Data types

CSC 256, SQL Programming

Data Types

- A type is a set of values
- Types give us additional constraints
- Each type supports different operations

Main ANSI SQL Data Types

- Character data types
  - CHARACTER
  - CHARACTER VARYING (or VARCHAR)
  - CHARACTER LARGE OBJECT
  - NCHAR
  - NCHAR VARYING
- Binary data types (for example, storing images)
  - BINARY
  - BINARY VARYING
  - BINARY LARGE OBJECT

Main ANSI SQL Data Types (continued)

- Number data types
  - NUMERIC
  - DECIMAL
  - SMALLINT
  - BIGINT
  - FLOAT
  - REAL
  - DOUBLE PRECISION
- Boolean data type (recall three-valued logic)
  - BOOLEAN
- Date/time data types
  - DATE
  - TIME
  - TIMESTAMP
  - INTERVAL

Note About Data Types

- Data types are the most common part of the ANSI standard that vendors do not adhere to.
- The general data types listed above are typically available
- These lecture notes will be about common Postgres data types (character, number, and date)
- Note: when designing a database we have the opportunity to choose types, for example, choosing between int types can save a lot of space if millions of rows

Character Data Types

- Types (`n` specifies max limit)
  - `char(n)`: pads unused space
  - `varchar(n)`: stores the exact characters without padding
  - `text`: near unlimited text
- Note: in PostgeSQL, `varchar` is implemented as `text` with a constraint

Locale

- Locale refers to preferences regarding alphabets, sorting, number formatting, etc.
- Different vendors handle locale differently
- PostgeSQL finds the locale from the system when installing

String Concatenation

- There are two main ways to do string concatenation: the `||` operator and the `concat` function.
- Example:

```
select 'a' || 'b';
select concat('a', 'b');
```

- Note: concatenating with a null value makes the whole result null

```
select 'a' || 'b' || null;
select concat('a', 'b', null);
```

The COALESCE Function

- The COALESCE function finds the first non-null value in a set of values.
- Example

```
select 'a' || 'b' || coalesce(null, 'unkown');
```

- This is sometimes useful to provide default values for null values depending on the operation.

Implicit Type Coercion

- The string concatenation operator will attempt to convert non-string values to string values.
- Example

```
select 'a' || 1 || 4.2 as string;
 string
--------
 a14.2
(1 row)
```

- This is called implicit type coercion.
- This happens with some other operations as well.

Integer Data Types

- Integers are numbers without a fractional component
  - `smallint`: 2bytes
  - `integer`: 4bytes
  - `bigint`: 8bytes
- Sometimes the integer types are aliased as `int2`, `int4`, `int8`
- During database design, you typically want to choose an integer type that is small but also can hold enough values.

Integer Operations

- Integers support the standard arithmetic operators.
  - addition (+)
  - subtraction (-)
  - multiplication (*)
  - integer division (/)
  - modulo (%)
- Integer division example:

```
select 5 / 2, 5 % 2;
 ?column? | ?column?
----------+----------
        2 |        1
(1 row)
```

Checking Types (PostgreSQL)

- PostgreSQL has function `pg_typeof` to get the data type of any value.

```
select
  pg_typeof(3),
  pg_typeof(2.14),
  pg_typeof(date '2022-01-01');
 pg_typeof | pg_typeof | pg_typeof
-----------+-----------+-----------
 integer   | numeric   | date
(1 row)
```

Type Coercion

- Recall that sometimes one type will be converted into another type depending on the operation.
- Example:

```
 select 3.0 / 2;
      ?column?
--------------------
 1.5000000000000000
(1 row)
```

  - Here the integer value 2 was coerced into a numeric type (type promotion)

Explicit Type Coercion

- PostgreSQL has three ways to explicitly coerce a value

```
select
  int '33',
  '33'::int,
  cast('33' as int)
```

- The third form is the ANSI SQL standard form

Fractional Number Types

- Exact fractional types – fixed point
  - `numeric(precision, scale)`
  - `scale` is the number of digits after the decimal point
  - `precision` is the total number of digits
  - Example: `numeric(5, 2)` has the values `000.00` to `999.99`
- Approximate fractional types – floating point
  - `real` and `double precision`
  - floating point types can save space but, there are various issues, for example round-off errors.

Exact versus Approximate Example

```
select
0.4235::numeric(5,4) * 10000000,
0.4235::real * 10000000;
 ?column?   |     ?column?
--------------+------------------
4235000.0000 | 4235000.014305115
(1 row)
```

Date and Time Types

- Main data types
  - `date`
  - `time`
  - `timestamp`: combination of date and time (and timezone: timestamp with zone)
  - `interval`
- In general, working with dates and times are tricky (especially with respect to time zones)

Date Type

- A date literal should use the ISO standard form: 'YYYY-MM-DD'

```
select date '2022-01-01';
```

- The date type does validation

```
select date '2022-02-30';
ERROR:  date/time field value out of range: "2022-02-30
LINE 1: select date '2022-02-30';
```

Date Arithmetic

- Date types support arithmetic operators
- Example: subtracting dates returns an integer representing the number of days

```
select date '2022-01-01' - date '2021-01-01';
 ?column?
----------
      365
(1 row)
```

- Note the we need to be careful of the data types of the operands on each side of the operation as well as the data type returned; check the documentation for a full list of supported operations.

Time Type

- A time literal should use the ISO standard form: 'HH:MM:SS'

```
select time '12:30:22';
```

- The `time` type does validation

```
select time '12:30:99';
ERROR:  date/time field value out of range: "12:30:99"
LINE 1: select time '12:30:99';
```

Timestamp Type

- The `date` and `time` types are not commonly used in practice; instead the `timestamp` is typically used
- The `timestamp` type combine the `date` and `time` into a single value
- Example:

```
select
  timestamp '2022-01-01 12:00 America/New_York';
  timestamp '2022-01-01 12:00 +5';
  timestamp '2022-01-01 12:00 EST';
```

- Note: the output is shown relative to local time zone.

Interval Type

- The `interval` type represents a span of time or a duration.
- Example: differences in timestamps result in an interval:

```
select timestamp '2022-01-01 12:00'
       - timestamp '2022-01-01 10:00';
 ?column?
----------
 02:00:00
(1 row)
```

Interval Type (continued)

- Example: add an `interval` to a `timestamp`

```
select timestamp '2022-01-01 12:00' + interval '1 Day';
      ?column?
---------------------
 2022-01-02 12:00:00
(1 row)
```

- `interval` values can be specified with "natural language"; check the documentation for interval phrases.

Additional PostgreSQL Data Types

- PostgeSQL supports various other data types:
  - monetary
  - enumerated
  - geometric
  - network address
  - bit string
  - text search
  - UUID
  - XML
  - JSON
  - arrays
  - composite
  - range
- Check the documentation for details.