NUEVAS DIRECCIONES EN EL ACESO E INTEGRACIÓN DE DATOS

Simone Santini

Escuela Politécnica Superior
Universidad Autónoma de Madrid

1975 (datos bancários)	2005 (datos *imaging*)
almacenados en ficheros, en formatos "ad-hoc"	almacenados en ficheros, en varios formatos estándar, directos a la visualización
para el acceso, se escriben proceduras en FORTRAN o Cobol	para el acceso, se usan proceduras en C, Java, Perl; si los datos se encuentran en un sitio web, las proceduras se escriben como *servlets*
es necesario conocer no sólo la estructura lógica de los datos, sino también la manera en que están puestos en el disco	es necesario conocer no sólo la estructura lógica de los datos, sino también el formato de datos y acaso la estructura del sitio web donde se encuentren.

En el caso de los datos bancários: la situación cambia hacia 1980 con las bases de datos relacionales, que permiten un aceso a la estructura *lógica* de los datos indipendientemente de su estructura física.

```
select  dept, avg(age)
from    employees
where   salary > 30.000
groupby dept
```

En el caso de datos iconográficos *(imaging)*, todavia no existe un formalismo que permita una tratación tan general. El problema no es exlusivamente tecnico: hay que analizar los asuntos a la base del modelo relacional:

i) una base de datos es formada de datos que pertenecen a ciertos *tipos base* definidos *a priori,* agrupados en estructuras omogeneas y completamente expresadas;

ii) tipos de datos más complejos (p.ej.objetos) son declarados explicitamente en sus componientes estructurales, y estas componientes son datos "macroscópicos" con un valor semántico definido;

iii) cada dato tiene una representación única en el sistema;

iv) los datos se agrupan en estructuras de un tipo definido (conjuntos de tuplas o, más recentemente, árboles XML).

Los datos iconográficos no respectan estos supuestos.

Ejemplo:

Volúmen: demasiado complejo para considerarse un tipo base y, además es necesario aceder a sus componentes en el curso de una interrogación.

No se puede considerar como una estructura (relacion) definida explicitamente. Imaginese una definición de relación:

Vol(X:int, Y:int, Z:int, value:float)

y una interrogacion sobre el tamaño de la region connecteada más grande en la cual una cierta condición sobre ela valor esta verificada, tal que el calculo de la región se tenga que expresar en SQL:

```
select count(v)
from   Vol
where  ...
```

Datos como volumenes se representan con características.

Definición de estructuras: muchos almacenes de datos interesantes no publican una estructura relacional. Ejemplos:

base de datos orientadas a objetos

data banks de biologia

ontologias

archivos de citaciones

En estos casos se podrán utitizar en pleno todas las posibilidaded que la fuente de datos ofrece sólo si el sistema de almacenamiento central o de integración de datos permite de exprimir las mismas estructuras que se encuentran en estas fuentes.

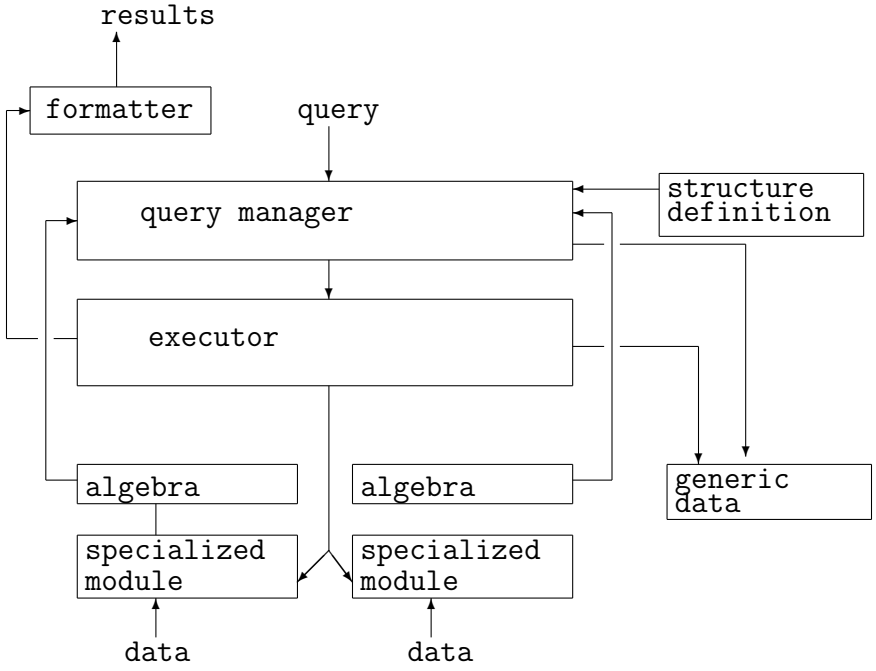
Por ejemplo, la banca de datos *gene ontology* es un grafo acíclico de 23.000 nodos generados por las relaciones *is-a* y *part-of*.
La "research intellectual property network" es una red de citaciones de patentes que es también estructurada como un grafo acíclico.

Los problemas generales que se deben enfrentar para definir una base de datos de imaging data:

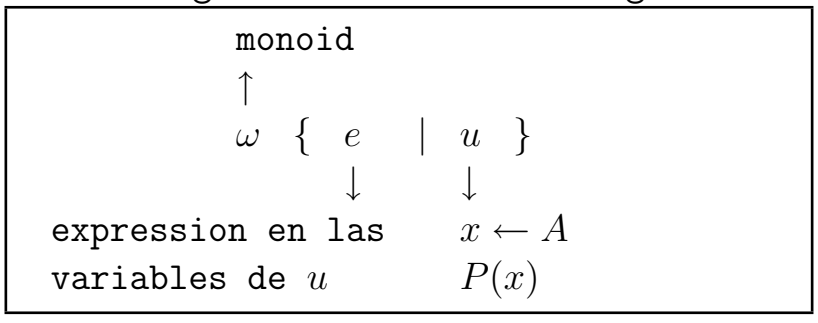
i) una integración en la base de datos de estructuras complejas y variads que incluyan conjuntos, listas, árboles, y grafos;

ii) una representación de tipos de datos abstractos, así que se pueda acceder a la estructura de datos iconográficos (como volumenes) sin que sea necesario representarlos explicitamente en el sistema;

iii) la posibilidad de usar representaciones diferentes para los datos que se almacenan en la base de datos.


results
formatter
query
query manager
structure
definition
executor
algebra
algebra
generic
data
specialized
module
specialized
module
data
data

forma general de una interrogación



$\omega\{e|x \leftarrow A\}$

```
r := null
foreach x in A do
    r := r ω e(x)
od
```

$\cup\{x^2|x \leftarrow [1,2,3]\}$

$r := \emptyset$
foreach $x \in A$ do
$r := r \cup \{x^2\}$
od

$\{1,3,9\}$

$\omega\{e|x \leftarrow A, P(x)\}$

```
r := null
foreach x in A do
    if P(x) then
        r := r ω e(x)
    fi
od
```

$\vee\{\texttt{true}|x \leftarrow [1,2,3], x > 2\}$

$r :=$ false
foreach $x \in A$ do
    if $x > 2$ then
        $r := r \vee$ true
    fi
od

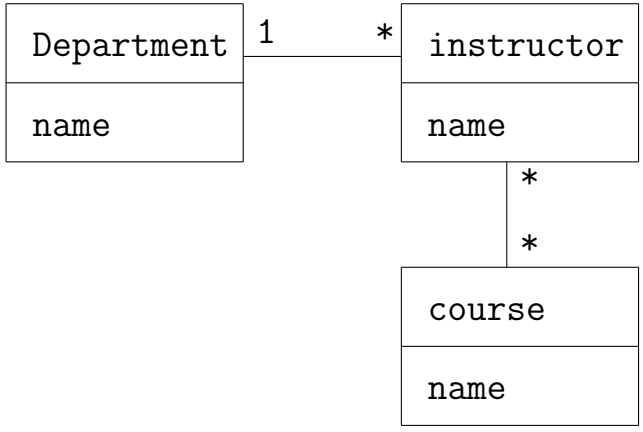
true

Esta sintaxis puede exprimir toda la algebra relacional (SQL), y las más comunes algebras para bases de datos orientadas a objetos (OQL). Y más...

```
class Department {
  name :  string;
  instructors :  set(instructor);
}

class instructor {
  name :  string;
  teaches :  set(course);
}

class course {
  name :  string;
}
```



$$\cup\{\texttt{e.name} \mid el \leftarrow \cup\{d.\texttt{instructors} | d \leftarrow \texttt{Depts}, d.\texttt{name} = CSE\},$$
$$e \leftarrow el, \vee\{c.\texttt{name} = cse101 | c \leftarrow e.\texttt{teaches}\}\}$$

Busca todos los profesores del departamiento CSE que enseñan cse101---se note que la relación entre profesores (instructors) y cursos es "many to many".

La primera interrogación saca el conjunto de todos los profesores de CSE que, de echo, es un conjunto de conjuntos. La expresión $e \leftarrow el$ hace un *unnesting* de estos conjuntos. La expresión final busca, entre todos estos profesores, los que enseñan cse101.

Para insertar grafos, se usan expresiones tipo patrones, de la misma familia de los patrones que se usan en XPath para hacer interrogaciones en árboles.

Las expresiones que yo uso permiten no solo la definición de patrones que partes de grafos pueden satisfacer, sino aún la definiciń de *variables* a las cuales partes del grafo que encaja con el patron se pueden asignar.

$$g \vdash \pi(x_1 : \pi_1, \ldots, x_n : \pi_n)$$

Esta expresión genera un conjunto de tuplas $\langle x_1, \ldots, x_n \rangle$, una para cada combinación de variables corrispondientes a una parte del grafo que encaja con el patrón.

grafos: $\Gamma(\nu)$
nodos: $\nu = (\mathtt{id} : \mathtt{skolem}, v : \mathtt{int})$

patron: $\omega\{e|x \vdash \pi, x \leftarrow G\}$

$G : \{\!|\Gamma\alpha|\!\}$, $G : \{\Gamma\alpha\}$, ...
$x : \Gamma(\alpha)$.

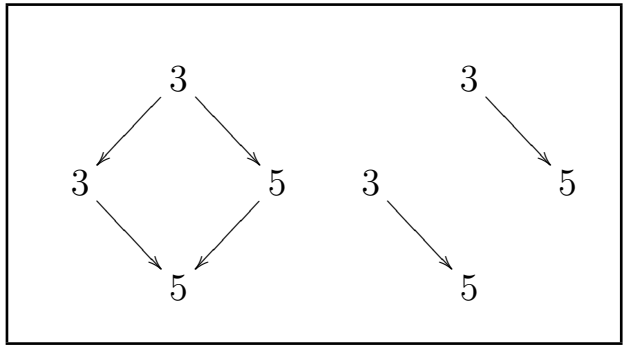
ejemplos: $\cup\{\{x\}|x \leftarrow G, x \vdash (v = 3)\}$ grafos con un nodo tal que $v = 3$;
$\cup\{n|x \leftarrow G, x \vdash n{:}(v = 3)\}$ *nodos* con $v = 3$
nota: $n : \{\nu\}$.

Nota: para simplificar la notación, desde este momento escribiré las condiciones del tipo $(v = 3)$ como 3.

satisfación de un patron: $g \models \pi$

Composición serial:
$g_1 \models \pi_1, g_2 \models \pi_2 \Rightarrow g \models \pi_1 — \pi_2$: g es el "sink to source" de g_1, g_2.

$\uplus\{n|x \leftarrow G, x \vdash n{:}(3 — 5)\} \Longrightarrow$



$\uplus\{(n, m)|x \leftarrow G, x \vdash (n{:}3 — m{:}5)\} \Longrightarrow$ $\{\!|(3, 5), (3, 5)|\!\}$

repetición de patrones:

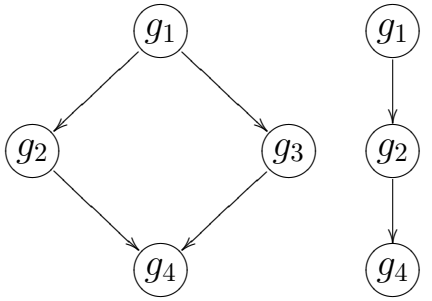
$$\pi_1[\text{—}\pi_2](n) \equiv \pi_1 \overbrace{\text{—}\pi_2\text{—}\pi_2 \cdots \text{—}\pi_2}^{n}$$

$\pi_1[\text{—}\pi_2](n,m)$ $\quad$ $\pi_1[\text{—}\pi_2]* \equiv \pi_1[\text{—}\pi_2](0,\infty)$ $\quad$ $\pi_1[\text{—}\pi_2]+ \equiv \pi_1[\text{—}\pi_2](1,\infty)$

---

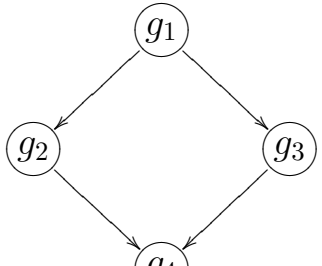
composición paralela:

$\pi_1\{\text{—}\pi_2, \text{—}\pi_3\}\text{—}\pi_4$



este patrón no requiere que los dos caminos sean distintos.

$\pi_1!\{\text{—}\pi_2, \text{—}\pi_3\}\text{—}\pi_4$



este si!

$\omega \ \{ \ e \ | u\}$
$\downarrow \quad \downarrow$

expressions para construir resultados

Todas la variabiles definidas en u pueden ser mencionadas en e

Types: ν : nodos *(nodes)*;
$\Pi(\nu)$: caminos *(paths)*;
$\Gamma(\nu)$: grafos *(graphs)*

$\mathtt{dom}(\alpha) \subseteq \mathtt{dom}(\Pi(\alpha)) \subseteq \mathtt{dom}(\Gamma(\alpha))$

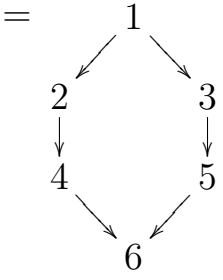
gmin, gmax: monoid min y max para grafos:

$$g_1 \leq g_2 \Leftrightarrow N(g_1) \subseteq N(g_2) \wedge E(g_1) \subseteq E(g_2)$$

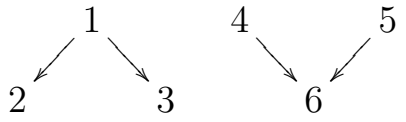
$a, b, : \Gamma(\alpha)$: a—b connecta b a a;

"merge": une dos grafos identificando los nodos con valores iguales.

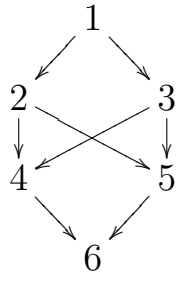
$g =$



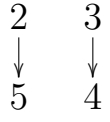
$$\cup\{(p,q) \mid g \vdash p : (1\{\text{—}2, \text{—}3\}),\ g \vdash q : (\{4\text{—}, 5\text{—}\}6)\}$$



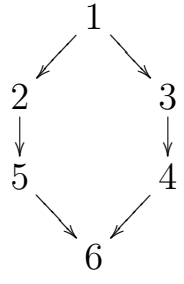
$$\cup\{p\text{—}q \mid g \vdash p : (1\{\text{—}2, \text{—}3\}),\ g \vdash q : (\{4\text{—}, 5\text{—}\}6)\}$$



$$\cup\{(r\text{—}s, t\text{—}w) \mid g \vdash p : (1\{\text{—}r : 2, \text{—}t : 3\}),\ g \vdash q : (\{w : 4\text{—}, s : 5\text{—}\}6)\}$$



$$\cup\{\texttt{merge}\{p,q,r,s,t,w\} \mid g \vdash p : (1\{\text{—}r : 2, \text{—}t : 3\}),\ g \vdash q : (\{w : 4\text{—}, s : 5\text{—}\}6)\}$$



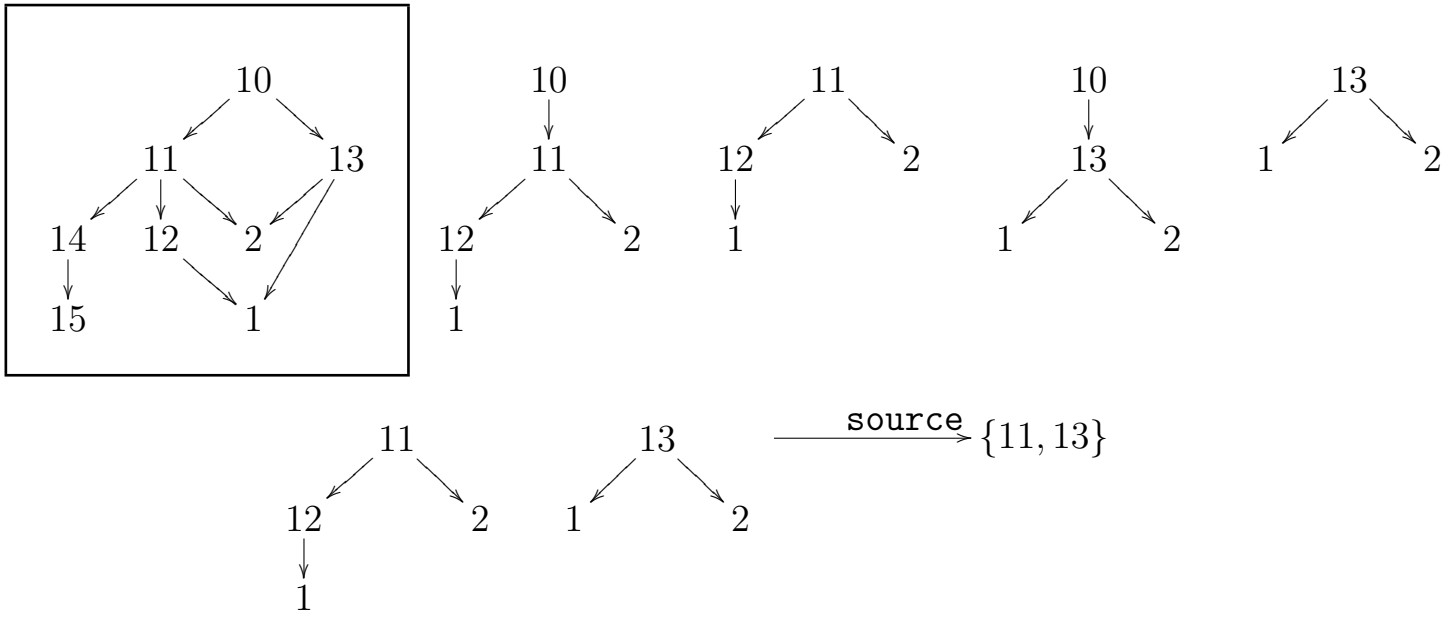
$$\texttt{merge}\{p,q,r,s,t,w\} = \texttt{merge}(p, \texttt{merge}(q, \texttt{merge}(r, \texttt{merge}(s, \texttt{merge}(t, w)))))$$

$$\boxed{\cup\{p | g \vdash p : (1\{\text{—}2\text{—}5, \text{—}3\text{—}4\}\text{—}6)\}}$$

Antepasado más proximo *(least common ancestor)* de los nodos con id=1 e id=2

($DB = \{\!|\Gamma(\alpha)|\!\}$)

$$\cup\{\texttt{source}(x)|x \leftarrow \texttt{gmin}\{\ a|$$
$$x \vdash a : ((\texttt{true})\{[\!\!-\!\!-(\texttt{true})] * \!\!-\!\!-(id = 1), [\!\!-\!\!-(\texttt{true})] * \!\!-\!\!-(id = 2)\}),$$
$$x \leftarrow \texttt{DB}$$
$$\}$$
$$\}$$



Unos ejemplos de una red de patentes:

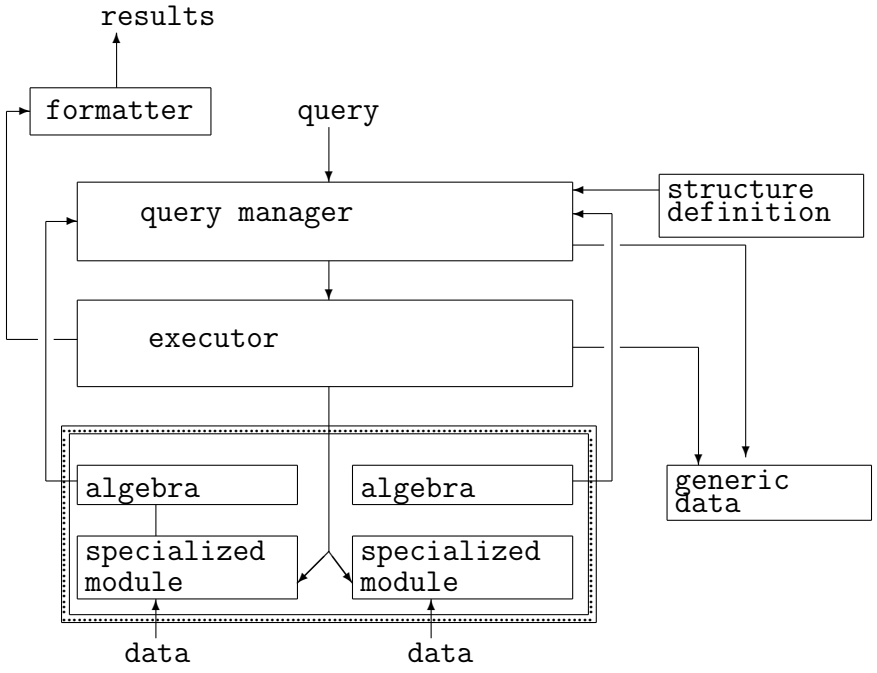
find all patents that can be traced to patent X

$$\cup\{a|g \leftarrow DB, g \vdash a : t[\text{—}t] * \text{—}(name = X)\}$$

"Find all patent that have two distinct citation paths to X"

$$\cup\{a|g \leftarrow DB, g \vdash a : t!\{[\text{—}t]*, [\text{—}t]*\}\text{—}(name = X)\}$$

Tipos de datos complejos: la estructura de los datos iconográficos (imaging) es demasiado fina, y sus elementos no tienen bastante valor semántico para que se puedan almacenar directamente. En este caso, el sistema que se está desarrollando prevee su definición como *tipos de datos abstractos,* definidos por sus *algebras*.



En el caso específico, la idea es de proceder a una definición completa y general de algebras de caracteristicas *(feature algebras)* para traducir interrogaciones en la *algebra interfaz* del tipo de datos a un *query plan* expresado en las algebras de características.

Ejemplo: definición de datos de imágenes.

Abstract data type:

```
type: image, region, color;
constraint image < region;
function: sim : image, image -> float;
          segment : image -> {region};
          area : region -> float;
          carea : image, color -> float;
          creg : image, region, color -> float;
end
```

Se asuma que las imagenes son representadas con dos caratterísticas: un histograma de color y un conjunto de regiones, cada una con su histograma de color. En el sistema se añade la seguiente definición

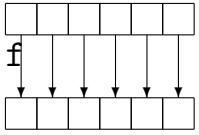
```
feature: hist, sphist;
representation: h : image -> hist;
                sh : image : sphist;
functions:      hval : hist, color -> float;
                sval : sphist, region, color -> float;
equations:      carea(img, col) == hval(h(img), col);
                carea(img, col) == sum(r in segment(img))
                                     (sval(sh(img), r, color));
                creg(img, r, color) = sval(sh(img), r, color);
end
```

Para permitir este tipo de traducción entre tipos de datos y caratterísticas, se está procediendo a una definición y clasificación general de los tipos de caratterísticas de imagenes, y a una definición de algebras adecuatas para los varios tipos de imagenes.

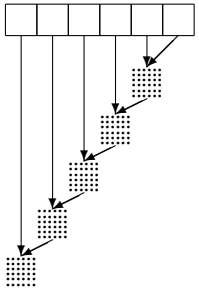
Para dar un ejemplo, se han empezado a estudiar los datos organizados en *array*. Se consideran sólo array unidimensionales, ya que un array de dimensionalidad n se puede considerar como un array unidimensional de arrays de dimensionalidad $n-1$.

La algebra de array solo consiste en dos operatores:

i) la aplicación de función m: $m.f.[u_1, \ldots, u_n] = [f(u_1), \ldots, f(u_n)]$;

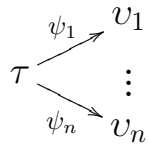


ii) la reducción r: $m.f.[u_1, \ldots, u_n] = f(u_1, f(u_2, \cdots, f(u_n, 0) \cdots))$;



Basandose en estos operadores, y en funciones f especializadas, se ha definido, por ejemplo, una algebra de histograma, aplicándola a histogramas de color, de direcciones de *edges*, etc.

características:



cada característica permite una implementación eficiente de una *parte* del algebra del tipo.

Algebra de funciónes:

evaluación:

$$\frac{f : \alpha \to \beta \qquad a : \alpha}{f.a : \beta}$$

composición:

$$\frac{f : \delta \to \beta \qquad g : \alpha \to \delta}{f \circ g : \alpha \to \beta}$$

producto Cartesiano:

$$\frac{f : \alpha \to \beta \qquad g : \delta \to \gamma}{f \times g : \alpha \times \delta \to \beta \times \gamma}$$

homomorphismos:

$$\frac{f : \alpha \to \nu(\beta)}{[\tau, \nu](f) : \tau(\alpha) \to \nu(\beta)}$$

$[\tau, \nu] : (\alpha \to \nu(\beta)) \times \tau(\alpha) \to \nu(\beta)$;

condicional:

$$\frac{f : \alpha \to \beta \qquad g : \alpha \to \beta \qquad P : \alpha \to \mathbf{2}}{\mathtt{if}(P, f, g) : \alpha \to \beta}$$

Constants:

identity: $\iota : \alpha \to \alpha : x \mapsto x$;

projections: $\pi_i : \alpha_1 \times \cdots \times \alpha | n \to \alpha_i : (x_1, \ldots, x_n) \mapsto x_i$;

diagonal: $\Delta : \alpha \to \alpha \times \alpha : x \mapsto (x, x)$.

El futuro...

Incluir sistemas de base de datos en sistemas de flujo de datos: en el caso de tipo de datos imaging, muy a menudo el aceso a una base de datos se efectua como parte de un proceso mas complejo que incluye el procesamiento de los datos que se buscan. Estos procesos muchas veces se expresan como flujos de datos, y la integración de bases de datos en sistemas de flujo de datos ofrece muchas oportunidades de optimización.

Ejemplo: base de datos de pacientes con (entre otros) la edad del paciente y un volumen MRI del cerebro.