[/Band|c00t]

The Bandicoot Language

code reuse for the relational model

Ostap Cherkashin, Julius Chrobak

Emerging Languages Camp 2012

Project Bandicoot

- set based programming system
- improve the interface to the relational model
- new language and runtime

Use Case

Two CSV files:

books

title	**genre**	**pages**	**price**
Foo	fiction	100	120.00
Bar	poetry	2000	190.00
Xyz	poetry	2000	50.50

discounts

title	**rate**
Foo	0.90
Bar	0.80

We want to:

- apply discount and find all books with a price greater than 100.0$
- list the genres of those books

Solution

```
$ bandicoot start -p 80 -d volume -s state -c program.b

$ curl --data-binary @books.csv http://localhost/Store

$ curl --data-binary @discounts.csv http://localhost/Expensive
title string,genre string,pages int,price real,newPrice real
Foo,fiction,100,120.00,108.00
Bar,poetry,2000,190.00,152.00

$ curl --data-binary @discounts.csv http://localhost/Genres
genre string
fiction
poetry
```

Solution

```
var books {
    title string,
    genre string,
    pages int,
    price real,
};

fn Store(b) {
    books = b;
}

fn Expensive(discount) {
    return select (newPrice > 100.0)
                  (extend (newPrice = price * rate)
                          (join shelf discount));
}

fn Genres(discount) {
    return project (genre) (Expensive discount);
}
```

Variables

```
var books {
    title string,
    genre string,
    pages int,
    price real,
}; # this is a SET
```

```
title |   genre | pages |  price
  Foo | fiction |   100 | 120.00
  Bar |  poetry |  2000 | 190.00
  Xyz |  poetry |  2000 |  50.50
```

User Defined Functions

```
fn Store(b) {
    books = b;
}

fn Expensive(discount) {
    return select (newPrice > 100.0)
                  (extend (newPrice = price * rate)
                          (join books discount));
}

fn Genres(discount) {
    return project (genre) (Expensive discount);
}


$ curl --data-binary @books.csv http://localhost/Store
```

Built-In Operators

Unary

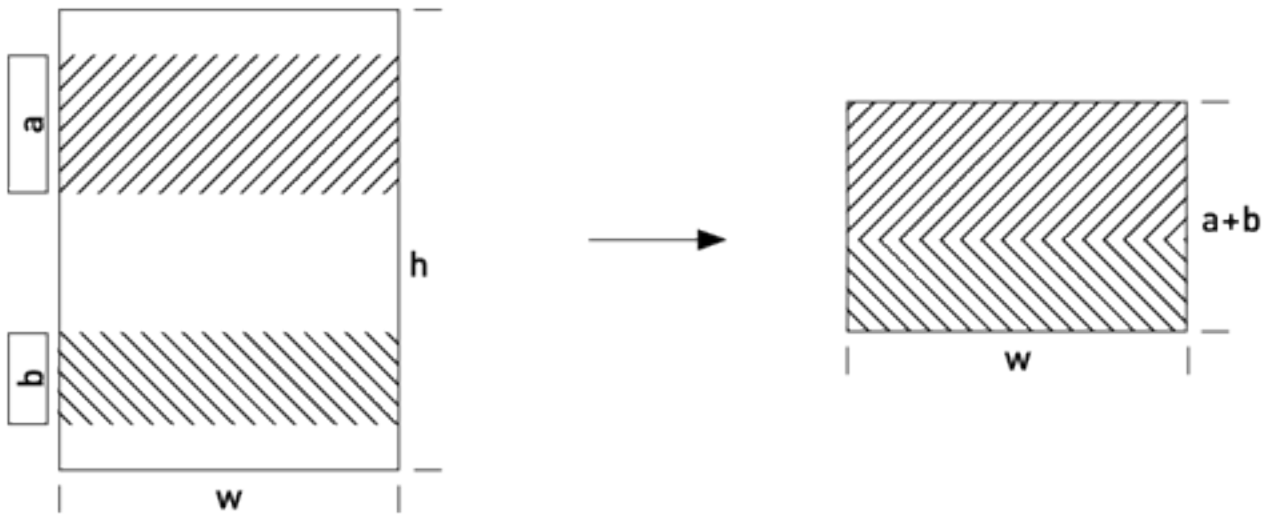
- select
- project
- extend
- rename

Binary

- join
- union
- minus
- summary

http://bandilab.org/specification.html

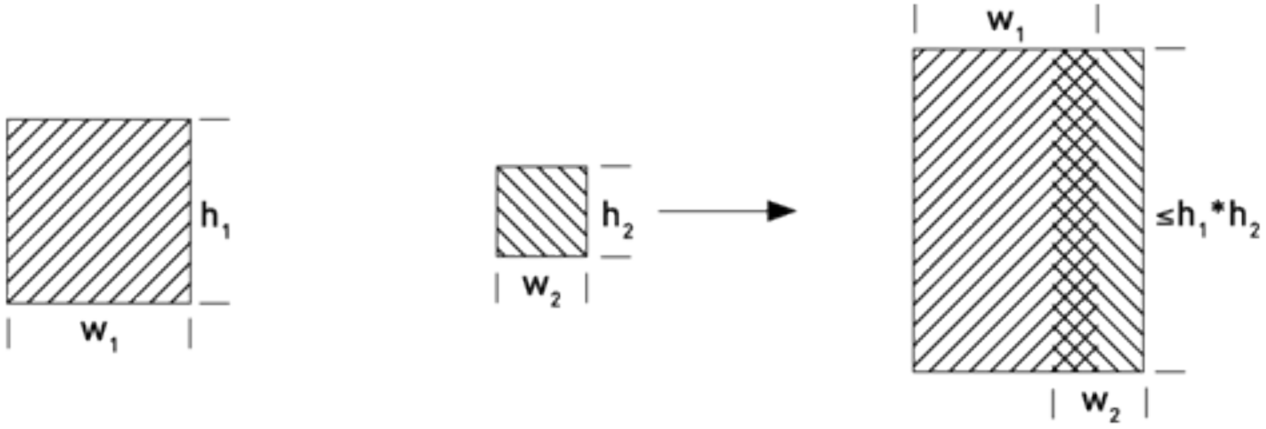
Select



```
select (price > 100.0) books
```

```
title |   genre | pages |  price         title |   genre | pages |  price
  Foo | fiction |   100 | 120.00   ->     Foo | fiction |   100 | 120.00
  Bar |  poetry |  2000 | 190.00           Bar |  poetry |  2000 | 190.00
  Xyz |  poetry |  2000 |  50.50
```

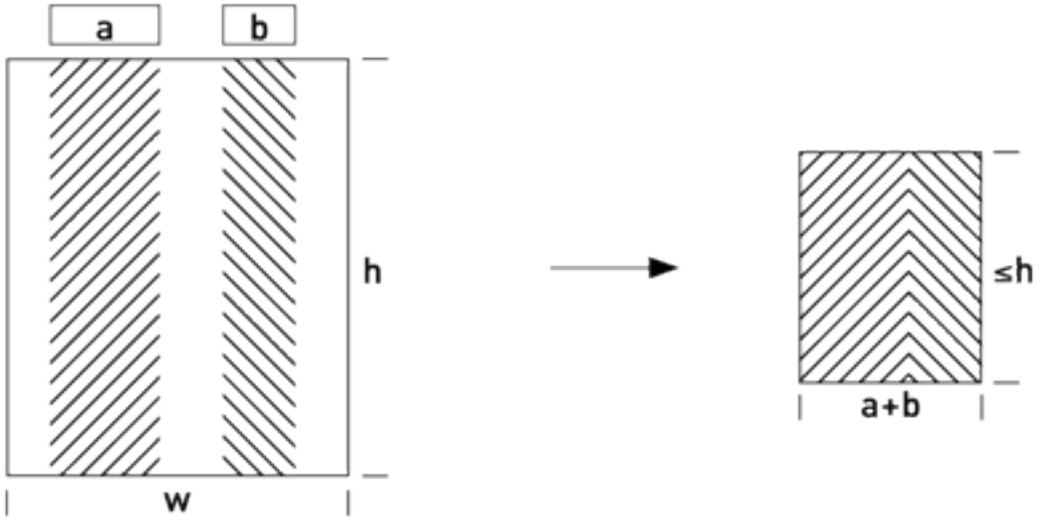
Join



```
join books discount
```

```
title |  price | ...    title | rate       title |  price | rate | ...
  Foo | 120.00 |      *   Foo | 0.90   ->   Foo | 120.00 | 0.90 |
  Bar | 190.00 |          Bar | 0.80          Bar | 190.00 | 0.80 |
  Xyz |  50.50 |
```

Project



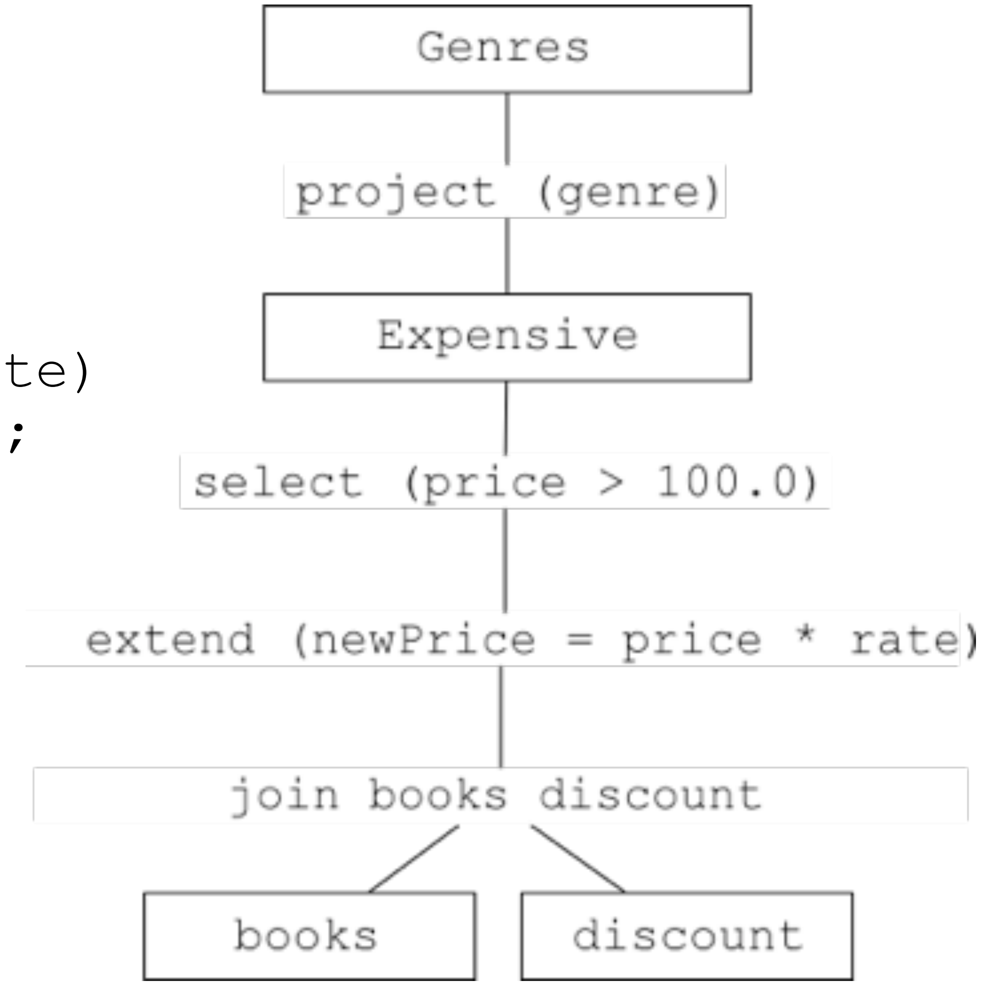
```
project (genre, pages) shelf
```

```
title |   genre | pages |  price          genre | pages
  Foo | fiction |   100 | 120.00   ->  fiction |   100
  Bar |  poetry |  2000 | 190.00        poetry |  2000
  Xyz |  poetry |  2000 |  50.50
```

Code Reuse

```
fn Expensive(discount) {
    return select (newPrice > 100.0)
                  (extend (newPrice = price * rate)
                          (join shelf discount));
}

fn Genres(discount) {
    return project (genre) (Expensive discount);
}
```



Let’s Recap

- variables
- operators
- functions
- everything is a relation

Attribute Sets

```
type BookId { title string }
type Books { BookId, genre string, price real }
type Discounts { BookId, rate real }

var books Books;
```

- relations are all about attributes
- type composition

Modules

```
fn UpdateStock() {
    books = discounts.Apply books;
}
```

- group things together
- hide complexity
- simplify collaboration

Collaborate

- some of the presented features are in development
- source code: http://github.com/bandilab
- project web site: http://bandilab.org
- try on-line at http://mingle.io