Introduction to PostgreSQL data types

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

datacamp

Data categories in PostgreSQL

- Text
- Numeric
- Temporal
- Boolean
- Others: Geometric, Binary, Monetary

Example 1: representing birthdays

- Cathy: May 3rd, 2006
- Possible representations
  - "May 3, 2006" (text)
  - "5/3/2006" (text)
  - 2006-05-03 (date)

Example 2: tracking payment status

- Did attending member pay?
- Possible representations:
  - "Yes"/"No" (text)
  - "Y"/"N" (text)
  - 'true'/'false' (boolean)
- Specific types provide restriction on values

Example 3: trip distances

- Mark flew 326 miles for client meeting
- Possible representations:
  - "326 miles" (text)
  - "326" (text)
  - 326 (numeric)

Let's practice!

CREATING POSTGRESQL DATABASES

Defining text columns

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

Using text in PostgreSQL

```
CREATE TABLE book (
    isbn CHAR(13) NOT NULL,
    author_first_name VARCHAR(50) NOT NULL,
    author_last_name VARCHAR(50) NOT NULL,
    content TEXT NOT NULL
);
```

Text data types: `TEXT`, `VARCHAR(N)`, `CHAR(N)`

The TEXT data type

- Strings of variable length
- Strings of unlimited length
- Good for text-based values of unknown length

The VARCHAR data type

- Strings of variable length
- Strings of unlimited length
- Restriction can be imposed on column values
  - `VARCHAR(N)`
  - *N* - maximum number of characters stored
  - Column can store strings with less than *N* characters
  - Inserting string longer than *N* is error
- `VARCHAR` without *N* specified equivalent to `TEXT`

```
first_name VARCHAR(50) NOT NULL;
```

The CHAR data type

- `CHAR(N)` values consist of exactly *N* characters
- Strings are right-padded with spaces
- `CHAR` equivalent to `CHAR(1)`

```
isbn CHAR(13) NOT NULL;
```

Let's practice!

CREATING POSTGRESQL DATABASES

Defining numeric data columns

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

datacamp

Numeric data with discrete values

```
CREATE TABLE people.employee {
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(10) NOT NULL,
    last_name VARCHAR(10) NOT NULL
}
```

Numeric data with discrete values

```
CREATE TABLE people.employee {
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(10) NOT NULL,
    last_name VARCHAR(10) NOT NULL,
    num_sales INTEGER
}
```

Integer types

Type	Description	Range
`SMALLINT`	small-range integer	-32768 to +32767

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Integer types

Type	Description	Range
`SMALLINT`	small-range integer	-32768 to +32767
`INTEGER`	typical choice for integer	-2147483648 to +2147483647

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Integer types

Type	Description	Range
`SMALLINT`	small-range integer	-32768 to +32767
`INTEGER`	typical choice for integer	-2147483648 to +2147483647
`BIGINT`	large-range integer	-9223372036854775808 to 9223372036854775807

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Integer types

Type	Description	Range
`SMALLINT`	small-range integer	-32768 to +32767
`INTEGER`	typical choice for integer	-2147483648 to +2147483647
`BIGINT`	large-range integer	-9223372036854775808 to 9223372036854775807
`SERIAL`	autoincrementing integer	1 to 2147483647

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Integer types

Type	Description	Range
`SERIAL`	autoincrementing integer	1 to 2147483647
`BIGSERIAL`	large autoincrementing integer	1 to 9223372036854775807

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Numeric data with continuous values

```
CREATE TABLE people.employee {
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(10) NOT NULL,
    last_name VARCHAR(10) NOT NULL,
    num_sales INTEGER
}
```

Numeric data with continuous values

```
CREATE TABLE people.employee {
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(10) NOT NULL,
    last_name VARCHAR(10) NOT NULL,
    num_sales INTEGER,
    salary DECIMAL(8,2) NOT NULL
}
```

`DECIMAL (precision, scale)`

Floating-point types

Type	Description	Range
`DECIMAL` or `NUMERIC`	user-specified precision	131072 digits before the decimal point;16383 digits after the decimal point

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Floating-point types

Type	Description	Range
`DECIMAL` (`NUMERIC`)	user-specified precision	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
`REAL`	variable-precision	6 decimal digits precision

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Floating-point types

Type	Description	Range
`DECIMAL` (`NUMERIC`)	user-specified precision	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
`REAL`	variable-precision	6 decimal digits precision
`DOUBLE PRECISION`	variable precision	15 decimal digits precision

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Let's practice!

CREATING POSTGRESQL DATABASES

Defining boolean and temporal data columns

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

datacamp

Boolean and temporal data

```
CREATE TABLE book (
    isbn CHAR(13) NOT NULL,
    author_first_name VARCHAR(50) NOT NULL,
    author_last_name VARCHAR(50) NOT NULL,
    content TEXT NOT NULL
);
```

Boolean and temporal data

```
CREATE TABLE book (
    isbn CHAR(13) NOT NULL,
    author_first_name VARCHAR(50) NOT NULL,
    author_last_name VARCHAR(50) NOT NULL,
    content TEXT NOT NULL,
    originally_published DATE NOT NULL,
    out_of_print BOOLEAN DEFAULT FALSE
);
```

The BOOLEAN data type

- Three possible values
  - `true` state
  - `false` state
  - `NULL` (unknown state)
- Common for representing yes-or-no scenarios
- Can be defined with keyword `BOOL` or `BOOLEAN`

```
in_stock BOOL DEFAULT TRUE;
```

Temporal data types

Type	Descriptions	Format
`TIMESTAMP`	represents a date and time	2010-09-21 15:47:16
`DATE`	represents a date	1972-07-08
`TIME`	represents a time	05:30:00

Let's practice!

CREATING POSTGRESQL DATABASES