Teórico 3 – Fundamentos de matemática

Introducción

Al igual que hicimos con lógica, deberemos hacer algunas modificaciones a nuestra teoría clásica para adaptarla a nuestras necesidades particulares.

Veremos particularidades relacionadas a la teoría de conjuntos (cardinalidad, elementos duplicados y elementos NULL).

Conceptos básicos

Cardinalidad

Un conjunto puede ser finito o infinito, pero para nuestra aplicación a las bases de datos nuestros conjuntos serán siempre finitos. Los conjuntos finitos tienen la particularidad de que su cardinalidad es siempre un número natural que denota el número de elementos que contiene. Llamaremos vacío al conjunto que no contiene elementos (cardinalidad 0) y llamaremos conjunto unitario (singleton) al conjunto que contiene un elemento (cardinalidad 1).

Como hicimos en el caso de la lógica, podría sernos útil definir funciones. Supondremos que tenemos una función COUNT que dado un conjunto retorna su cardinalidad (en definitiva, cuenta los elementos que tiene el conjunto).

Predicados sobre elementos

También podría ser útil recordar el método familiar de definir conjuntos por comprensión, esto es, seleccionar algunos elementos de un conjunto conocido, por medio de alguna propiedad (un predicado). Deberemos tener entonces alguna forma de realizar una selección (SELECT). Por ejemplo, los naturales mayores que 2 y menores que 7 se podrían seleccionar de alguna forma.

S = { n : N | n > 2 AND n < 7 }

Algo a tener en cuenta es que podemos seleccionar el propio conjunto de origen, al especificar una propiedad que siempre sea cierta, y llamaremos a una selección de este tipo, selección irrestricta (o FULL SELECT):

F = { n : N | 1 = 1 } ...o simplemente, F = { n : N | }

De manera análoga, podríamos obtener el conjunto vacío, especificando una condición que sea siempre falsa:

V = { n : N | 1 = 2 }

Una selección es, en definitiva, un subconjunto. Decimos que A es subconjunto de B si cada elemento de A pertenece a B, y decimos que A está incluido en B. Necesitaremos una función para indicar que un elemento pertenece a un conjunto. Usaremos la función booleana IN para predicar sobre la pertenencia de un elemento en un conjunto. Por ejemplo:

5 IN { n : N | n > 2 AND n < 7 } ...o equivalentemente, 5 IN S

Predicados cuantificados

Entre elementos y conjuntos tenemos un problema de impedancia, no podemos comparar elementos con conjuntos. Sin embargo es común tener predicados como "18 es mayor que todo elemento de S" (cuantificador universal) y "5 es igual que algún elemento de S", "existe un elemento de S igual a 5" o "5 pertenece a S" (cuantificador existencial e idea de pertenencia).

Admitiremos entonces predicados cuantificados, que predican sobre relaciones de orden entre un elemento y una cuantificación de elementos de un conjunto. Siguiendo los ejemplos:

18 > ALL (S)

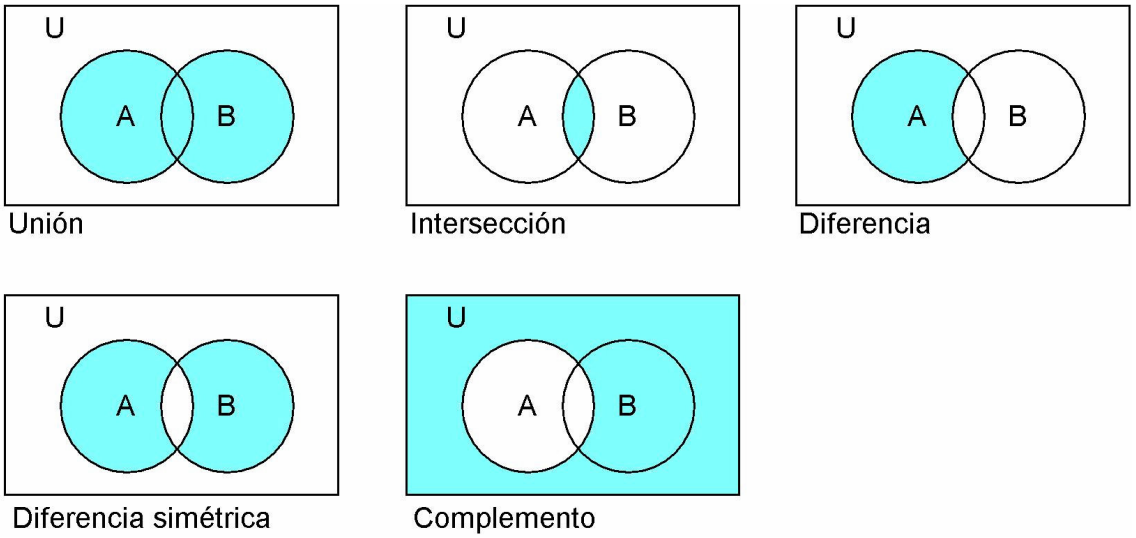
5 = ANY (S) ...o equivalentemente, 5 IN S

El cuantificador existencial sirve para predicar sobre la pertenencia (un elemento pertenece a un conjunto si EXISTE en el conjunto un elemento IGUAL). Sin embargo, el cuantificador existencial tiene usos más allá de la idea de pertenencia. Por ejemplo, 5 es menor que algún elemento de S:

5 < ANY (S)

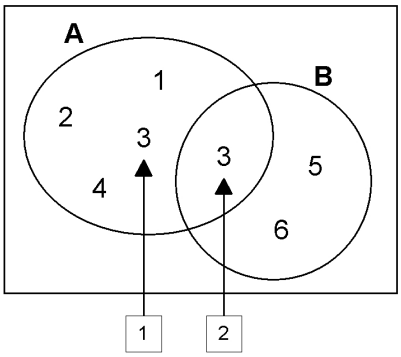
Operaciones sobre conjuntos y elementos repetidos

Ahora recordaremos las operaciones básicas de conjuntos: unión, intersección, diferencia, diferencia simétrica y complemento. Siempre se puede considerar que existe un conjunto universal, necesario para la idea de complemento (complemento respecto al conjunto universal).



Una característica de los conjuntos, en matemáticas, es que no tienen elementos repetidos. Esta condición se puede relajar para admitir elementos repetidos, y estas colecciones se denominan multiconjuntos.

La extensión de las operaciones antes mencionadas para multiconjuntos no es tan simple como parecería. Consideremos un multiconjunto A que contiene los elementos 1, 2, 3, 3 y 4 (note que el 3 está dos veces) y otro multiconjunto B que contiene los elementos 3, 5 y 6.



Estaremos de acuerdo en que la intersección de los multiconjuntos A y B contendría un elemento 3, pero deberíamos estar seguros que los elementos 3 del conjunto A son exactamente iguales e intercambiables, de otra forma, si pudiéramos distinguirlos (primer 3 y segundo 3 del conjunto A)...

¿cuál estaría en la intersección con B?

Otro aspecto a considerar al trabajar con multiconjuntos, es que tal vez se quieran eliminar los elementos repetidos, manteniendo sólo los distintos. Podría ser útil contar para tales efectos con una función DISTINCT, que tome un multiconjunto y retorne un conjunto (sin repetidos).

En nuestro ejemplo:

```
DISTINCT A = {1, 2, 3, 4}
```

También podríamos definir operadores de unión, intersección y diferencia que retornen multiconjuntos o que retornen conjuntos sin repetidos. Llamaremos UNION_ALL, INTERSECT_ALL y EXCEPT_ALL a los operadores que preservan los elementos repetidos, y por lo tanto devuelven multiconjuntos, y llamaremos UNION, INTERSECT y EXCEPT a los operadores que devuelven conjuntos (o sea, eliminan repetidos).

Es importante destacar que la eliminación de los repetidos en los casos de UNION, INTERSECT y EXCEPT se hace en primer lugar, sobre los operandos, y después se resuelve la operación. Em otras palabras, convierten primero los multiconjuntos en conjuntos y después realizan la operación.

En nuestro ejemplo:

```
A UNION B = {1, 2, 3, 4, 5, 6}
A UNION_ALL B = {1, 2, 3, 3, 4, 5, 6}
A INTERSECT B = {3}
A INTERSECT_ALL B = {3}
A EXCEPT B = {1, 2, 4}
A EXCEPT_ALL B = {1, 2, 3, 4}
```

Por ejemplo, la diferencia simétrica se podría extender a multiconjuntos, y podríamos definir:

```
A SYMMETRIC_DIFFERENCE B = (A EXCEPT_ALL B) UNION_ALL (B EXCEPT_ALL A)
```

Queda como tarea calcular la diferencia simétrica de los conjuntos del ejemplo.

Ahora que hemos definido la intersección, recordaremos que dos conjuntos se denominan disjuntos cuando su intersección es el conjunto vacío. Al haber definido la unión, podemos recordar con propiedad el concepto de partición; una partición de un conjunto S no vacío, es un conjunto de conjuntos disjuntos cuya unión es S.

Tipos de datos

Nos será muy útil, darle nombres (como abreviaturas) a ciertos conjuntos finitos, que utilizaremos como tipos de datos, por ejemplo:

```
DATE            = { d | d es una fecha entre el 01/01/0001 y el 12/12/9999 }
INTEGER         = { i | i es un entero entre -2.147.483.648 y 2.147.483.647 }
NUMBER(P, S)    = { n | n es un decimal de P dígitos de los cuales S son decimales }
VARCHAR(N)      = { v | v es un string de hasta N caracteres }
```