A Functional Model for Data Analysis

Nicolas Spyratos

Laboratoire de Recherche en Informatique

Universite Paris Sud 11

France

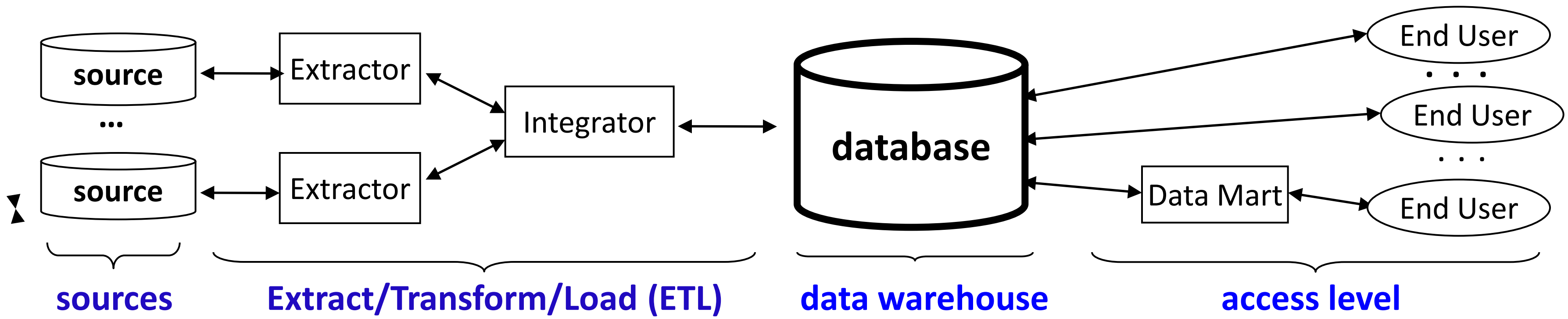
«The quest for knowledge used to begin with grand theories. Now it begins with massive amounts of data. Welcome to the Petabyte Age »

(WIRED, July 2008)

in several fields data is accumulated over long periods of time and analyzed in order to discover tendencies, outliers, potential problems etc. *(cf. business intelligence, dashboards, …)*

the data to be analyzed often comes from several possibly heterogeneous sources therefore it has to be 'homogenized' before analysis takes place *(cf data warehousing)*

Data Warehousing



some facts relevant to my talk:

- **the data warehouse is usually implemented as a relational database**
- **its schema is usually not normalized (usually a 'star schema')**
- **the query language consists mostly of group by queries**
- **the record based storage of the relational model doesn't seem to fit well the needs of data analysis**
- **due to the very large data volumes reuse of query results becomes important**

my talk concerns the last two points:

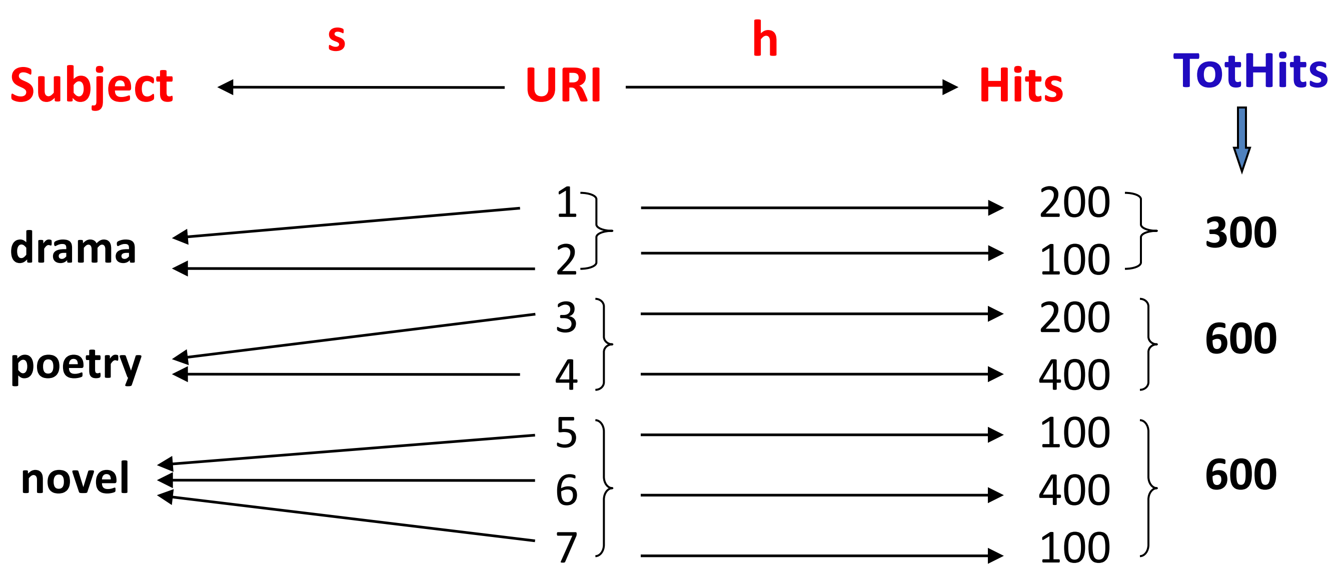
- **a functional data model and a language of analytic queries, in which it's possible to study query rewriting (including in presence of constraints)**

→ the functional data model that we use is the one proposed by Buneman and Fraenkel

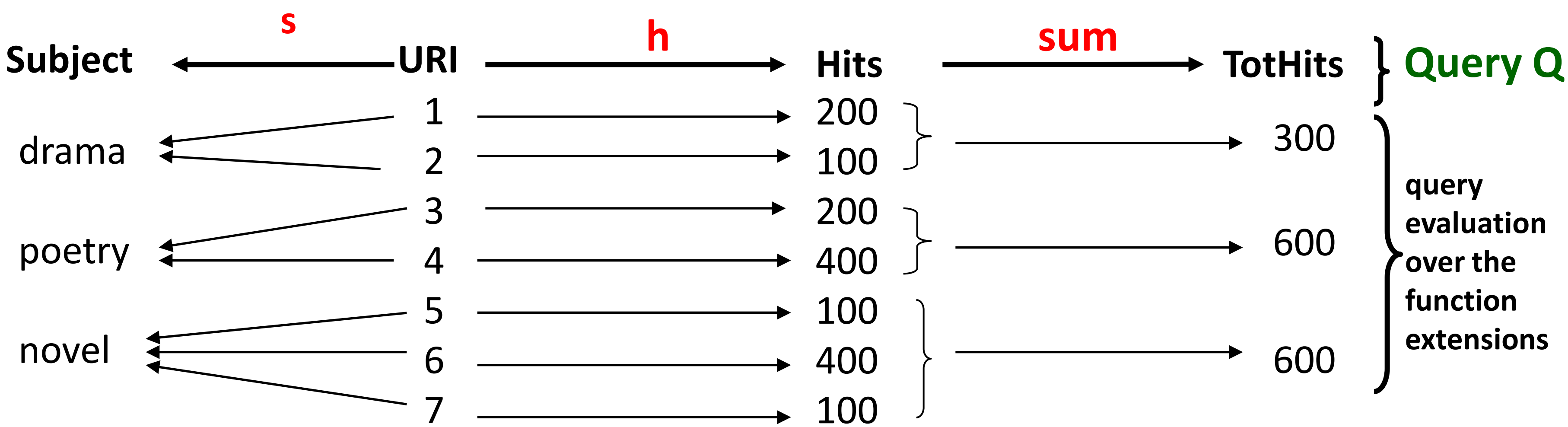
the basic concepts through examples

Assume a digital document collection, each document being identified by a URI and described by its subject and number of hits. To compute the total number of hits per subject we can proceed as follows:

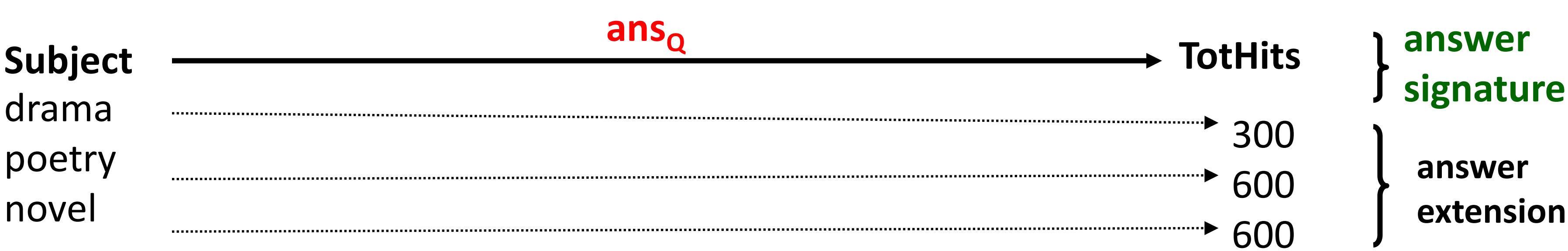
group the documents by subject *(using the function s)*
measure the number of hits *(by applying the function h)*
aggregate the measures in each group *(using the operation sum)*



more formally, grouping, measuring and aggregation needs three functions:



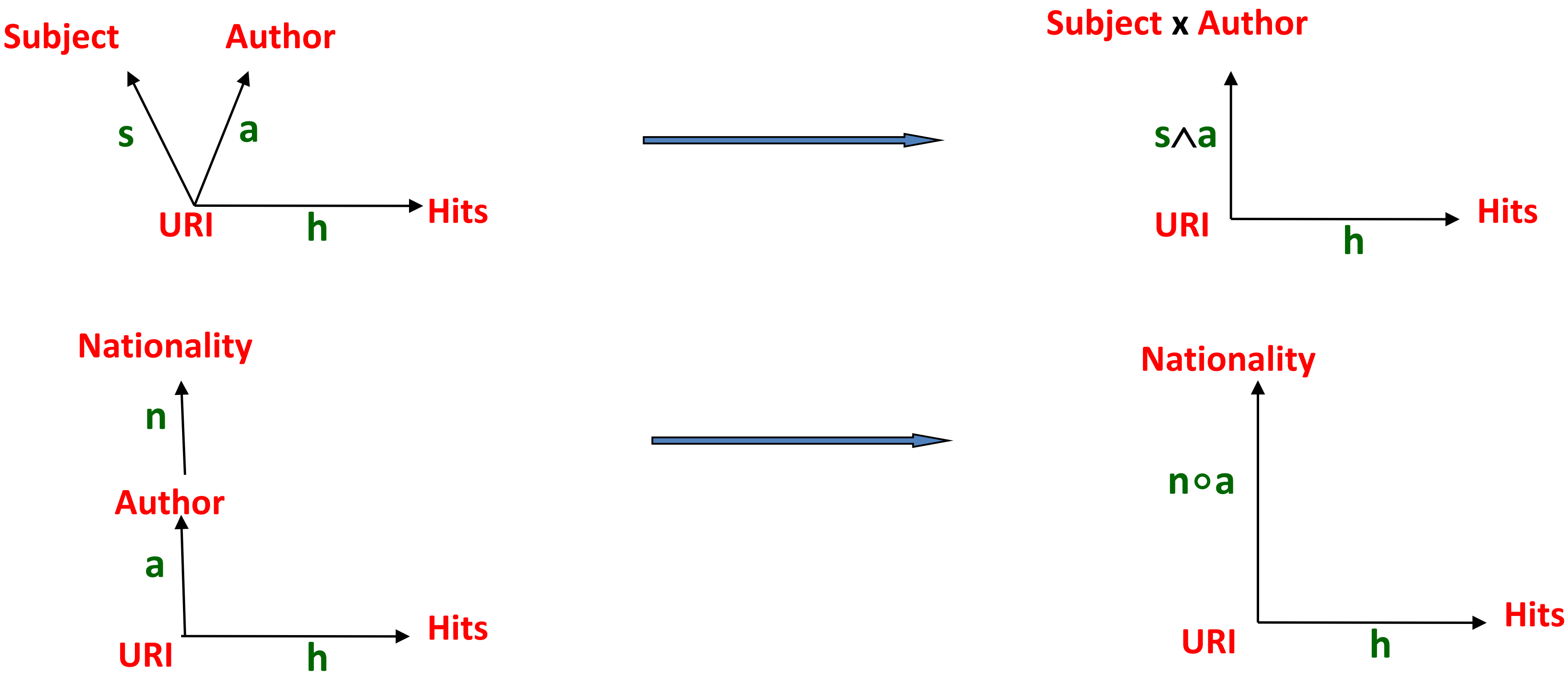
… and the end result of the query is a function from Subject to TotHits:



so an analytic query Q is a triple of functions such as <s, h, sum>
and the answer is a function as well, namely ans_Q: target(s)→target(sum)

question: ***what if we had a combination of grouping functions?***

answer: ***replace them by one function and do as before!***



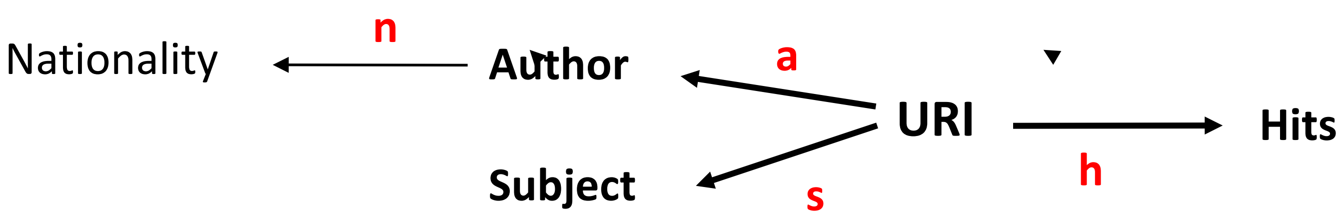
in general:

- **we combine functions using four operations : pairing, composition, projection, restriction**
- **and we use combinations of functions to form analytic queries such as <n∘a, h, sum>**

the model – schema *(or what are the functions that one can combine)* :

oriented, labeled, acyclic, connected graph in which

- **there is a single root** *(modeling the objects to be analyzed),*
- **each node is associated with a set of values** *(or domain, as in the relational model)*
- **all arrow labels are distinct** *(thus allowing for "parallel" arrows)*



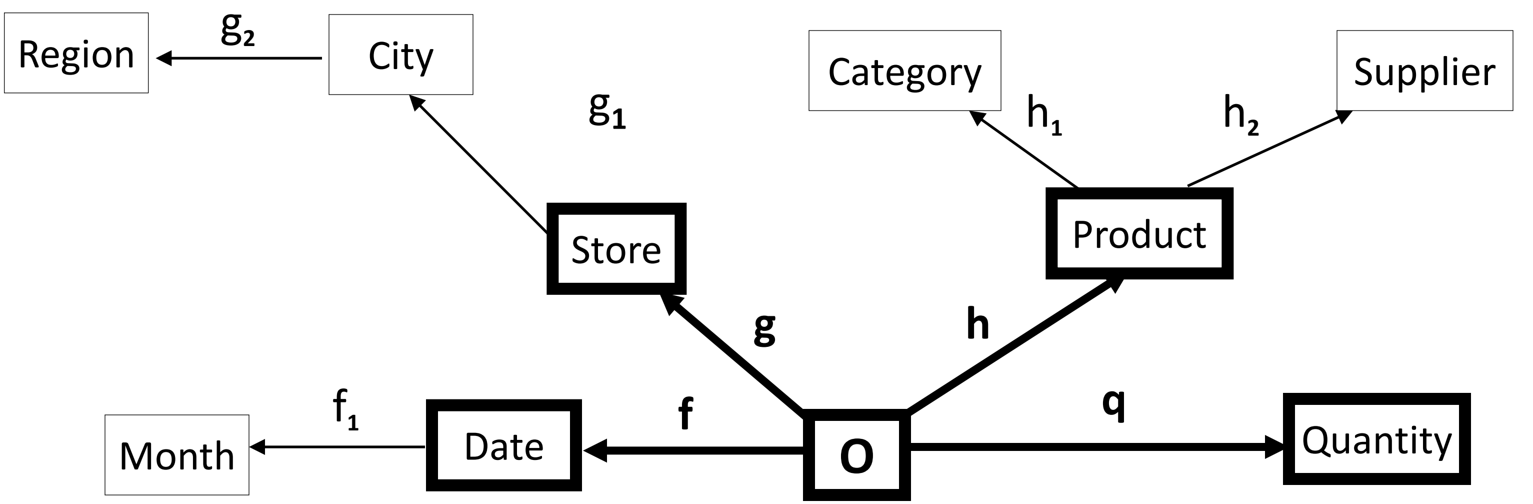
the successors of the root are the attributes of the objects modeled by the root, while the remaining nodes are attribute dependent ***indicators*** *to be used in the analyses*

→ **in fact the (reduced) set of functional dependencies in a relational BCNF table leads directly to such a schema** (simply choose a key as root and add indicators)

another example of schema:

a catering company delivering various products to a number of stores

(O represents delivery invoice numbers)

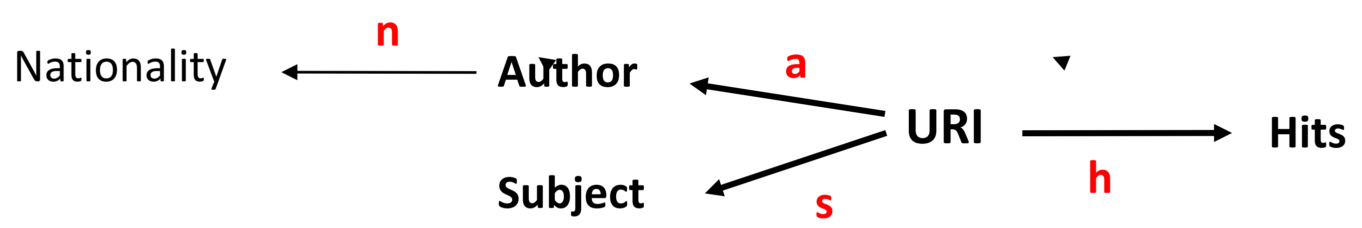


the model – database

a mapping δ that associates

- **each node A with a finite subset δ(A) of its domain**
- **each arrow f: A→B with a finite total function δ(f): δ(A)→ δ(B)**

(i.e. a database is a set of finite function extensions - one for each arrow in the schema)



Schema
(set of function signatures)

A

URI	Author
1	A1
2	A2
3	A1
4	A3
5	A4
6	A1
7	A2

S

URI	Subject
1	drama
2	drama
3	poetry
4	poetry
5	novel
6	novel
7	novel

H

URI	Hits
1	200
2	100
3	200
4	400
5	100
6	400
7	100

N

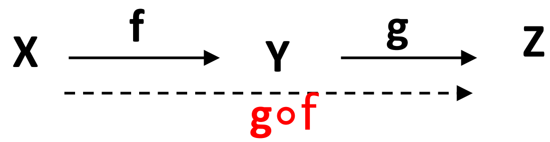
Author	Nationalitity
A1	French
A2	German
A3	Greek
A4	French

Database
(set of function extensions)

→ could be represented directly in MonetDB

the model – functional algebra

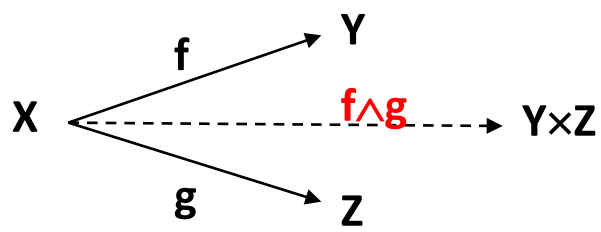
composition
(join + projection)



$g \circ f(x) = g(f(x))$

pairing

(join)



$f \wedge g(x) = \langle f(x), g(x) \rangle$

restriction

(selection)

$D \subseteq X$, $X \xrightarrow{f} Y$ $\quad D \xrightarrow{f/D} Y$

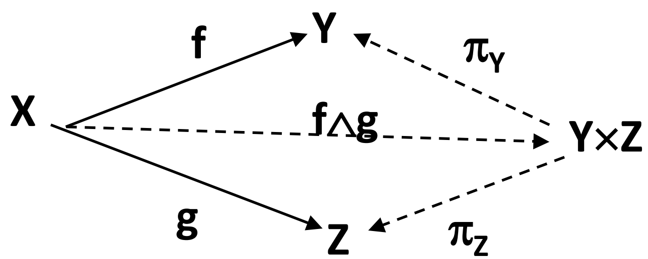
$f/D(x) = f(x) \quad \forall x \in D$

projection
(projection)

$A_1 \times \ldots \times A_n \xrightarrow{\pi_i} A_i$

$(a_1, .., a_i) = a_i$

the operations of this algebra are tied together by a fundamental property:



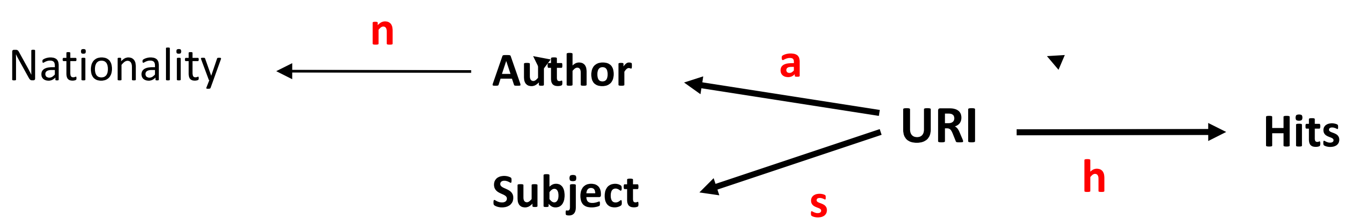
$\pi_Y \circ (f \wedge g) = f$

$\pi_Z \circ (f \wedge g) = g$

the model – path expression:

a well formed expression whose operands are arrows of the schema and whose operations are those of the functional algebra

Ex: s, $n \circ s$, $s \wedge a$, $s \wedge (n \circ a)$, …….



evaluation of path expression e w.r.t. database δ:

the result of replacing each arrow in e by the corresponding database function and performing the indicated operations *(the result is always a function)*

→every node A is equipped with two "extreme" path expressions: π_A and $\pi_\varnothing$

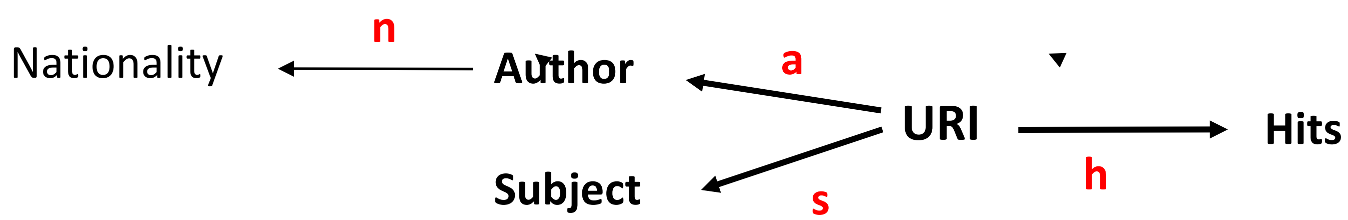
(evaluated as identity function and constant function over A, respectively, assuming δ(A) nonempty)

the model – analytic query:

triple $Q= <c, m, op>$, where c, m are path expressions with common source, and op an operation over the target of m

c is called the *classifier* or *grouping function*, m the *measure* and op the *aggregate operation*

Ex: $Q= <s \wedge (n \circ a), h, sum>$ *asks for total hits by subject and author nationality*



evaluation of the analytic query $Q=<c, m, op>$ is done in two steps:

1/ evaluate c and m to obtain two functions with common source (call them c and m as well)

2/ for each y in range(c) do

begin ***Group:*** compute the inverse $c^{-1}(y)$ { let $c^{-1}(y) = \{x_1, \ldots, x_r\}$}

Measure: for each $x \in c^{-1}(y)$ compute m(x) {this step returns a tuple $t(y)= < m(x_1), \ldots, m(x_r)>$}

Aggregate: apply the operation op to the tuple t(y) {call the result op(t(y))}

Answer define $ans_Q(y)= op(t(y)$

end

answer signature: ans_Q: target(c) → target(op)

Ex: $Q= <s \wedge (n \circ a), h, sum>$ ans_Q: Subject×Nat → TotHits

remarks on the definition of an analytic query:

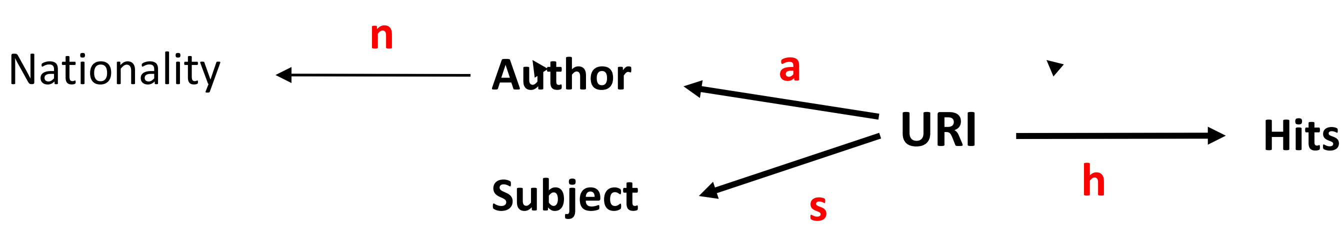
- **the grouping function is formed by composing along one or more paths then pairing, eventually restricting along the nodes of each path** (“where” clause of group by)

- **the answer to a query being a function, it can be restricted to a subset of its domain** (“having” clause of group by)

- **in the answer signature ans_Q: target(c) $\rightarrow$ target(op), the attributes in target(c) appear in the schema while target(op) does not. As a consequence we can introduce a name for target(op), by indicating it when we specify the query, e.g. <s, h, sum>/TotHits** (“as” in the select clause of group by)

- **the roles of c and m in a query can be interchanged** (with a possible change in the operation)

Ex: Q= <s, h, sum> versus Q’= <h, s, count>



reasoning in the model

- **query result exploration**

- **reusing query results**
  - comparing queries
  - rewriting a query in terms of another (comparable) query
  - managing a cache for reusing query answers

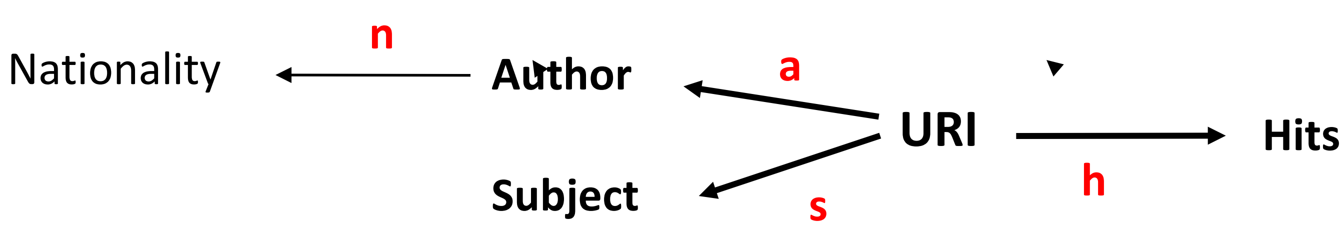
reasoning in the model- query result exploration

the basic idea:
a query answer being a function, it might have more than one equivalent representations

Example:

$Q = \langle s \wedge a, h, sum \rangle$
$ans_Q: Subject \times Author \rightarrow TotHits$

$$(Subject \times Author \rightarrow TotHits) \equiv (Subject \rightarrow (Author \rightarrow TotHits))$$
$$\equiv (Author \rightarrow (Subject \rightarrow TotHits))$$



such equivalences are useful for exploring result visualizations

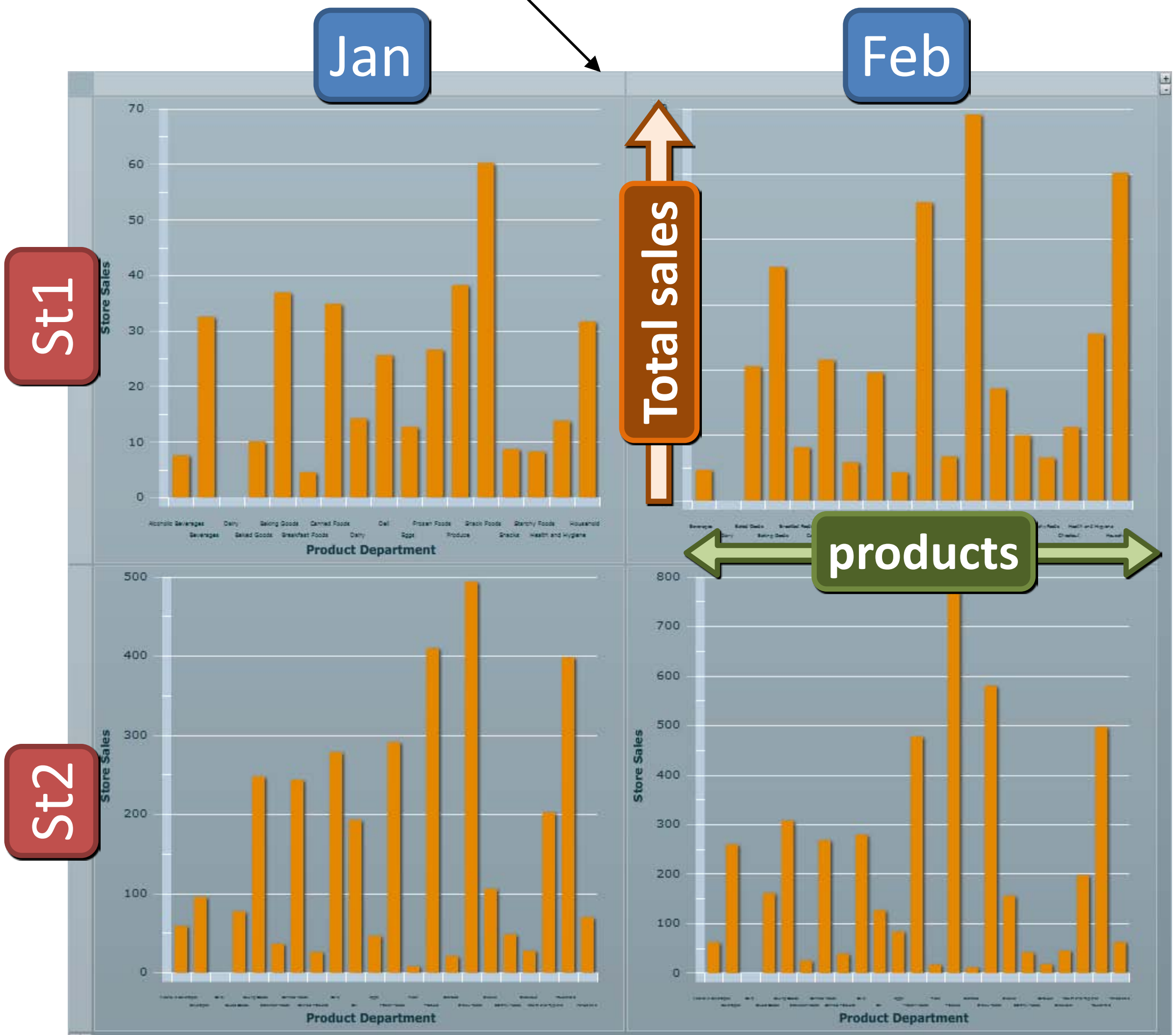
StorexProdxMonth→Tot ≡ StorexMonth→(Prod→Tot)

evaluation result

StorexProdxMonth →Total sales

STORE	PROD	MONTH	TOTAL SALES
St1	Prod1	Jan	103
St1	Prod1	Feb	204
St1	Prod1	March	251
…	…	…	…

result visualization



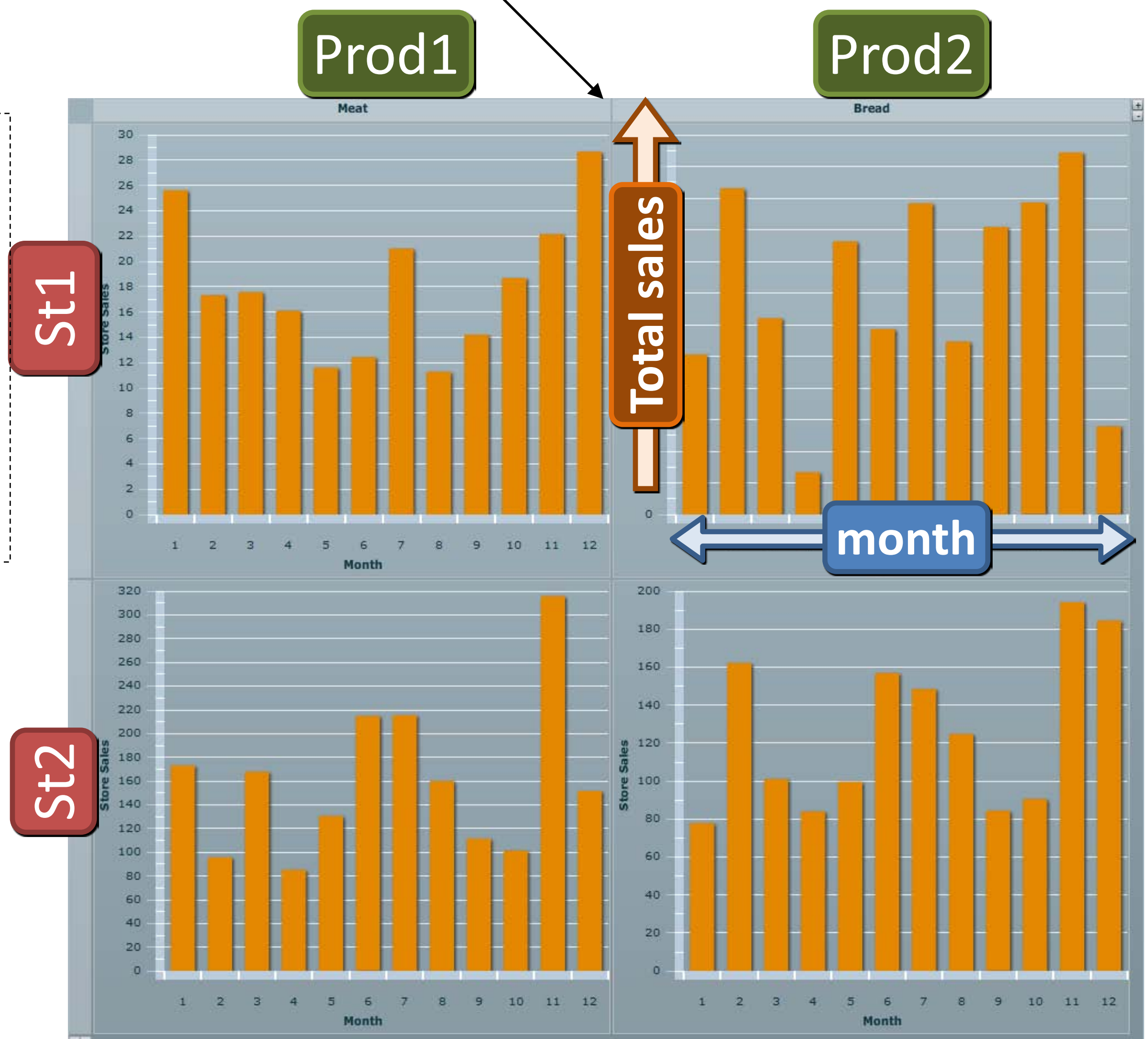
Store and Month act as grid parameters each cell showing total sales by product

StorexProdxMonth→Tot ≡ StorexProd→(Month→Tot)

evaluation result

STORE	PROD	MONTH	TOTAL SALES
St1	Prod1	Jan	103
St1	Prod1	Feb	204
St1	Prod1	March	251
…	…	…	…

result visualization



Store and Product act as grid parameters each cell showing total sales by month

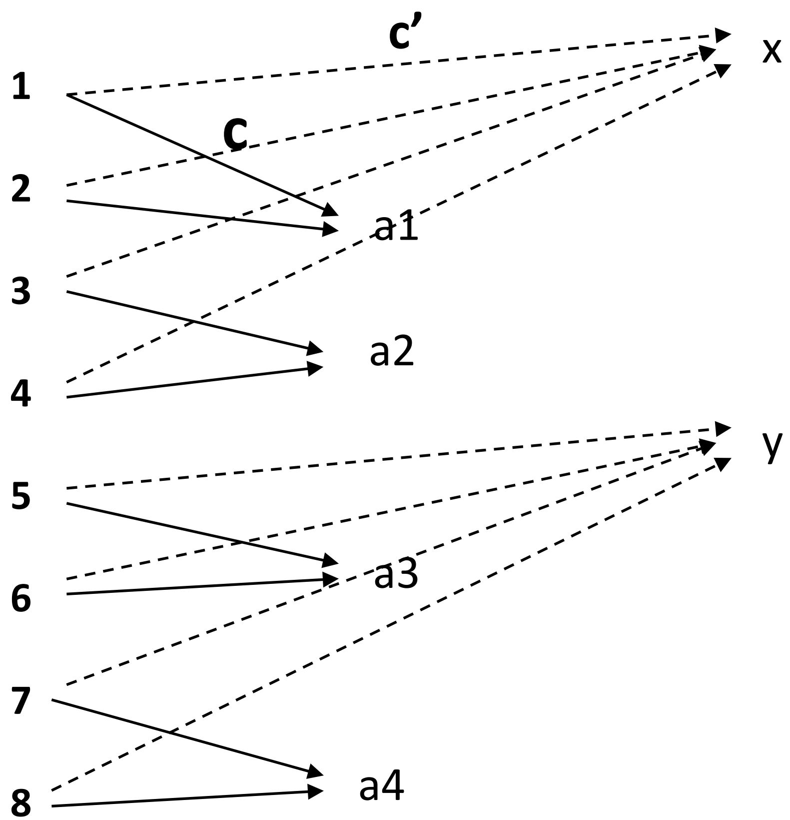
reasoning in the model- reusing query results

consider two analytic queries, Q=<c, m, op> and Q' =<c', m, op>

their grouping functions can be ordered (up to equivalence) **:**

c ≼ c' iff c(o)=c(o') entails c'(o)=c'(o')

c≡c' iff c ≼ c' and c' ≼ c

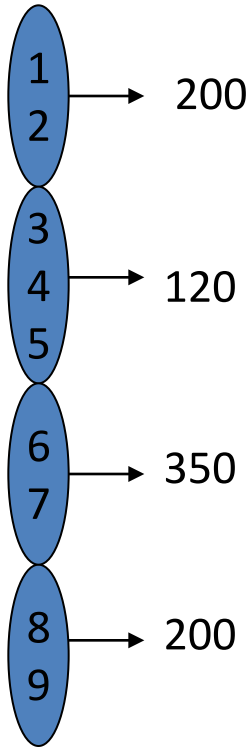


c ≤ c' as each group of c is contained in a group of c'

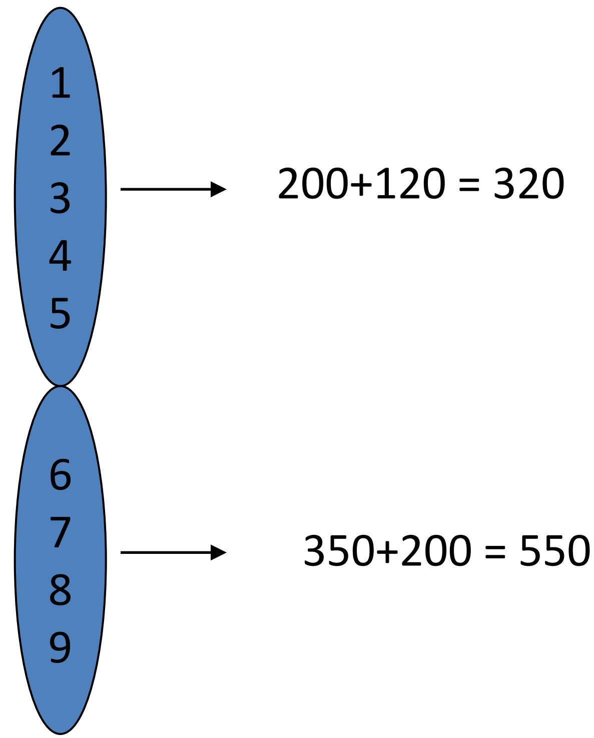
claim:

given two queries Q= <c, m, op> and Q'= <c', m, op>,

if c ≼ c' then Q' can be evaluated using the answer of Q *(for most common operations)*



evaluation of Q



evaluation of Q' from the answer of Q

there remains one question: ***how can we tell whether*** **c ≼ c'**

basic fact:

$f \preccurlyeq g$ iff there is h s.t. $h \circ f = g$ (it follows that: $f \preccurlyeq g \circ f$, $f \wedge g \preccurlyeq f$)

it turns out that all comparisons between two classifiers fall into one of the following cases:

Case 1: ***$f_n \circ \ldots \circ f_1 \preccurlyeq g_m \circ \ldots \circ g_1$ iff there is h s.t. $h \circ (f_n \circ \ldots \circ f_1) = g_m \circ \ldots \circ g_1$***

Case 2: ***$f_1 \wedge \ldots \wedge f_n \preccurlyeq g_1 \wedge \ldots \wedge g_m$ iff $\forall g_i \exists$ sub-pairing $s_i = f_{j1} \wedge \ldots \wedge f_{ji}$ s.t. $s_i \preccurlyeq g_i$ $\forall i$***

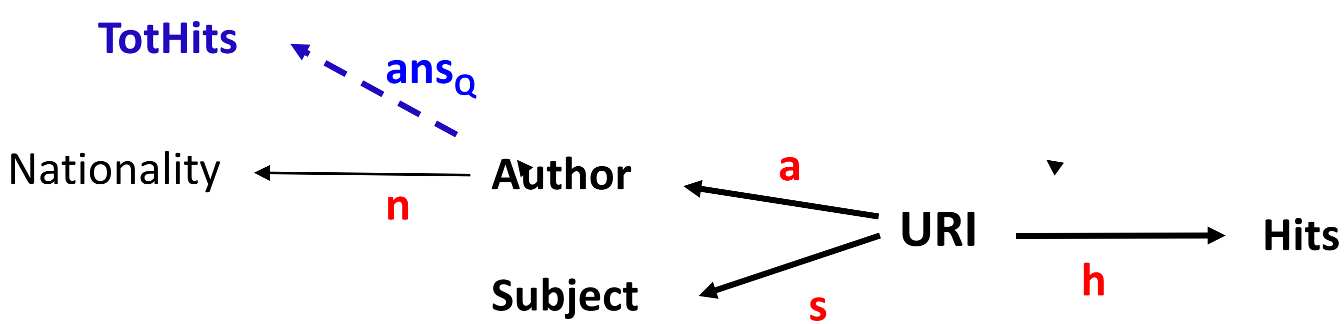
Case 3: ***$f_n \circ \ldots \circ f_1 \preccurlyeq g_1 \wedge \ldots \wedge g_m$ iff $f_n \circ \ldots \circ f_1 \preccurlyeq g_i$ $\forall i$***

Case 4: ***$f_1 \wedge \ldots \wedge f_n \preccurlyeq g_m \circ \ldots \circ g_1$ iff there is sub-pairing s of $f_1 \wedge \ldots \wedge f_n$ s.t. $s \preccurlyeq g_m \circ \ldots \circ g_1$***

assuming c ≼ c' how can we rewrite Q' so that to reuse the result of Q

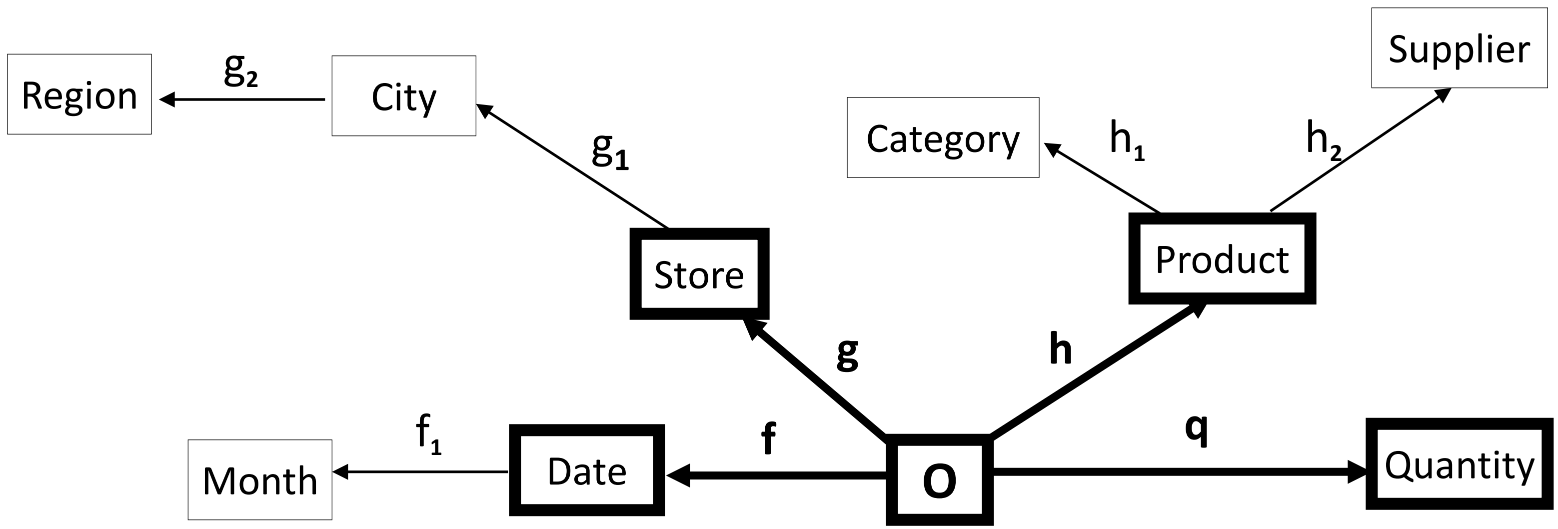
the basic idea:
the answer to a query being a function, it can be used as a measure in any 'larger' query

Ex:
let Q=<a, h, sum> (total hits by author), **Q'=<n∘a, h, sum>** (total hits by nationality)
then Q'= <n∘a, h, sum>= <n, ans_Q, sum> = <n, <a, h, op>, op> (*but notice the change of origin*)



the basic rewriting rule: Q= <e∘e', m, op> = <e, <e', m, op>, op>
(with the understanding that <e', m, op> is to be evaluated before <e∘e', m, op>)

examples

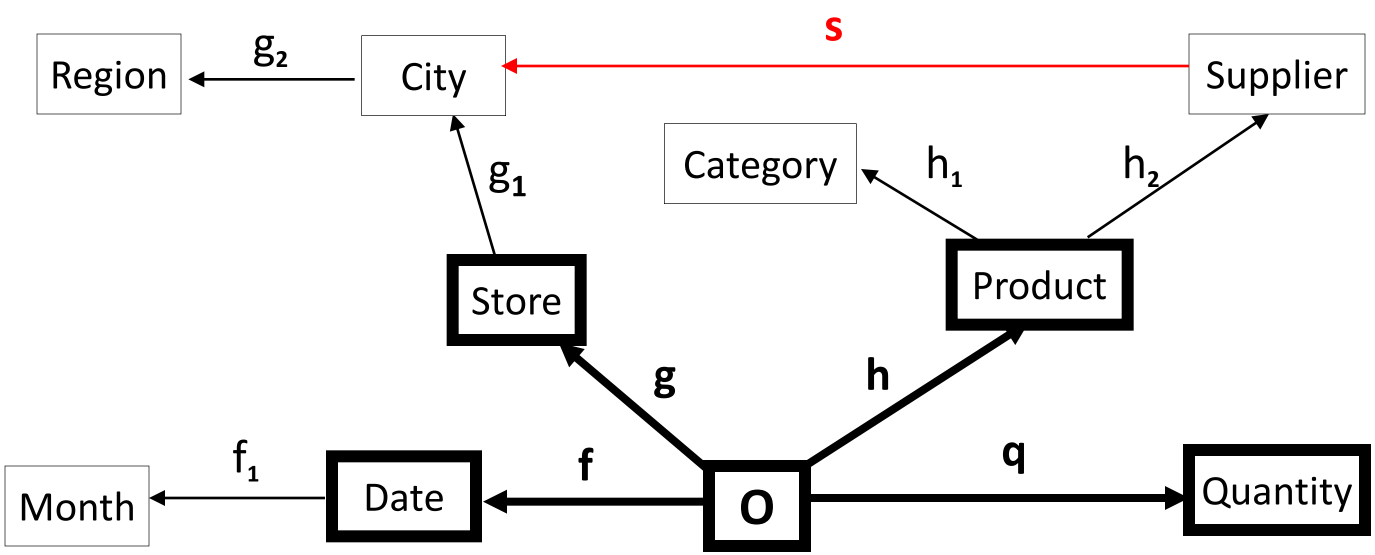


$$< g_2 \circ g_1 \circ g, q, sum> = < g_2, <g_1 \circ g, q, sum>, sum> = < g_2, <g_1, <g, q, sum>, sum>, sum>$$
$$= <g_2 \circ g_1, < g, q, sum>, sum>$$

$$<f, q, sum> = <\pi_f \circ (f \wedge g), q, sum>, sum> = <\pi_f, <f \wedge g, q, sum>, sum> = <\pi_f, <f \wedge h, q, sum>, sum>$$
$$= <\pi_f \circ (f \wedge h), q, sum>, sum> = <\pi_f, <f \wedge h, q, sum>, sum> = <\pi_f, <f \wedge h, q, sum>, sum>$$

→ which of the possible rewritings will be used depends on the contents of the cache

exploiting constraints during rewriting



assume the following integrity constraint: $g_1 \circ g = s \circ h_2 \circ h$ *(suppliers supply only stores in their own city)*
then we have several possibilities of rewritings:

$< g_2 \circ g_1 \circ g, q, sum> = < g_2, <g_1 \circ g, q, sum>, sum> = < g_2, <g_1, <g, q, sum>, sum>, sum> = \ldots\ldots$

$= < g_2, < s \circ h_2 \circ h, q, sum>, sum> = < g_2, < s \circ h_2, <h, q, sum>, sum> = \ldots\ldots$

such constraints can appear as restrictions within a query and not at schema level
(if so, they can be used only for the rewriting of that particular query)

constraints can also be introduced by indicators that depend on two or more attributes

reusing query results - how can we optimize cache management?

if a query Q'= <c', m, op> is submitted to the system then do the following:

if there is query Q= <c, m, op> with c $\preccurlyeq$ c' and ans_Q is in the cache
then begin rewrite Q' as Q"= <h, ans_Q , op>;
evaluate Q'' at target(c);
return $ans_{Q''}$
{in case of multiple, equivalent such Q a choice must be made}
end
else begin evaluate Q'' at source(c');
store $ans_{Q'}$ in the cache;
return $ans_{Q'}$
end

→ **the cache always contains answers of queries that are pair wise incomparable**

concluding remarks

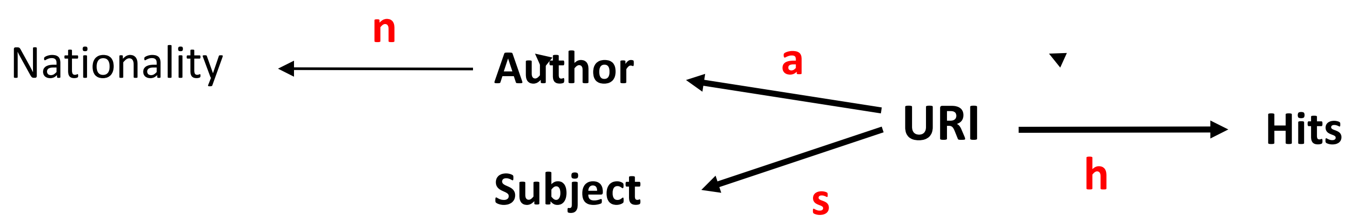
- **the model we have seen can be mapped to relational star schema**

- **limited experimention with the model** (KP-Lab, Assets)

visual formulation of queries

key observation

if the schema is a tree then to formulate an analytic query it is sufficient to give the targets for grouping and measuring plus the aggregate operation



$Q=\langle s \wedge (n \circ a), h, sum \rangle$ can be specified as <{Subject, Nationality}, {Hits}, Sum>

if the schema is represented graphically with clickable nodes then Q can be formulated visually:

click on **Subject;** ***click on*** **Nationality** ***end***
(at this point the system infers the grouping function)
click on Hits ***end***
(at this point the system infers the measuring function and puts the allowed operations in a pop-up menu)
click on **Sum** ***end***
(at this point the system constructs the analytic query)

E

E N D

D