Exprimer le néant dans divers langages de programmation

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 7 mai 2008. Dernière mise à jour le 12 mai 2008

http://www.bortzmeyer.org/exprimer-neant.html

Tous (ou presque tous ?) les langages de programmation permettent de noter une donnée qui n'existe pas, le rien, le néant. Mais chacun utilise un symbole différent.

En SQL, c'est le `NULL`. Ainsi, un booléen en SQL est tri-valué : vrai, faux ou `NULL`. Cela agace prodigieusement certains théoriciens comme Date <http://www.bortzmeyer.org/sql-standard.html>. Voici un exemple avec une valeur égale à `NULL` :

```
tests=> CREATE TABLE Data (Name TEXT,  Ready BOOLEAN);
CREATE TABLE
tests=> INSERT INTO DATA Values ('toto', false);
INSERT 0 1
tests=> INSERT INTO DATA Values ('tata', true);
INSERT 0 1
tests=> INSERT INTO DATA (Name) Values ('truc');
INSERT 0 1

tests=> SELECT * FROM Data;
 name | ready
------+-------
 toto | f
 tata | t
 truc |
(3 rows)

tests=> SELECT * FROM Data WHERE Ready IS NULL;
 name | ready
------+-------
 truc |
(1 row)
```

Et un exemple illustrant bien que `NULL` n'est ni vrai ni faux. La troisième ligne, avec le nom `truc`, n'apparaitra pas :

```
tests=> SELECT * FROM Data WHERE Ready;
 name | ready
------+-------
 tata | t
(1 row)

tests=> SELECT * FROM Data WHERE NOT Ready;
 name | ready
------+-------
 toto | f
(1 row)
```

PHP utilise également `NULL`.

Python utilise `None`. N'importe quelle variable peut prendre la valeur `None`. Par contre, Python est typé, même si c'est dynamiquement typé, et on ne peut pas effectuer n'importe quelle opération sur `None` :

```
>>> 2 + None
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: unsupported operand type(s) for +: 'int' and 'NoneType'
```

Si on essaie avec des booléns, `None` a un comportement qui peut être déroutant <`http://docs.python.org/ref/Booleans.html`> : il est quelque part entre `True` et `False`.

```
>>> x = True and False
>>> print x
False
>>> x = True and None
>>> print x
None
>>> x = False and None
>>> print x
False
>>> x = False or None
>>> print x
None
```

Perl utilise `undef`. N'importe quelle variable peut prendre cette valeur. Contrairement à Python, toutes les combinaisons sont possibles avec `undef` :

```
 % perl
print 2 + undef, "\n";
2

% perl
print 2  and undef, "\n";
2
```

Lisp utilise traditionnellement `nil`, qui sert aussi à noter le booléen Faux (merci à Emmanuel Saint-James pour son aide sur Lisp). `#f` qui sert à noter « faux » et uniquement « faux » a été introduit par Scheme, qui a supprimé la confusion qui existait dans Lisp <http://www.cs.yale.edu/homes/dvm/nil.html> sur ce sujet. La liste vide `()` convient aussi. Voici un exemple en Common Lisp :

```
; Renvoie Faux donc nil
> (or (= 2 3) (= 2 1))
NIL

> (or (= 2 3) (= 2 2))
T

; La liste vide est fausse
> ()
NIL

; On peut l'utiliser avec les opérateurs booléens
> (or (= 2 3) ())
NIL

> (or (= 2 2) ())
T
```

Pascal désigne les pointeurs invalides par `nil`. Pascal est statiquement et assez fortement typé et les variables des autres types n'ont pas de valeur `nil` possible. Ada est très proche de Pascal sur ce point, mais utilise `null` au lieu de `nil`.

Ruby utilise également `nil`.

Haskell est typé et ne permet pas de valeur « nulle » sauf si on utilise la monade Maybe <http://www.haskell.org/ghc/docs/latest/html/libraries/base/Data-Maybe.html> qui permet d'avoir une valeur `Nothing`. Peu de fonctions sont définies sur les variables `Maybe` (à part tester si elles contiennent quelque chose et, si oui, quoi).

Visual Basic a le même terme, `Nothing`.

C utilisait traditionnellement `0`, valable aussi bien pour les entiers, les caractères et les pointeurs. Aujourd'hui, la bonne pratique est plutôt d'utiliser la macro `NULL`.

À noter que les langues, pas seulement les langages de programmation, peuvent avoir ce souci de noter l'inexistence. Le lojban a "*zo'e*" pour cela. Pour dire « je parle en lojban », où il faut normalement spécifier le public et le sujet **avant** la langue utilisée, une des méthodes est d'indiquer les arguments manquants par "*zo'e*" donc, ici "*mi tavla zo'e zo'e la lojban*".

Merci à Bertrand Petit pour sa suggestion originale.