilizaremos para obtener rápidamente información de una tabla.



Para esto consideraremos una tabla con datos personales de las personas que iban en el Titanic ([titanic.csv](#)), y una columna para saber si sobrevivieron. Este dataset se encuentra disponible en Kaggle, en el desafío de los sobrevivientes del Titanic.

Utilizaremos distintas queries, las cuales son comunes en SQL, para obtener tablas más pequeñas que nos entreguen información más específica.



3. Queries comunes: Sobrevivientes del Titanic

1. Para comenzar cree una base de datos la cual se utilizará para los ejemplos del curso.
2. Cree una tabla simple e insertele 2 filas. (**CREATE TABLE** e **INSERT INTO**).
3. Cree una tabla a partir del archivo [titanic.csv](#). Debe tener cuidado en como define las variables de la tabla. (**CREATE TABLE**)
4. Seleccione solo las columnas *passengerid*, *age* y *survived* para generar su primera consulta de tipo **SELECT**. Luego cree una tabla de tipo temporal con esta.
5. Si analiza los datos, la tabla temporal creada tiene valores nulos en la columna edad ¿Cuantos son estos valores? (**COUNT**) Filtre la tabla temporal creada para que muestre solo las personas que tengan un valor mayor a 60 en la columna edad y muestre los primeros 5 datos. (**WHERE** y **LIMIT**)
6. Volviendo a la tabla titanic, seleccion las columnas *passengerid*, *age* y *survived*, pero ahora filtre las personas que fueran mayores a 60 o menores a 10, y que además sobrevivieron. Cree a unatambién en esta misma selección una columna que indique si es un niño o un adulto mayor. (**CASE**, **AND**, **IN** y **OR**)



3. Queries comunes: Sobrevivientes del Titanic

7. Seleccione una nueva tabla con las mismas variables que hemos estado utilizando, pero de las personas con nombre *William* o *Hanna*, y ordenelos por edad. (**Wildcards** y **ORDER BY**)
8. Seleccione una nueva tabla con las mismas variables que hemos estado utilizando, pero ahora agrupe los datos en 3 categorías (Niños, Jovenes, Adultos, Adultos Mayores). En esta tabla cuente las personas correspondientes a cada categoría, calcule para las edades el promedio de dos maneras diferentes, el mínimo, el máximo; además cuente la cantidad de personas que sobrevivio. (**CASE**, **COUNT**, **AVERAGE**, **MIN**, **MAX** and **SUM**)
9. Cree la misma tabla anterior como una tabla temporal, asignandole un nombre a las variables que acaba de crear, ahora escoja solo 1 opción para el promedio de las edades.
10. Exporte esta tabla como un archivo .csv en su escritorio.



4. Funciones para el análisis de datos



DataOwl

4. Funciones para el análisis de datos



5. Introducción a NoSQL



DataOwl

5. Introducción a NoSQL

IV. Structured Query Language

Introduction to Data Science

Data Everywhere

June 12, 2020



DataOwl