Date and Time

CIP 2015-08-06

What has changed since the last call?
More consistent treatment of epoch conversion

Converting from a numerical epoch timestamp to a DateTime value:

WITH timestamp() AS epochTS
RETURN datetime({epochMillis: epochTS}) AS now

WITH secondsSinceEpoch, nanos
RETURN datetime({epochSeconds: secondsSinceEpoch, nanosecond: nanos})

Converting a DateTime value to a numerical epoch timestamp:

WITH datetime() AS now
RETURN now.epochMillis AS timestamp,
     now.epochSeconds AS secondsSinceEpoch, now.nanosecond AS nanos
Comparison of values with same time and different timezone

Given two temporal instant values, \( a \) and \( b \):

- If \( a \) and \( b \) are of different types,
  they are *incomparable* (\( a < b, a \leq b, a \geq b, \) and \( a > b \) yield null) and *not equal* (\( a = b \) yields false and \( a \neq b \) yields true)
- If \( a \) and \( b \) have the same type, but represent different points in time,
  \( a < b \) is true iff \( a \) represents a point in time before \( b \), implying that \( b > a \) is true
- If \( a \) and \( b \) have the same type, represent the same point in time, but have a different timezone offset,
  \( a < b \) is true iff \( a \) has a timezone west of \( b \), i.e. iff the local time of \( a \) is “before” the local time of \( b \) (*this is what has been changed - they used to be treated as equal*)
- If \( a \) and \( b \) have the same type, represent the same point in time, have the same timezone offset,
  but a different timezone identifiers,
  \( a < b \) is true iff \( a \) has a timezone identifier that is *alphabetically before* the timezone identifier of \( b \) (*this has been added*)
- If \( a \) and \( b \) have the same type, represent the same point in time, and have the same timezone,
  then *they are equal*
Not changed, but more details on: Timezones

DateTime and Time support timezones and offsets
(LocalDateTime and LocalTime are the same types without timezone support)

Timezone names use the IANA (Olson) TZ database.

Time only supports offsets, since a date is required to convert a zone id to an offset. However timezone id is allowed as input, and uses the current date to compute the offset.

Timezone data can be input and read through these fields:

- **timezone** - the name of the timezone (tzid), or the same as offset if no named zone is used. For Time this always returns offset, for input it can be either an offset string or a timezone id.
- **offset** - the offset from UTC of the instant, as a formatted string, i.e. "+02:00"\(^1\) in output this is always formatted to contain hours and minutes, but will contain seconds if the offset is not an even number of minutes (i.e. "+00:50:48"\(^2\)). For input it is ok to leave off the minutes - that is interpreted as them being zero.
- **offsetMinutes** - the total offset in minutes (as an integer), i.e. 120\(^1\)
- **offsetSeconds** - the total offset in seconds (as an integer), i.e. 3048\(^2\)

\(^1\) The current timezone offset of Sweden (CEST)
\(^2\) The LMT offset of the Neo4j Malmö office