Is it wise to differentiate NaN and NULL?

In DBMS:

- *NULL* is defined to specify unknown (missing) value
- *NULL* properties:
  - count as 0:
  - select count(null) ==> 0
- Postgres accepts NaN and Inf as floating-point values, but...
  (see next slides)

In computers:

- NaN (not a number) , +Inf, -Inf have well-defined properties:
  - 0/0 = NaN
  - 1/0 = +Inf, -1/0=-Inf
  - Test equality (x==x) is *false* when x is NaN
- In computations:
  - log(0) = -Inf
  - log(-1) = NaN
  - etc...

Comparison C vs DBMS: the simplest table

DBMS (Postgres)

```
select * from t
     v
-----------
         0
         1
        -1
  Infinity
 -Infinity
       NaN
      null
-----------
```

From C

```
/* Vector: */
double v[6] = {
  0,
  1,
 -1,
 1./0., /* +Inf */
-1./0., /* -Inf */
 0./0.  /* NaN  */
};
```

Comparison C vs DBMS: equality test

DBMS (Postgres)

```
select a.v,b.v
from t a, t b
where a.v=b.v ;
```

```
      v     |     v
------------+-----------
 -Infinity  | -Infinity
         -1 |         -1
          0 |          0
          1 |          1
  Infinity  |  Infinity
        NaN |         NaN
```

From C

```
for(i=0;i<6;i++)
for(j=0,j<6;j++)
  if(v[i]==v[j])
  printf(v[i],v[j]);
```

```
  0     0
  1     1
 -1    -1
 inf   inf
-inf  -inf
```

logical comparison not valid in PostgreSQL
(NaN is NOT equal to NaN)

Comparison C vs DBMS: inverse value

DBMS (Postgres)

```
select 1/v from t
     v
---------------------
ERROR:  division by
  zero
```

From C

```
for(i=0;i<6; i++)
  printf(v,1/v[i]);

  0   inf
  1    1
 -1   -1
 inf   0
-inf  -0
 nan  nan
```

Arithmetic computation not valid in PostgreSQL

Conclusion

- Computations using *NaNs* in DBMS do not behave as expected
- ... but *NULL* in DBMS behaves like *NaN* in computations and logical comparisons

Therefore use null as the DBMS equivalent of NaN which represents the VOTable and FITS NULL numbers