Curso Introducción a SQL Espacial sobre PostGIS

Structured Query Language (SQL)

Tipos de datos

formaSIG

La plataforma de aprendizaje en SIG Libre

sigte SERVEI DE SISTEMES D'INFORMACIÓ GEOGRÀFICA I TELEDETECCIÓ Universitat de Girona

UdGFormació FUNDACIÓ UNIVERSITAT DE GIRONA: INNOVACIÓ I FORMACIÓ

Tipos de datos

Los grupos de datos que vamos a tratar en esta lectura son:

1.- Numéricos.

2.- Alfanuméricos.

3.- Booleanos.

4.- Fechas y Horas.

Numéricos

Cuando vayamos a necesitar un tipo de datos numérico lo primero que debemos analizar es si ese valor debe contener valores decimales o no. No es lo mismo almacenar valores de superficie que valores relativos al número de habitantes. Si bien una superficie, requerirá con toda probabilidad valores decimales, el número de habitantes siempre será un valor entero (y positivo). Tenemos pues una primera división de datos numéricos: datos enteros y datos reales (también conocidos como datos de coma flotante).

Los valores enteros se pueden definir matemáticamente como valores numéricos y discretos, con un nivel de precisión basado en la unidad. Algunos ejemplos de valores enteros son 4, 1234, -23 y 0.

Otra característica fundamental que debemos analizar antes de elegir un tipo de datos concreto, es el rango de valores que puede alcanzar ese atributo. No será lo mismo un tipo de datos que se maneje entre 1 y 200, que uno que oscile entre 1 y 123908734 aunque ambos sean de tipo entero. Valores numéricos elevados requieren más recursos y por lo tanto son más lentos en su manejo (operaciones, comparaciones, etc).

Los recursos destinados para almacenar cualquier tipo de datos se miden por bytes (1 byte equivale a 8 bits). Un entero que utilice un solo byte solo permitirá representar valores comprendidos entre 0 y 256 (2 elevado a la octava potencia). Hay que tener presente que cualquier operación matemática que se lleva a cabo en una computadora se lleva a cabo en

base dos (sistema binario) y no en base 10 como estamos habituados los humanos. Para saber más del sistema binario podéis consultar el enlace http://es.wikipedia.org/wiki/Sistema_binario .

Listado de tipos numéricos. Recursos utilizados y rango de representación.

Tipo	Tamaño de almacenamiento	Descripción	Rango
smallint	2 bytes	Entero pequeño	-32768 to +32767
integer	4 bytes	Entero	-2147483648 to +2147483647
bigint	8 bytes	Entero grande	-9223372036854775808 to 9223372036854775807
decimal	variable	Precisión especificada por el usuario	Hasta 131072 dígitos enteros (antes del decimal). Hasta 16383 dígitos decimales
numeric	variable	Precisión especificada por el usuario	Hasta 131072 dígitos enteros (antes del decimal). Hasta 16383 dígitos decimales
real	4 bytes	Precisión variable con posibilidad de decimales	Precisión de hasta 6 dígitos decimales
double precision	8 bytes	Precisión variable con posibilidad de decimales	Precisión de hasta 15 dígitos decimales
serial	4 bytes	Entero autoincrementable	De 1 hasta 2147483647
bigserial	8 bytes	Entero grande	De 1 hasta 9223372036854775807

		autoincrementable	

Los tipos numeric y decimal son especialmente particulares, pues permiten al usuario definir la cantidad de recursos (bytes) que se van a destinar para almacenar un valor.

Numeric y decimal requieren dos parámetros: precisión y escala. Precisión es el número total de dígitos permitidos y escala es el número de dígitos decimales permitidos.

Veamos un ejemplo de uso.

```
CREATE TABLE mis_numeros (
Numero decimal(10,2)
);
```

Algunos valores permitidos para la columna número serían:

1234567890 (10 dígitos en total, 0 decimales).

123456789.2 (10 dígitos en total, 1 decimal)

3456.38 (6 dígitos en total, 2 decimales)

-12389 5 (dígitos en total, 0 decimales)

Pero en ningún caso permitirá almacenar valores tales como:

12345678912 <- 11 dígitos en total, 0 decimales

123456789,12 <- 11 dígitos en total, 2 decimales

Los tipos serial y bigserial permiten crear secuencias numéricas. Estas secuencias permiten definir comportamientos de autoincrementación[1]. De este modo, cada vez que se inserte una nueva fila, el tipo de datos serial (o bigserial) se autoincrementará en X unidades, donde X está definida por la secuencia. El valor por defecto del incremento X es una unidad.

Alfanuméricos

Los tipos de datos alfanuméricos permiten almacenar cadenas de texto. Atributos relativos a nombres, direcciones postales, direcciones electrónicas, etc son claros ejemplos de valores alfanuméricos.

PostgreSQL incorpora los siguientes tipos de datos alfanuméricos.:

Tipo	Descripción
character varying(n)	Permite almacenar valores alfanúmericos hasta 'n' caracteres de longitud máxima. Valores con menos de 'n' caracteres requerirán menos recursos del sistema.
varchar(n)	Equivalente a character varying(n)
character(n)	Permite almacenar valores alfanuméricos con 'n' caracteres de longitud fija. Valores con menos de 'n' caracteres serán rellenados con caracteres en blanco hasta ocupar 'n' caracteres.
char(n)	Equivalente a character(n)
text	Permite almacenar valores alfanuméricos sin límite de longitud.

A efectos del tamaño de la base de datos hay que tener en cuenta que cada carácter de la cadena de texto requiere 1 byte de memoria (8 bits). Como caso especial, la codificación UTF8 (cada vez más extendida) requiere entre 1 y 4 bytes para cada carácter codificado. Podéis leer más sobre UTF8 en http://es.wikipedia.org/wiki/UTF-8 .

Al introducir valores de texto debemos escribirlos empezando y terminando con comilla simple.

Ejemplo:

```
CREATE TABLE mi_texto(
```

```
Cadena varchar(10)
);

INSERT INTO mi_texto(Cadena) VALUES('Este texto no cabe');
INSERT INTO mi_texto(Cadena) VALUES('Este si');
```

Observa que la primera expresión INSERT INTO devuelve un error y no se introduce nada en la tabla, ya que la longitud de la cadena de texto es superior a lo especificado (10 caracteres).

Booleanos

El tipo de datos booleano solo permite dos posibles valores: cierto y falso. Un solo bit es suficiente para almacenar los valores cierto (1) y falso (0), y es por lo tanto el tipo de datos que consume menos recursos.

Este tipo de datos es muy útil para indicar si el registro actual cumple o no cierta condición. Si un cliente está activo o no, si un ciudadano tiene o no la nacionalidad, si un alumno es apto o no, si un cliente a pagado un recibo o no, etc.

Probablemente hayáis observado que existen otras posibilidades para almacenar este tipo de información. Por ejemplo podemos definir un tipo de datos character varying(2) para almacenar 'SÍ' o 'NO', o incluso podemos utilizar el tipo smallint para almacenar '1' (como cierto) y '0' como falso, pero en ambos casos estaremos exigiendo más recursos al sistema. Además no podremos utilizar los operadores booleanos (AND, OR, NOT) y, por si fuera poco, deberemos ejercer un mayor control para asegurar que solo vamos a introducir valores 'SÍ', 'NO', 1 o 0 según corresponda.

Datos de tiempo. Fechas, horas.

PostgreSQL permite varios tipos de datos para almacenar fechas y horas. Los diferentes tipos de fechas de PostgreSQL se relacionan, nuevamente, con la cantidad de recursos destinados por el sistema (bytes). De igual modo a como hemos visto para los tipos numéricos, un mayor número de bytes deriva en una mayor precisión.

Tipo	Tamaño de almacenamiento	Descripción	Rango inf.	Rango sup.	Precisión
timestamp	8 bytes	Incluye fecha y hora. Hasta 14 dígitos.	4713 A.C	5874897 D.C	Microsegundo.
date	4 bytes	Representa solo una fecha, sin hora	4713 A.C	5874897 D.C	1 día.
time	8 bytes	Representa solo una hora, sin fecha. Hasta 14 dígitos.	00:00:00	24:00:00	Microsegundo

Veámos un ejemplo, de más a menos precisión:

Timestamp: '2004-10-19 10:23:54'

date: '2004-10-19'

time: '10:23:54 '

Formatos. Podemos representar la fecha 8 de Enero de 1999 en cualquier de los siguiente formatos.

	January 8, 1999	

	1999-01-08	
	1/8/1999	
	1/18/1999	
	01/02/03	
	1999-Jan-08	
	Jan-08-1999	
	08-Jan-1999	
	99-Jan-08	
	08-Jan-99	
	Jan-08-99	
	19990108	
	990108	
	1999.008	
	J2451187	
	January 8, 99 BC	

Para el tipo 'time' tenemos los siguientes formatos.

	04:05:06.789	
	04:05:06	

	04:05	
	040506	
	04:05 AM	
	04:05 PM	
	04:05:06.789-8	
	04:05:06-08:00	
	04:05-08:00	
	040506-08	
	04:05:06 PST	
	2003-04-12 04:05:06 America/New_York	

PostgreSQL incorpora además algunas palabras reservadas que podemos utilizar para referirnos a ciertas fechas.

Palabra reservada	Tipos de datos válidos	Descripción
epoch	date, timestamp	Equivale a la fecha 1970-01-01 00:00:00+00 . Tiempo cero para los sistemas Unix
infinity	date, timestamp	Posterior a cualquier fecha (date o timestamp)
-infinity	date, timestamp	Anterior a cualquier fecha (date o timestamp)
now	date, time, timestamp	Hora, fecha actual (en el momento de su uso)

today	date, timestamp	Hoy a media noche (24:00:00)
tomorrow	date, timestamp	Mañana a media noche
yesterday	date, timestamp	Ayer a media noche
allballs	time	00:00:00 Todo ceros.

En general, cualquier valor que deba ser almacenado en un campo de tipo temporal debe ir escrito entre comillas simples ('). Por ejemplo, dada una tabla 'test':

```
create table test(
palabra char(10),
ts timestamp,
d date,
t time
);
```

Podemos usar la expresión siguiente para introducir la fecha u hora actuales mediante:

```
insert into test(palabra,ts,d,t) VALUES('now', 'now', 'now', 'now');
```

Puedes comprobar como el campo de tipo timestamp contiene tanto la fecha como la hora actuales, el campo de tipo date contiene sólo la fecha y el campo de tipo time sólo contiene la hora.

palabra character(10)	ts timestamp without time zone	d date	t time without time zone
now	2011-12-09 12:39:08.843	2011-12-09	12:39:08.843

Si ahora pruebas la siguiente expresión:

insert into test (palabra,ts,d) VALUES('today', 'today', 'today');

Comprobarás que el valor almacenado en el campo de tipo date es el mismo tanto si usamos now como si usamos today. Por lo contrario, el campo de tipo timestamp tiene la misma fecha en ambos registros pero el último (en el que usamos 'today') contiene las 0:00 como hora.

palabra character(10)	ts timestamp without time zone	d date	t time without time zone
now	2011-12-09 12:39:08.843	2011-12-09	12:39:08.843
today	2011-12-09 00:00:00	2011-12-09	

PostgreSQL también incorpora otros tipos de fechas que, además de la información que aparece en la tabla anterior, incluyen el huso horario. Para más información podéis consultar http://www.postgresql.org/docs/9.1/static/datatype-datetime.html

Edita: Servicio de SIG y Teledetección (SIGTE) de la Universitat de Girona

Año: 2013

Contenidos elaborados por: Toni Hernández Vallès

sigte
SERVEI DE SISTEMES D'INFORMACIÓ GEOGRÀFICA I TELEDETECCIÓ
Universitat de Girona

UdGFormació
FUNDACIÓ UNIVERSITAT DE GIRONA: INNOVACIÓ I FORMACIÓ

www.sigte.udg.edu/formasig

formasig@sigte.org