

Big Data Complex Data

A Simple Hard Problem

A Practical Abstract Framework

A Simple Hard Problem

A mutual fund manager needs to measure holdings overlap with other funds

```
Account defineMethod: [ | getHoldingsOverlap |
!lowerPct <- pctEq * 0.8; !upperPct <- ... ;
holdings send: [security].
select: [type isEquity].
collectListElementsFrom: [holdings].
groupedBy: [account].
select: [pctEq >= ^my lowerPct
&& pctEq <= ^my upperPct].
extendBy: [
!xref <- ^my holdings;
!ofactor <- groupList total: [
percentOfPort min: (^my xref at: security.
percentOfPort)
]
].
sortDown: [ofactor]
];
```

A Simple Hard Problem

A mutual fund manager needs to measure holdings overlap with other funds

Is this an application program or a database query?

```
Account defineMethod: [ | getHoldingsOverlap |
!lowerPct <- pctEq * 0.8; !upperPct <- ... ;
holdings send: [security].
select: [type isEquity].
collectListElementsFrom: [holdings].
groupedBy: [account].
select: [pctEq >= ^my lowerPct
&& pctEq <= ^my upperPct].
extendBy: [
!xref <- ^my holdings;
!ofactor <- groupList total: [
percentOfPort min: (^my xref at: security.
percentOfPort)
]
].
sortDown: [ofactor]
];
```

A Simple Hard Problem

A mutual fund manager needs to measure holdings overlap with other funds

Is this an application program or a database query?

This product space query can easily touch large parts of a large database.

```
Account defineMethod: [ | getHoldingsOverlap |
!lowerPct <- pctEq * 0.8; !upperPct <- ... ;
holdings send: [security].
select: [type isEquity].
collectListElementsFrom: [holdings].
groupedBy: [account].
select: [pctEq >= ^my lowerPct
&& pctEq <= ^my upperPct].
extendBy: [
!xref <- ^my holdings;
!ofactor <- groupList total: [
percentOfPort min: (^my xref at: security.
percentOfPort)
]
].
sortDown: [ofactor]
];
```

A Simple Hard Problem

A mutual fund manager needs to measure holdings overlap with other funds

Is this an application program or a database query?

This product space query can easily touch large parts of a large database.

Does the conceptual vocabulary exist in SQL to expose a group's constituents?

```
Account defineMethod: [ | getHoldingsOverlap |
!lowerPct <- pctEq * 0.8; !upperPct <- ... ;
holdings send: [security].
select: [type isEquity].
collectListElementsFrom: [holdings].
groupedBy: [account].
select: [pctEq >= ^my lowerPct
&& pctEq <= ^my upperPct].
extendBy: [
!xref <- ^my holdings;
!ofactor <- groupList total: [
percentOfPort min: (^my xref at: security.
percentOfPort)
]
].
sortDown: [ofactor]
];
```

A Simple Hard Problem

A mutual fund manager needs to measure holdings overlap with other funds

Is this an application program or a database query?

This product space query can easily touch large parts of a large database.

Does the conceptual vocabulary exist in SQL to expose a group's constituents?

Does the conceptual framework exist in either SQL or NoSQL to optimize this?

```
Account defineMethod: [ | getHoldingsOverlap |
!lowerPct <- pctEq * 0.8; !upperPct <- ... ;
holdings send: [security].
select: [type isEquity].
collectListElementsFrom: [holdings].
groupedBy: [account].
select: [pctEq >= ^my lowerPct
&& pctEq <= ^my upperPct].
extendBy: [
!xref <- ^my holdings;
!ofactor <- groupList total: [
percentOfPort min: (^my xref at: security.
percentOfPort)
]
].
sortDown: [ofactor]
];
```

A Simple Hard Problem

A mutual fund manager needs to measure holdings overlap with other funds

Is this an application program or a database query?

This product space query can easily touch large parts of a large database.

Does the conceptual vocabulary exist in SQL to expose a group's constituents?

Does the conceptual framework exist in either SQL or NoSQL to optimize this?

What if this application is run for a collection of accounts?

```
FundUniverse do: [^self getHoldingsOverlap  
first: 10 . do: [...]]
```

```
Account defineMethod: [ | getHoldingsOverlap |  
!lowerPct <- pctEq * 0.8; !upperPct <- ... ;  
holdings send: [security].  
select: [type isEquity].  
collectListElementsFrom: [holdings].  
groupedBy: [account].  
select: [pctEq >= ^my lowerPct  
&& pctEq <= ^my upperPct].  
extendBy: [  
!xref <- ^my holdings;  
!ofactor <- groupList total: [  
percentOfPort min: (^my xref at: security.  
percentOfPort)  
]  
].  
sortDown: [ofactor]  
];
```

A Simple Hard Problem

A mutual fund manager needs to measure holdings overlap with other funds

Is this an application program or a database query?

This product space query can easily touch large parts of a large database.

Does the conceptual vocabulary exist in SQL to expose a group's constituents?

Does the conceptual framework exist in either SQL or NoSQL to optimize this?

What if this application is run for a collection of accounts?

```
FundUniverse do: [^self getHoldingsOverlap  
first: 10 . do: [...]]
```

Or depends on the context dependent fabric of relationships in the data itself?

```
2 yearsAgo evaluate: [MyFund getHoldingsOverlap ...]
```

```
Account defineMethod: [ | getHoldingsOverlap |  
!lowerPct <- pctEq * 0.8; !upperPct <- ... ;  
holdings send: [security].  
select: [type isEquity].  
collectListElementsFrom: [holdings].  
groupedBy: [account].  
select: [pctEq >= ^my lowerPct  
&& pctEq <= ^my upperPct].  
extendBy: [  
!xref <- ^my holdings;  
!ofactor <- groupList total: [  
percentOfPort min: (^my xref at: security.  
percentOfPort)  
]  
].  
sortDown: [ofactor]  
];
```


A Simple Hard Problem

A mutual fund manager needs to measure holdings overlap with other funds

Is this an application program or a database query?

This product space query can easily touch large parts of a large database.

Does the conceptual vocabulary exist in SQL to expose a group's constituents?

Does the conceptual framework exist in either SQL or NoSQL to optimize this?

```
Account defineMethod: [ | getHoldingsOverlap |
!lowerPct <- pctEq * 0.8; !upperPct <- ... ;
holdings send: [security].
select: [type isEquity].
collectListElementsFrom: [holdings].
groupedBy: [account].
select: [pctEq >= ^my lowerPct
&& pctEq <= ^my upperPct].
extendBy: [
!xref <- ^my holdings;
!ofactor <- groupList total: [
percentOfPort min: (^my xref at: security.
percentOfPort)
]
].
sortDown: [ofactor]
];
```

What if this application is run for a collection of accounts?

```
FundUniverse do: [^self getHoldingsOverlap
first: 10 . do: [...]]
```

Or depends on the context dependent fabric of relationships in the data itself?

```
2 yearsAgo evaluate: [MyFund getHoldingsOverlap ... ]
```

Or both !!

A Practical Abstract Framework

The power of the arrow ...



A Practical Abstract Framework

The power of the arrow ...



... and category theory

A Practical Abstract Framework

The power of the arrow ...



... and category theory

A relationship centric view of the world...

A Practical Abstract Framework

The power of the arrow ...



... and category theory

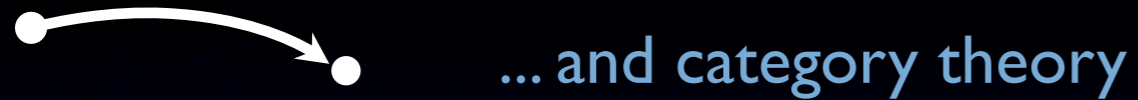
A relationship centric view of the world...

... in which Sets (nodes) are structure-less



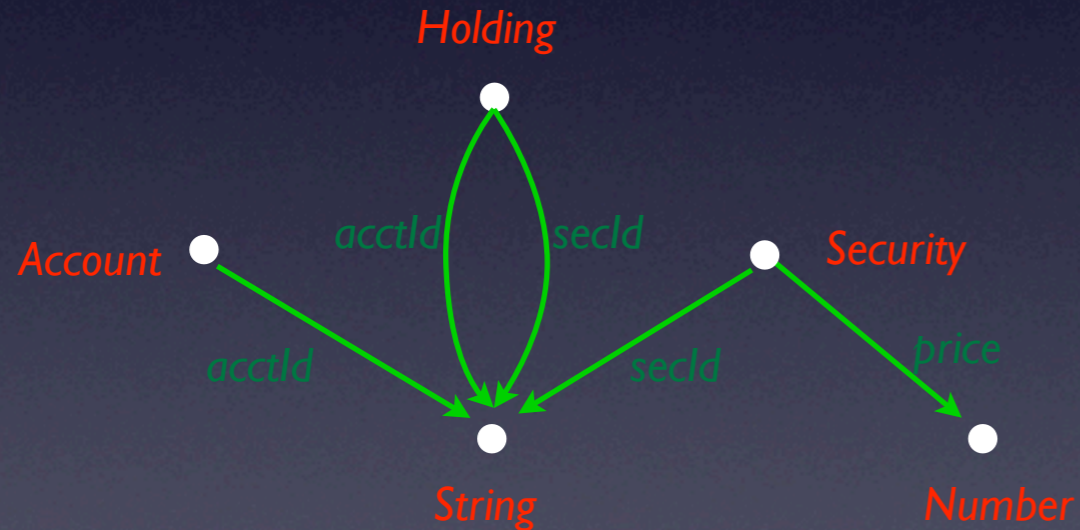
A Practical Abstract Framework

The power of the arrow ...



A relationship centric view of the world...

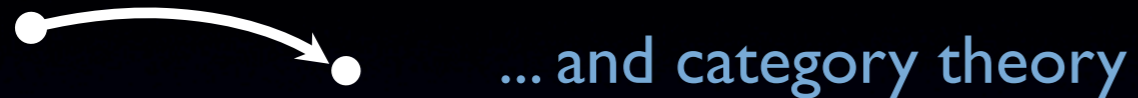
... in which Sets (nodes) are structure-less



... and Functions (arrows) are structure-rich

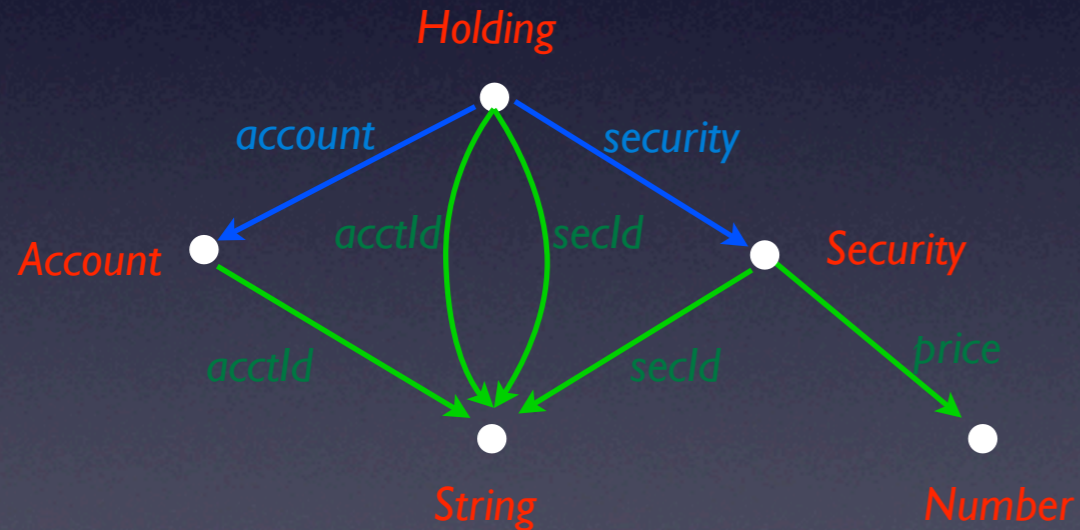
A Practical Abstract Framework

The power of the arrow ...



A relationship centric view of the world...

... in which Sets (nodes) are structure-less

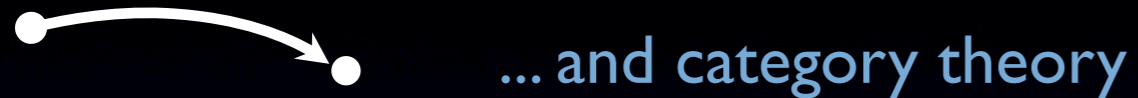


... and Functions (arrows) are structure-rich

... it generalizes the relational model

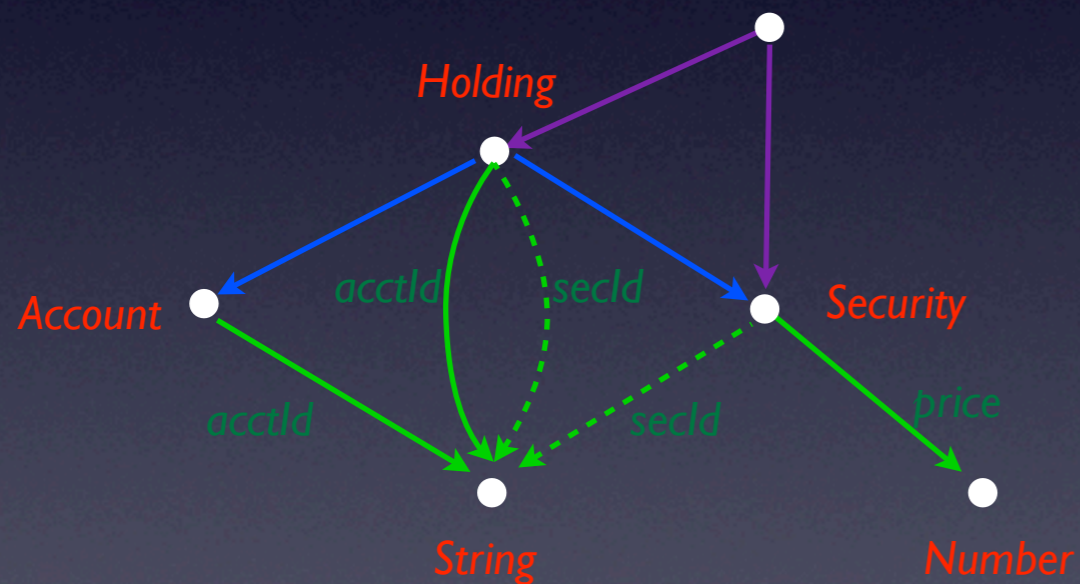
A Practical Abstract Framework

The power of the arrow ...



A relationship centric view of the world...

... in which Sets (nodes) are structure-less



... and Functions (arrows) are structure-rich


... it generalizes the relational model

... and models its computations (e.g. *join*)

A Practical Abstract Framework

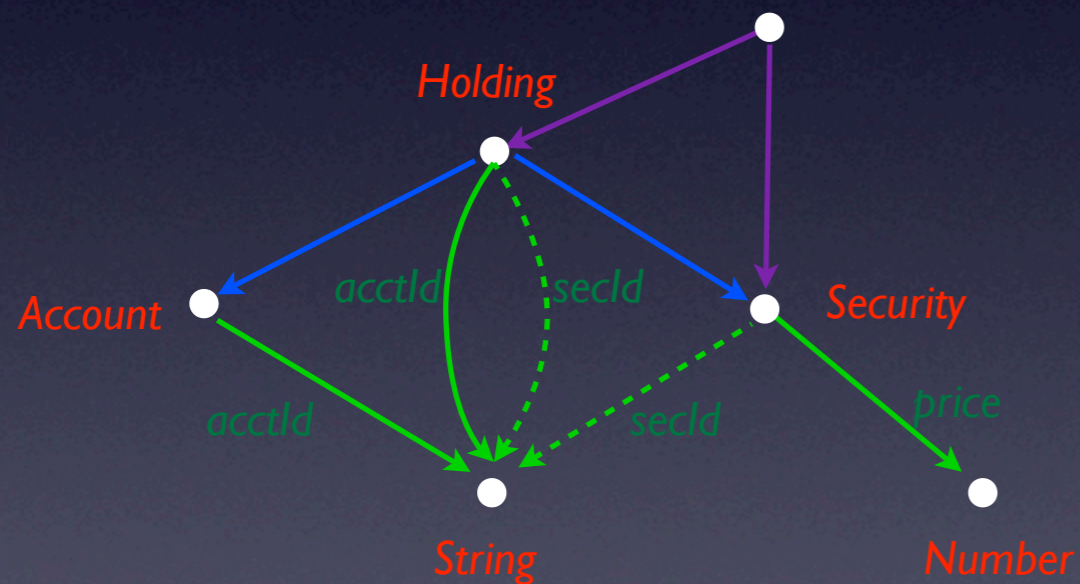
The power of the arrow ...

Something else to consider...

 ... and category theory

A relationship centric view of the world...

... in which Sets (nodes) are structure-less



... and Functions (arrows) are structure-rich

... it generalizes the relational model

... and models its computations (e.g. join)

A Practical Abstract Framework

The power of the arrow ...



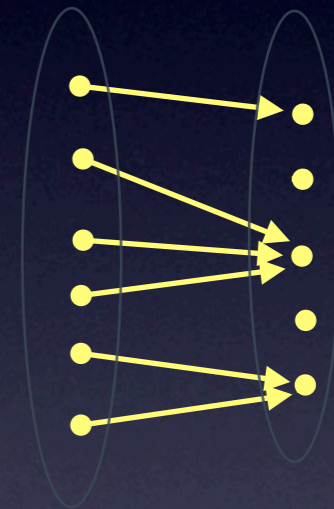
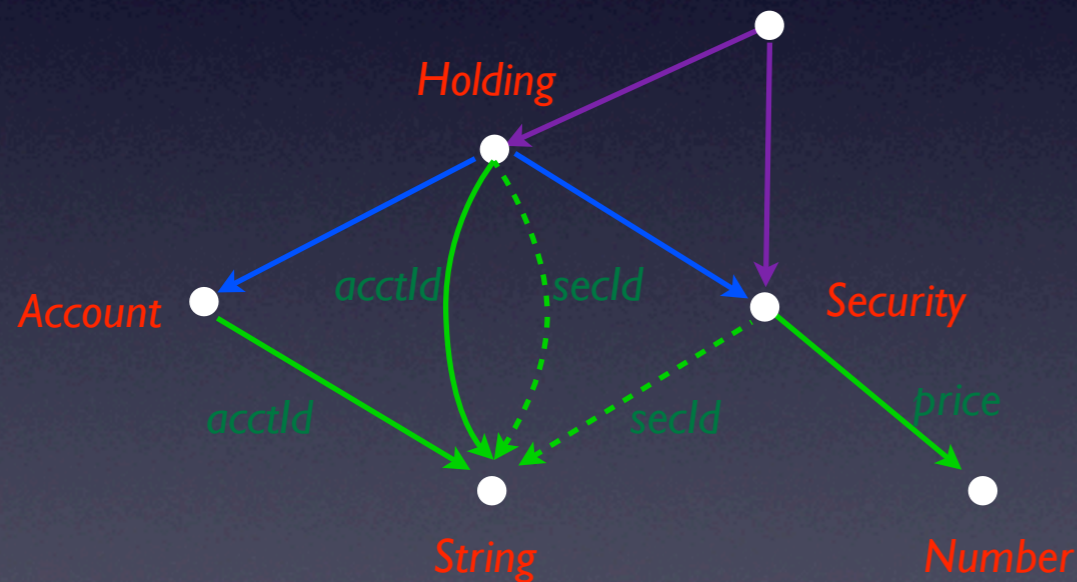
... and category theory

Something else to consider...

If reading an arrow in the natural (forward) direction models a functional relationship, what does reading it in reverse model?

A relationship centric view of the world...

... in which Sets (nodes) are structure-less



... and Functions (arrows) are structure-rich

... it generalizes the relational model

... and models its computations (e.g. join)

A Practical Abstract Framework

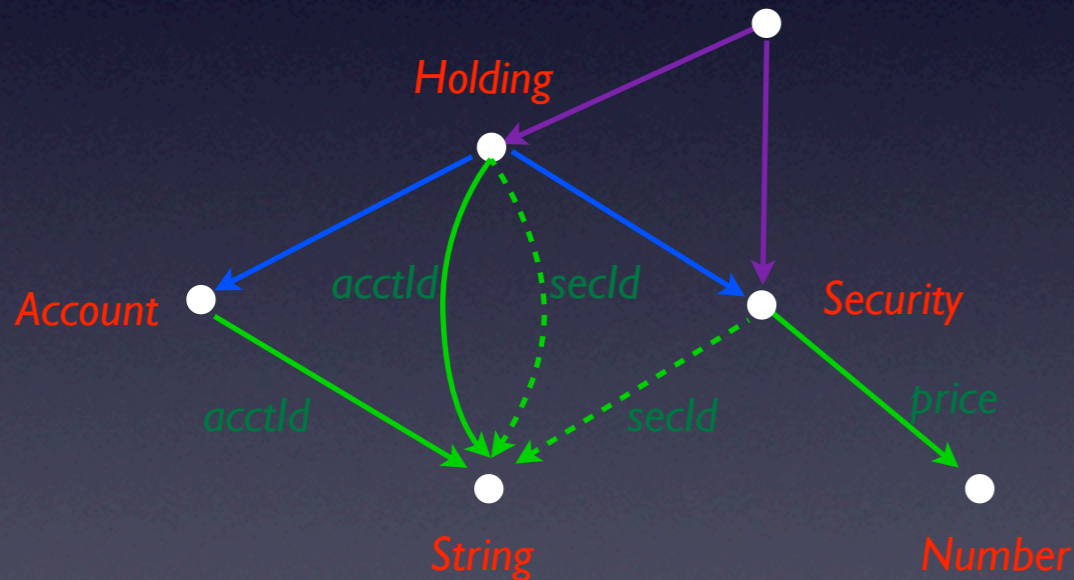
The power of the arrow ...



... and category theory

A relationship centric view of the world...

... in which Sets (nodes) are structure-less



Something else to consider...

If reading an arrow in the natural (forward) direction models a functional relationship, what does reading it in reverse model?



Collections !!!

... and Functions (arrows) are structure-rich

... it generalizes the relational model

... and models its computations (e.g. join)

A Practical Abstract Framework

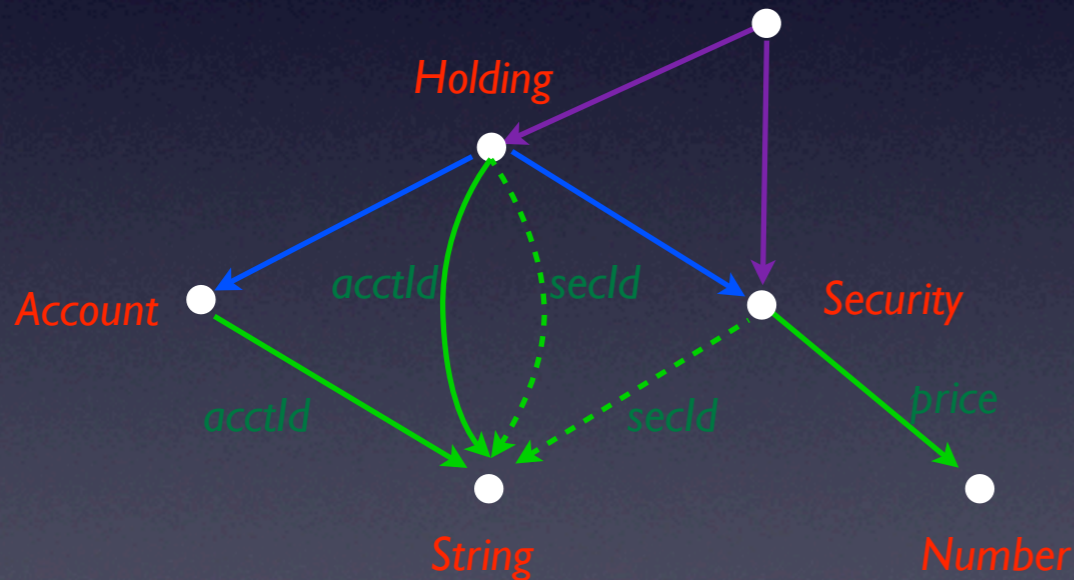
The power of the arrow ...



... and category theory

A relationship centric view of the world...

... in which Sets (nodes) are structure-less



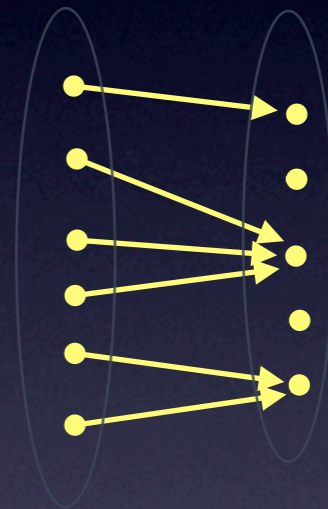
... and Functions (arrows) are structure-rich

... it generalizes the relational model

... and models its computations (e.g. *join*)

Something else to consider...

If reading an arrow in the natural (forward) direction models a functional relationship, what does reading it in reverse model?



Collections !!!

A proven, powerful, unifying theory for data base programming.

Intrigued?