TEMPORAL JOURNEY

Bank-Builder

GRINDROD BANK

PROJECT BIG BAOBAB

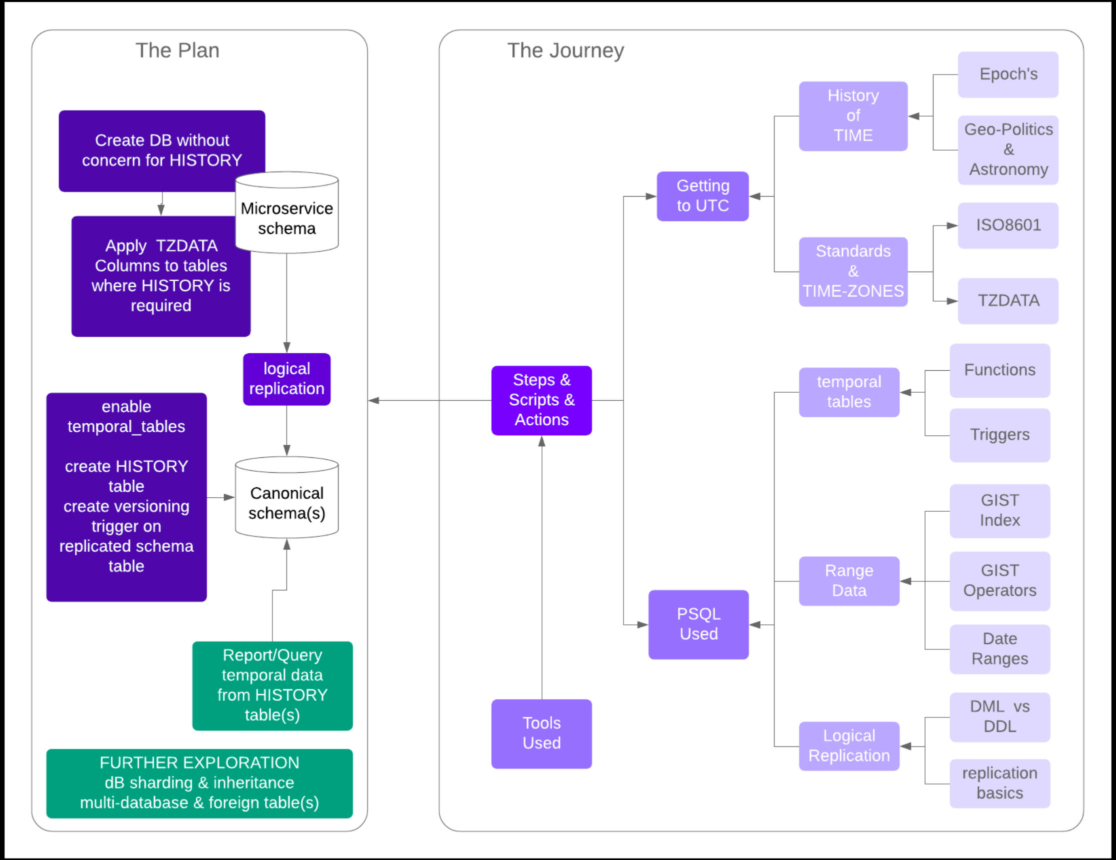
2019 https://github.com/Bank-Builder/temporal_journey/

Talk in Two Parts

- Getting to UTC
- PSQL Used

<@

- Final Word



PART I:

Getting to UTC

$$\Delta t' = \frac{\Delta t}{\sqrt{1 - \frac{v^2}{c^2}}},$$

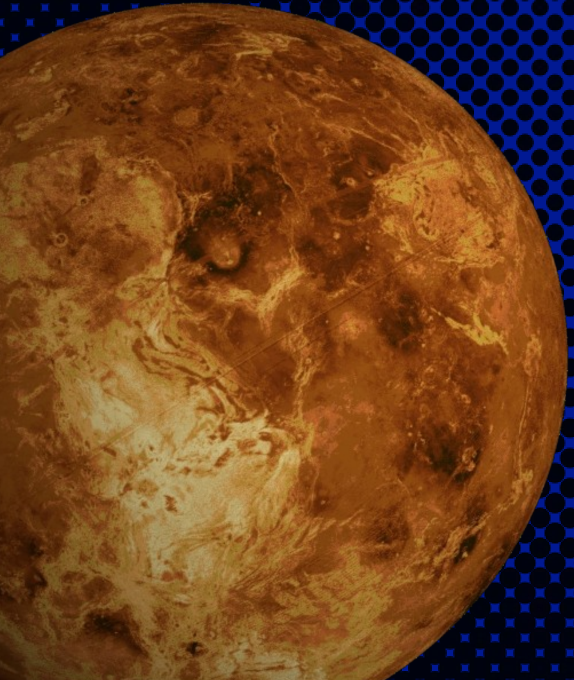
This talk flies by because it is so interesting to listen to, but the speed of rotation of all the astronomical bodies including the earth has apparently not changed.

For that bored person in the 3rd row however,
the world seems to have slowed, painfully, right down!

time zones - which exist because of planetary rotation wrt an external solar reference, and taken together with geo-location or latitude on our familiar ball, we then apply convention for convenience and throw in politics & habit so we can add inconvenient consequences such as daylight savings & international date lines etc.

astronomical units - such as rotations around an axis or rotations around the sun, days, seasons, years, heavenly orbits & epochs

Venus has a day that is about half a year long.



i.e. it rotates around the Sun in 224.65 Earth-days and around its own axis in 116.75 Earth days.
Put another way, it rotates around it's own axis in 1 Venus-day and around the Sun (a Venus year) in 1.92 Venus-days.

STANDARD TIME ZONES OF THE WORLD
-11
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1
0
+1
+2
+3
ARCTIC OCEAN
Greenland
(DENMARK)
U.S.
CANADA
UNITED STATES
MEXICO
NORTH PACIFIC OCEAN
NORTH ATLANTIC OCEAN
ICELAND
NORWAY
SWEDEN
FINLAND
UNITED KINGDOM
IRELAND
FRANCE
GERMANY
POLAND
UKRAINE
SPAIN
ITALY
KAZAKHSTAN
ALGERIA
LIBYA
EGYPT
MOROCCO
MAURITANIA
MALI
NIGER
CHAD
SUDAN
NIGERIA
VENEZUELA
COLOMBIA

An epoch is a significant time marker.

Lisp used 0-time to be 1900-01-01 00:00:00 UTC and Unix uses 1970-01-01 00:00:00 UTC,
and simply keep a continuous count of seconds
(as does postgreSQL).

Go, Python and .NET utilise Rata Die (R.D.)
<from Latin for "fixed-date">

The civil calender used by most countries is the Gregorian solar calendar, named after Pope Gregory XIII, who introduced it in October 1582, which has 365.2425 days in a year and caters for leap years.

This is a very close approximation for the actual 365.2422 days in a tropical year that is currently determined by the Earth's actual revolution around the Sun.

Leap Years

Every year that is exactly divisible by four is a leap year,
except for years that are exactly divisible by 100 - which are not,
except for when these centennial years are exactly divisible by 400 - then they are.

For example, the years 1700, 1800, and 1900 were not leap years,
but the year 2000 was and the 2100 wont be.

```
RD = JD − 1 721 424.5
//convert Gregorian date (Y/M/D) to JD Number:
JDN = (1461 × (Y + 4800 + (M − 14)/12))/4 +\
(367 × (M − 2 − 12 × ((M − 14)/12)))/12 − \
(3 × ((Y + 4900 + (M - 14)/12)/100))/4 + D − 32075

//US Day of week (sun:0- sat:6) :
w1 = mod(jdn + 1 ,7)

//ISO Day of Week (mon:1 - sun:7) :
w0 = mod(jdn, 7) + 1
```

SO..MAY YOU NEVER
HAVE TO
DO THIS IN YOUR CODE
...

However, people tend to prefer words to numbers, and go out of their way to name things such as:

- months &
- days of week &
- times of day etc.

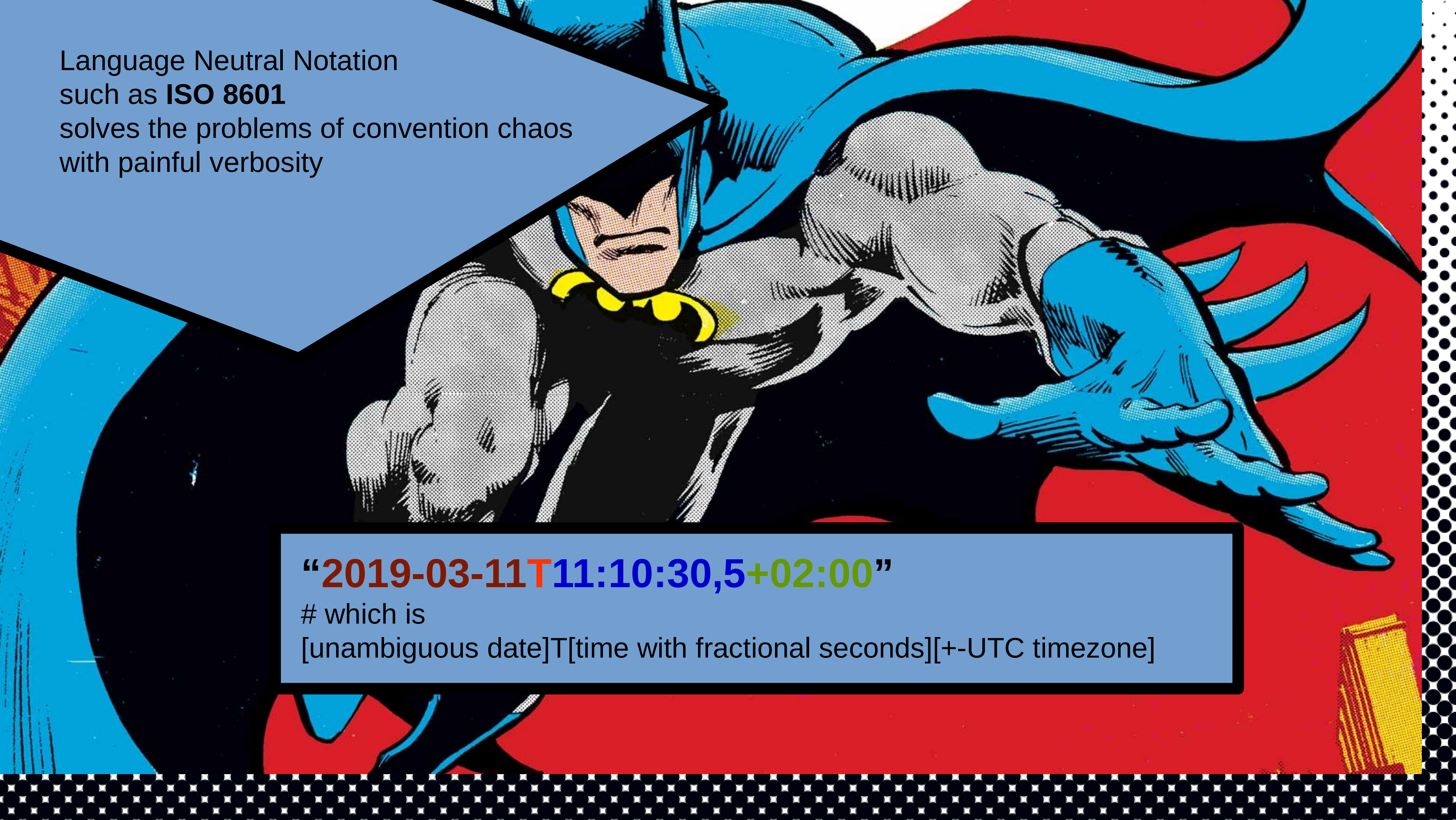
This also creates convention chaos e.g. ***mm/dd/yy*** versus ***dd/mm/CCyy***

or worse just ***mm/dd*** which

```
ls -l
```

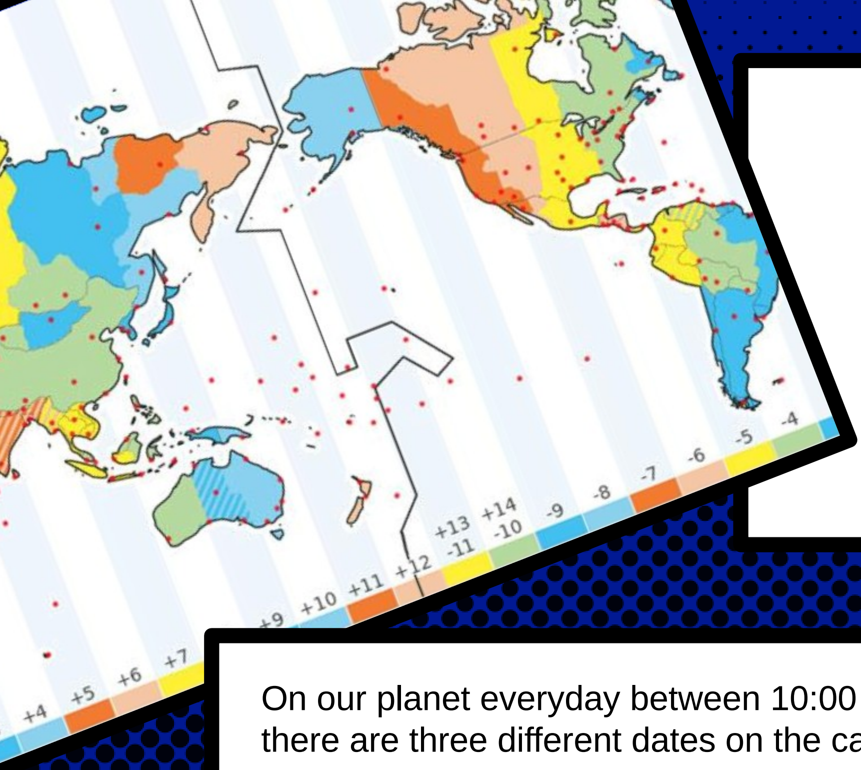
uses by default, so you probably want to do something like:

```
ls -l --time-style=+%C%y/%m/%d
```



Language Neutral Notation
such as **ISO 8601**
solves the problems of convention chaos
with painful verbosity

“2019-03-11T11:10:30,5+02:00”
# which is
[unambiguous date]T[time with fractional seconds][+-UTC timezone]

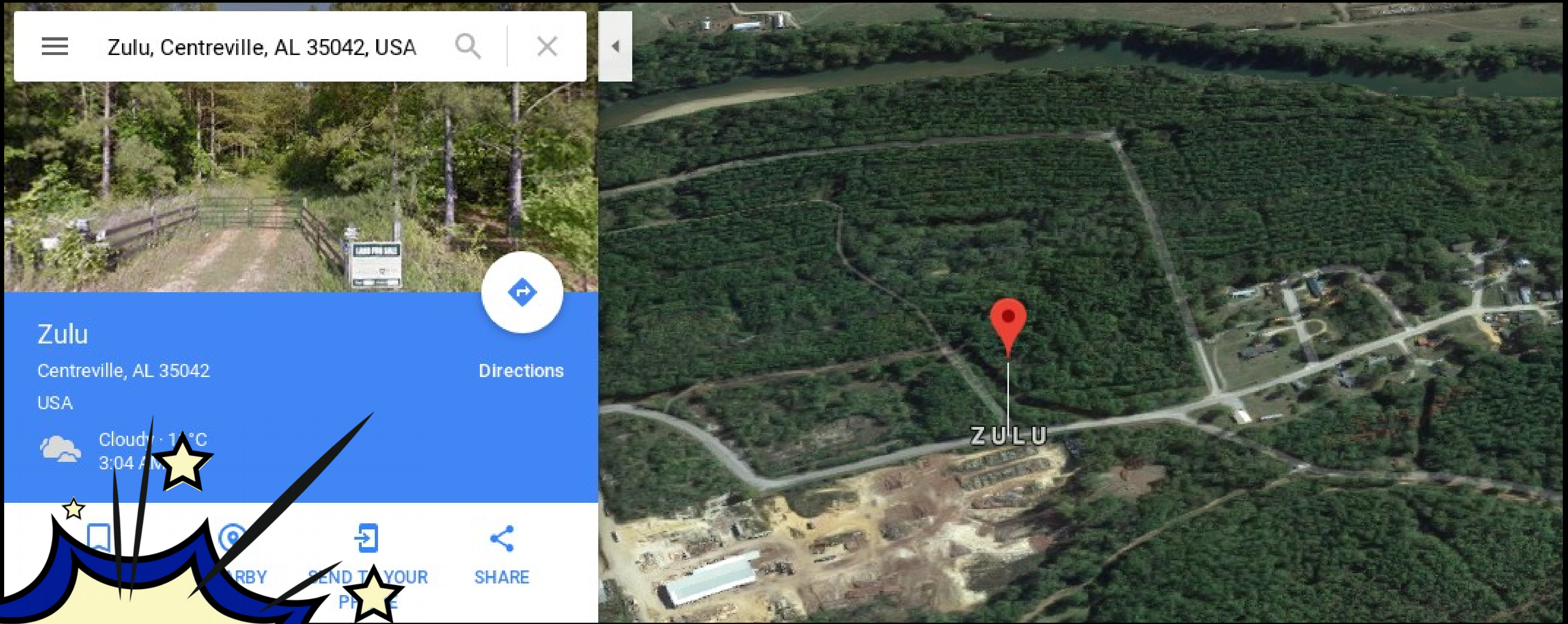


International Date Line (IDL)

at 180 deg (-24 hrs west to east), and for our pleasure the time-zone zigzags to take into account countries and geopolitical regions - so it is possible to cross the IDL from Baker Island to Tokelau (just 1061km) and have to add 25 hrs forward = 1 day + 1 hr because of time zigzags across time-zones.

On our planet everyday between 10:00 and 11:59 UTC there are three different dates on the calender in use.

10:30 UTC it is 2nd May
23:30 UTC-11 (American Samoa) it is May 1st
06:30 UTC-4 (New York) it is May 2nd, and
00:30 UTC+14 it is May 3rd (Kiritimati is always the first country to celebrate New Year)


Zulu, Centreville, AL 35042, USA

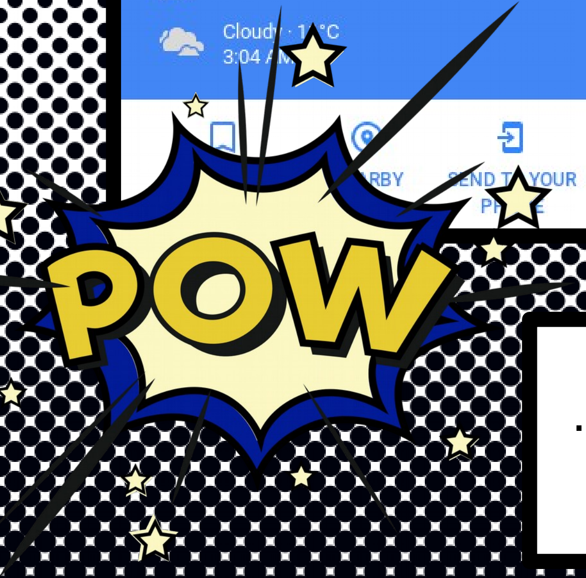
Zulu

Centreville, AL 35042

USA

Cloudy · 1 °C

3:04 AM



...whatever time zone you use, every day there is a time when you're only a second away from tomorrow...

timedatectl

```
      Local time: Mon 2019-01-07 18:42:55 SAST
  Universal time: Mon 2019-01-07 16:42:55 UTC
        RTC time: Mon 2019-01-07 16:42:55
       Time zone: Africa/Johannesburg (SAST, +0200)

       System clock synchronized: yes
systemd-timesyncd.service active: yes
                 RTC in local TZ: no
```

systemd

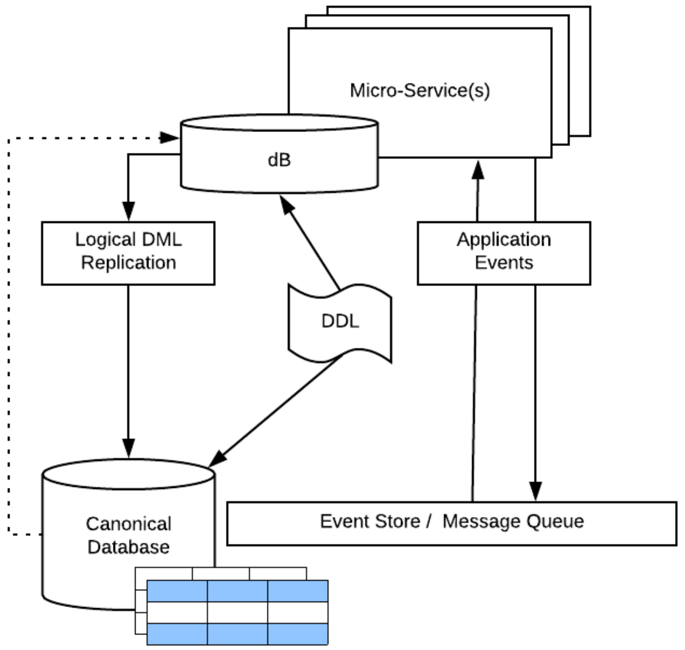
```
timedatectl list-timezones
sudo timedatectl set-timezone Africa/Johannesburg
```

It seems that we simply cannot get away from time
or
is it that time keeps getting away from us?

PART II:

PSQL Used

MICRO-SERVICES & LOGICAL DB REPLICATION



TEMPORAL TABLES

- git clone git@github.com:GrindrodBank/temporal_tables.git
- add versioning_function.sql to db
- CREATE TABLE <my_table>
- ALTER <my_table> to add “sys_period” column
- CREATE TABLE <my_table>_history (LIKE <my_table>);
- add the versioning_trigger using the versioning_function

Now go ahead and insert, update
& delete and see what happens

LOGICAL REPLICATION

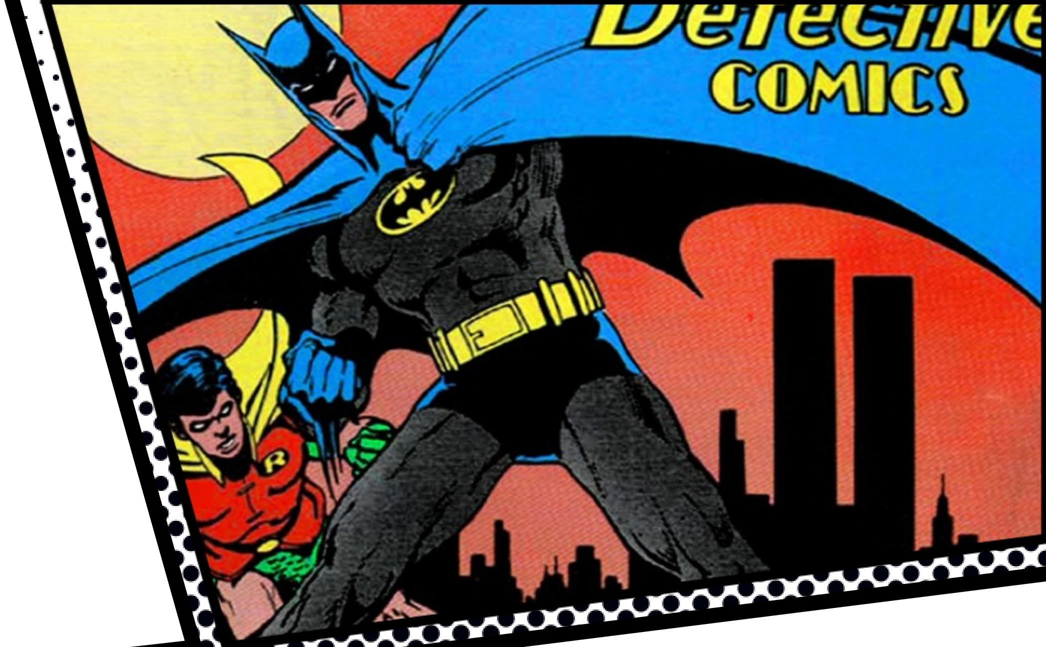
- SET wal_level=logical
- CREATE ROLE for REPLICATION
- GRANT read access to the replication user
- CREATE PUBLICATION FOR <my_table>
- Do this for each micro-service schema for all tables

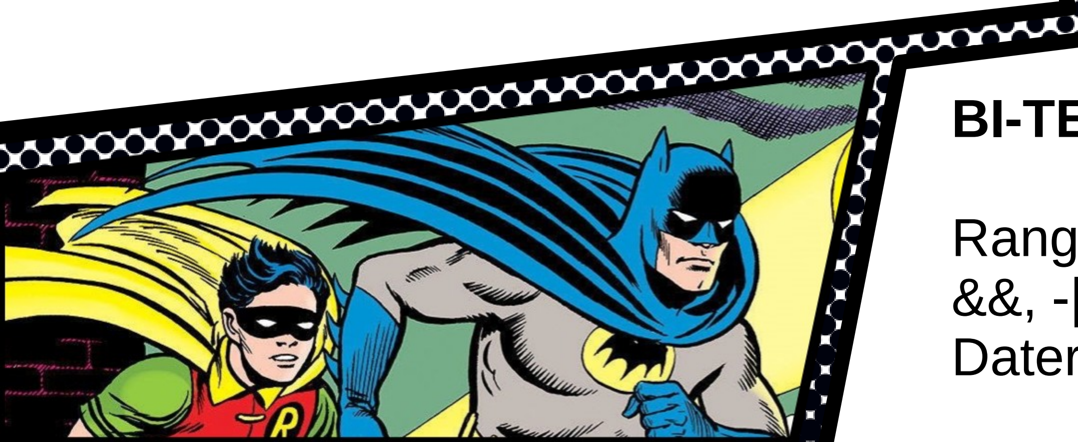
On the **canonical database** :
CREATE SUBSCRIPTION(s) with CONNECTION(s) to PUBLICATION(s)

Explicitly ENABLE ALWAYS TRIGGER versioning_trigger for logical replication to trigger temporal table events, even if they will trigger without this for local DML events.

BI-TEMPORAL DATA

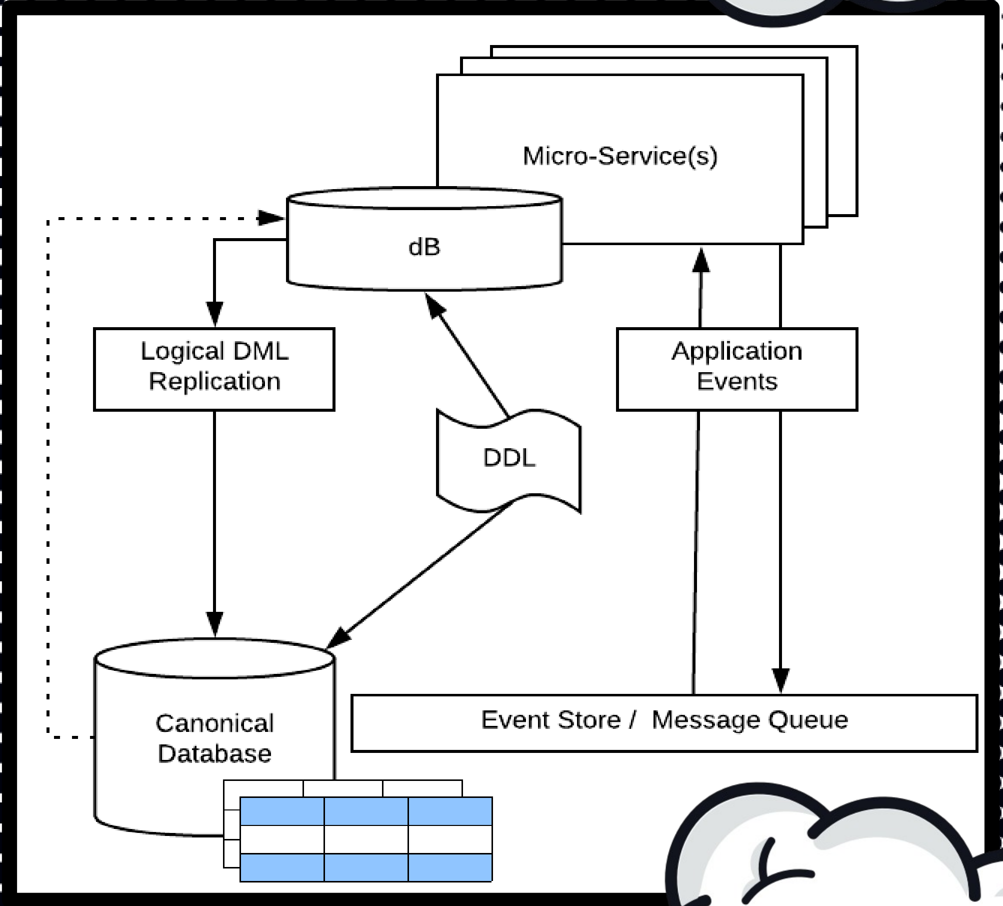
- Validity vs Audit
- ADD COLUMN of type daterange
- ADD && CONSTRAINT
- POPULATE validity_period

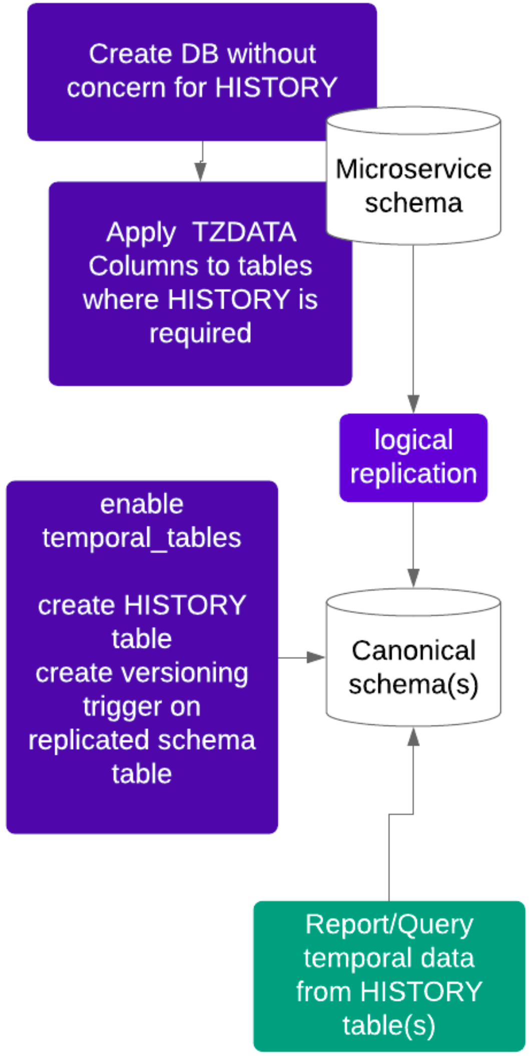


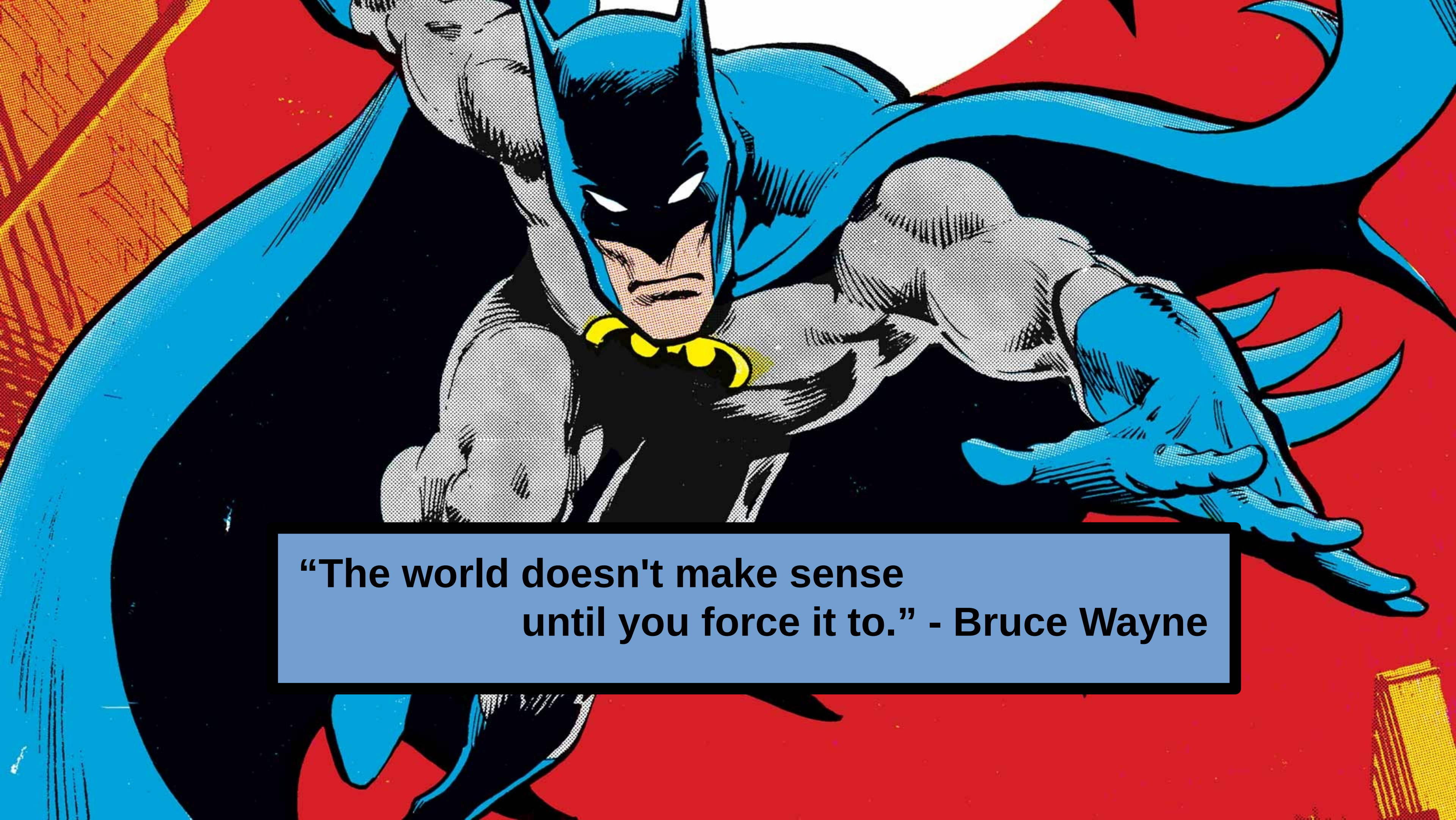


BI-TEMPORAL QUERIES

Range queries
&&, -|-, >>, &<
Daterange ‘[)’ vs ‘[]’


Micro-Service(s)
dB
Logical DML Replication
DDL
Application Events
Canonical Database
Event Store / Message Queue


Create DB without concern for HISTORY
Microservice schema
Apply TZDATA Columns to tables where HISTORY is required
logical replication
enable temporal_tables
create HISTORY table
create versioning trigger on replicated schema table
Canonical schema(s)
Report/Query temporal data from HISTORY table(s)



“The world doesn't make sense until you force it to.” - Bruce Wayne