

INF1343, Winter 2012

Data Modeling and Database Design

Yuri Takhteyev

Faculty of Information
University of Toronto



This presentation is licensed under Creative Commons Attribution License, v. 3.0. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/>. This presentation incorporates images from the Crystal Clear icon collection by Everaldo Coelho, available under LGPL from <http://everaldo.com/crystal/>.

Week 12

Storage, Structure,
and Performance

The Final Project

1. Congratulation!

2. Moving Forward

- <https://github.com/yuri/padawan>
- Try Python's MySQLdb module
- Python web frameworks
 - Object-relational mappers (e.g., with Django)

The Final Exam

Time & Place:

Room 538 (5th floor of Bissel,
requires code)

Scope

- Drawing ER diagrams
- Implementing ER design as SQL
- SQL queries
 - including joins and grouping
- Miscellaneous
 - see website for sample questions

Storage

Persistent

Easy to read

Easy to update

Cost-effective

Fragment of an ancient stone inscription with multiple lines of text in an unknown script, possibly cuneiform or a similar ancient writing system. The text is arranged in approximately 15 horizontal lines across the fragment. The script is highly stylized and densely packed. The fragment is dark and shows signs of weathering and age.

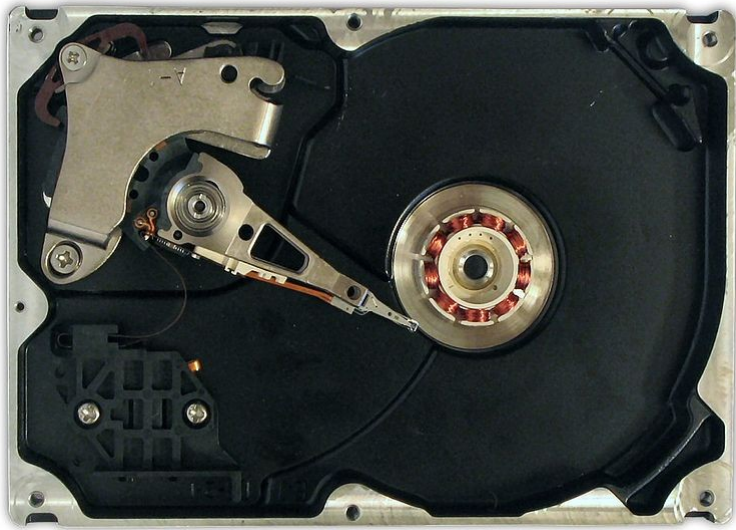
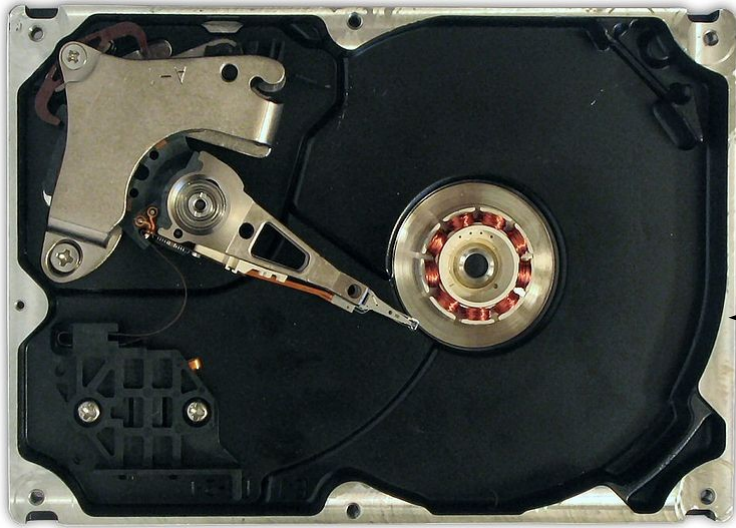


image sources

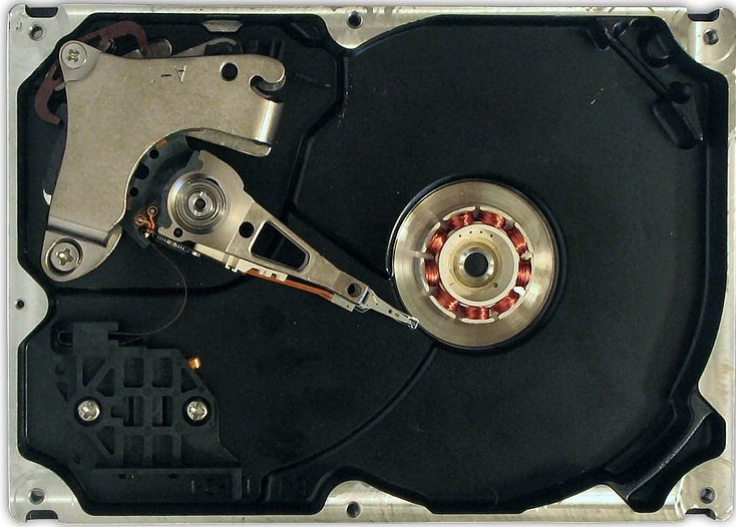
http://en.wikipedia.org/wiki/File:Hard_disk_dismantled.jpg

http://en.wikipedia.org/wiki/File:Memory_module_DDRAM_20-03-2006.jpg

<http://en.wikipedia.org/wiki/File:Targetape.jpg>



caching



“hierarchical
storage
management”

Structure

Finding a “storable” representation

- Not losing information
- Allowing for easy retrieval
- Allowing for easy update
- Minimizing storage size

CSV

simple!

relatively small

but what to do with NULLs?

and how would it perform?

CSV

657807938,559562982,23.47

755280276,889208095,590.32

934625720,459538801,44.66

852067113,660539228,1684.77

+ another billion rows

CSV

657807938,559562982,23.47 ↩

755280276,889208095,590.32 ↩

934625720,459538801,44.66 ↩

852067113,660539228,1684.77 ↩

+ another billion rows

CSV

657807938,559562982,23.
47↵755280276,88920809
5,590.32↵934625720,459
538801,44.66↵852067113
,660539228,1684.77↵.....

Fixed-Length

657807938	559562982	002347
755280276	889208095	059032
934625720	459538801	004466
852067113	660539228	168477

+ another billion rows

start of Nth row = $N * \text{length of row}$

Fixed-Length

657807938	559562982	002347
755280276	889208095	059032
934625720	459538801	004466
852067113	660539228	168477

+ another billion rows

How do we **add** a row?

How do we **delete** a row?

How do we **find** a row?

Sort it?

755280270029178799023934

755280276889208095059032

755280279890089234000020

755290123939191721000129

+ another billion rows

Finding is easier!

Inserting is **harder!**

Complexity Analysis

“Big-O” analysis

Best sorting methods: $O(n \log(n))$

Searching a list: $O(n)$

Searching a sorted list: $O(\log(n))$

Inserting into a sorted list: $O(n)$

And the second field?

755280270029178799023934

755280276889208095059032

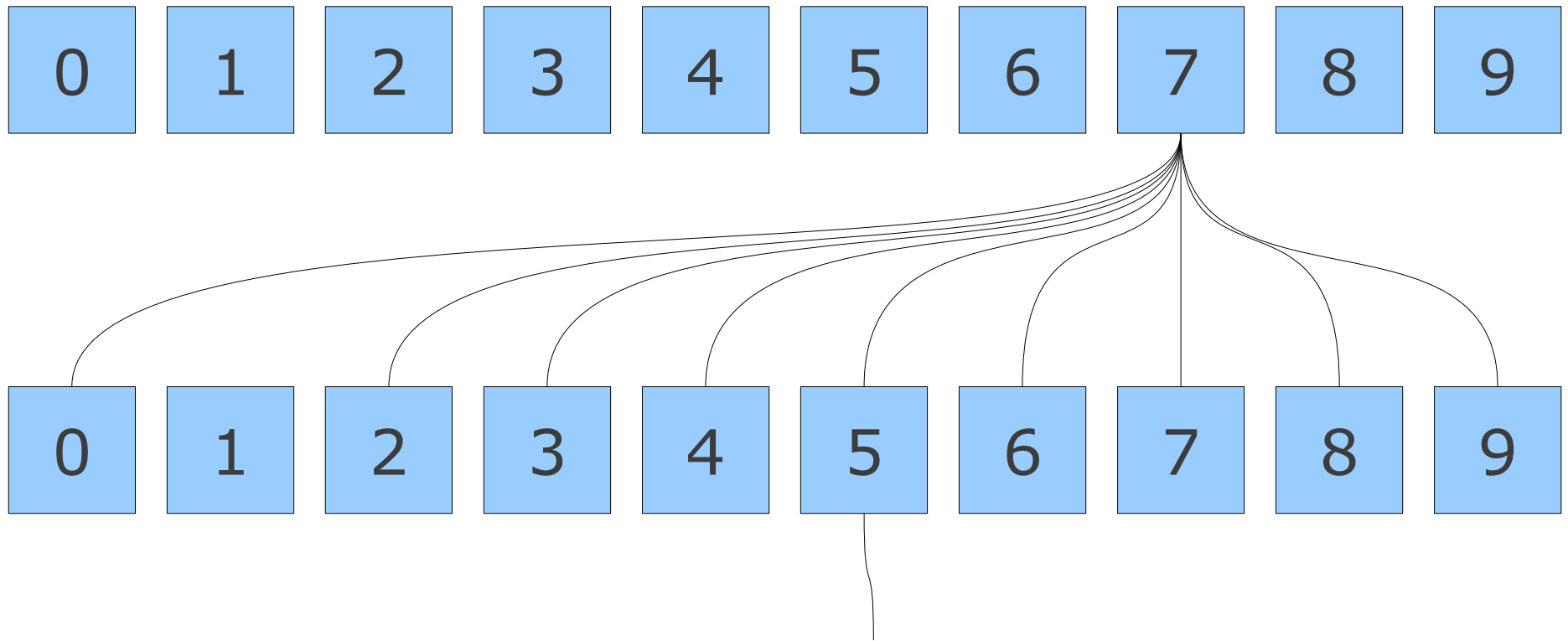
755280279890089234000020

755290123939191721000129

+ another billion rows

Sort by the 2nd field?
But avoid duplication!
(Essentially an index.)

Trees



755280276,889208095,590.32
and other items that start with 75
(does not need to be ordered)

Fixed vs Balanced

Fixed:

E.g., ISAM

Balanced:

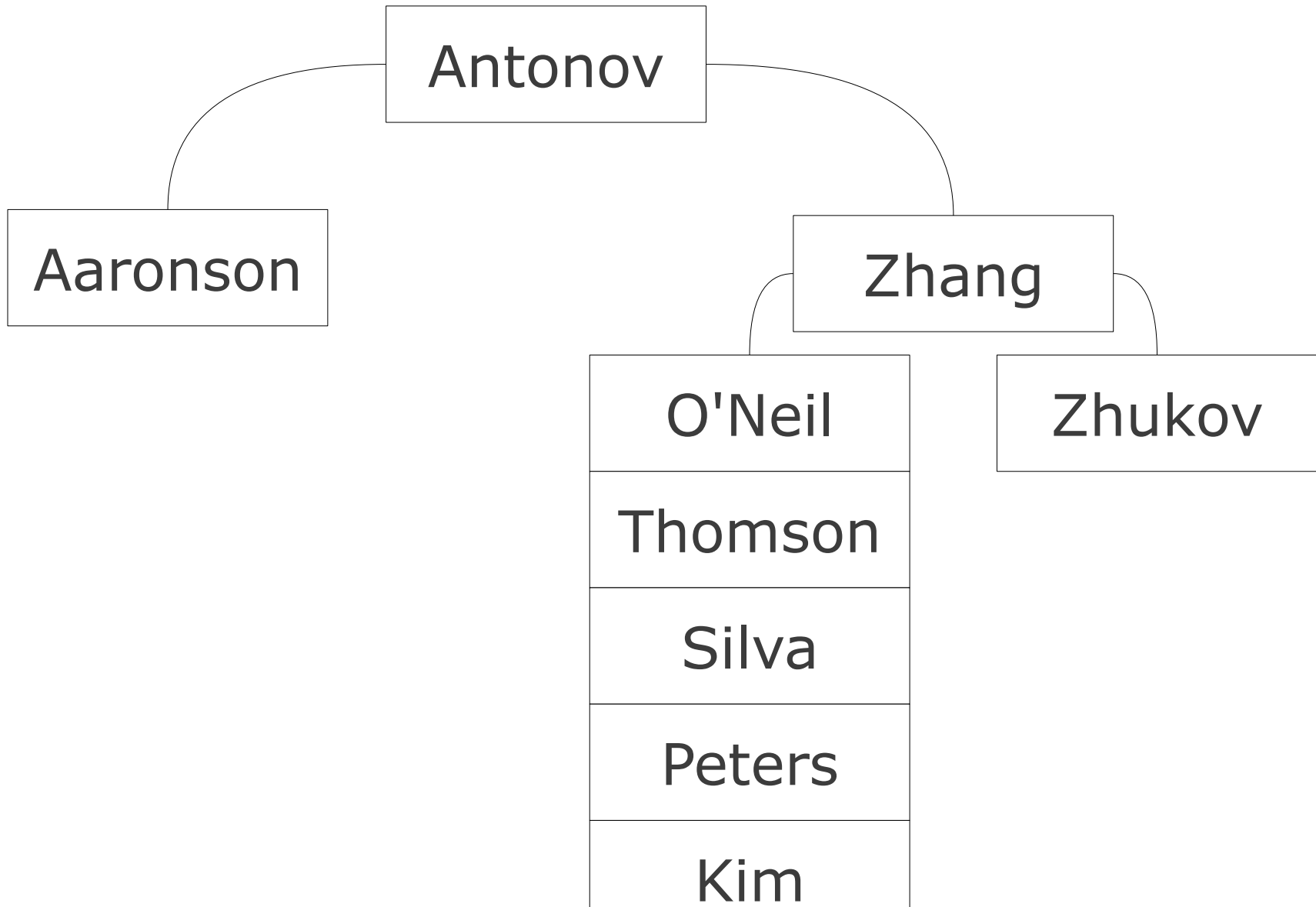
E.g., B+ Trees

A: 4%

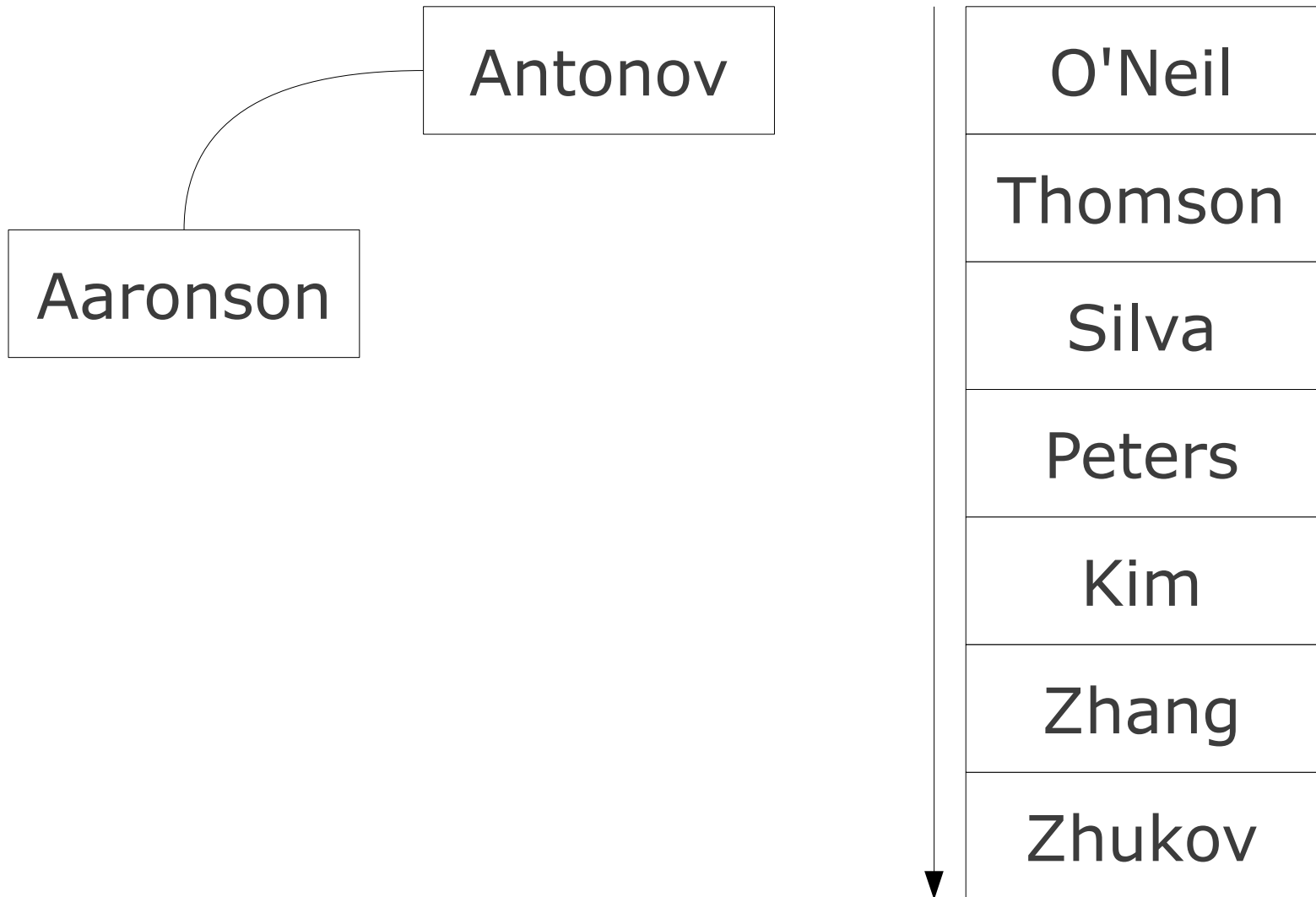
M: 9.5%

Q: 0

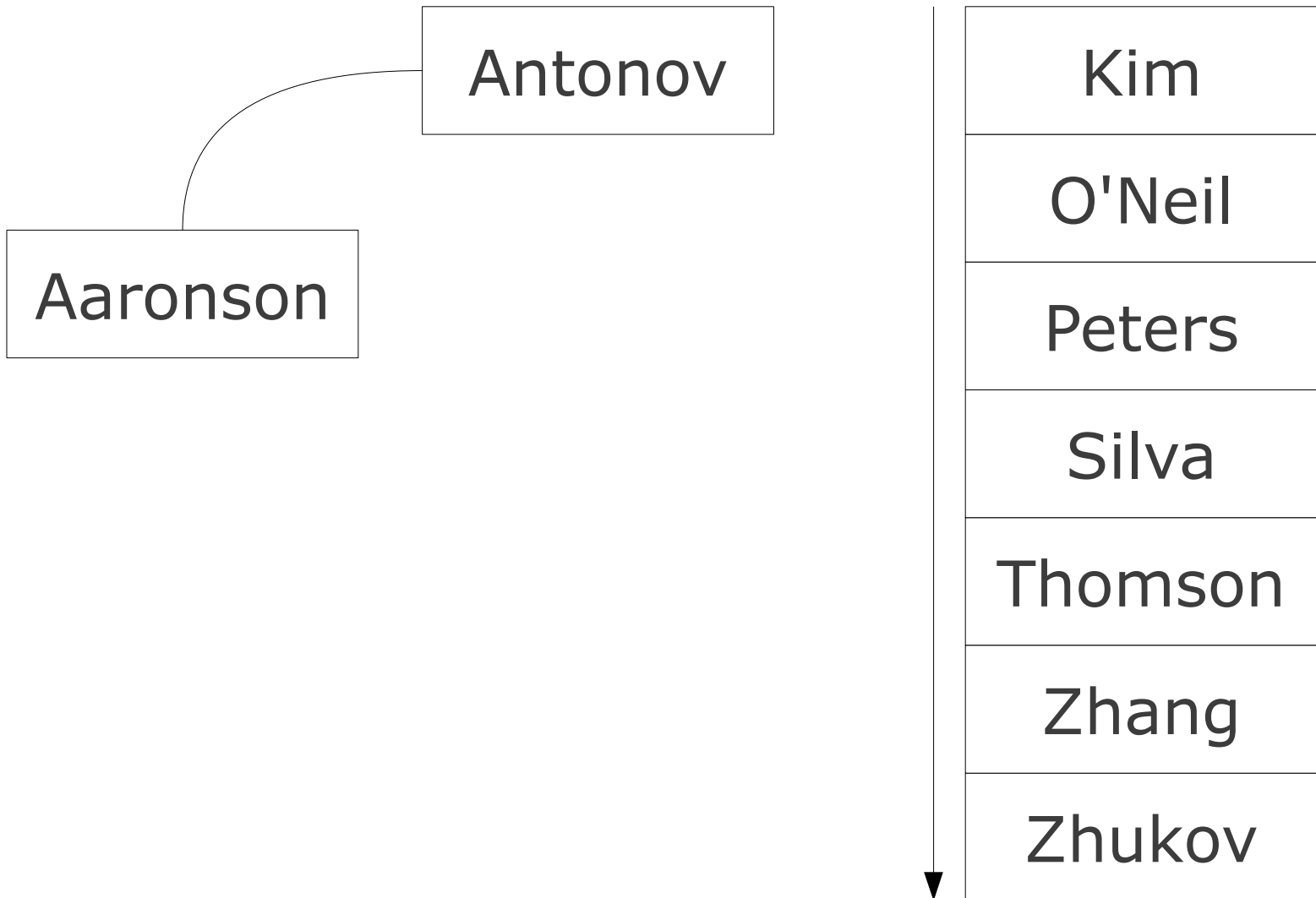
Balanced Tree



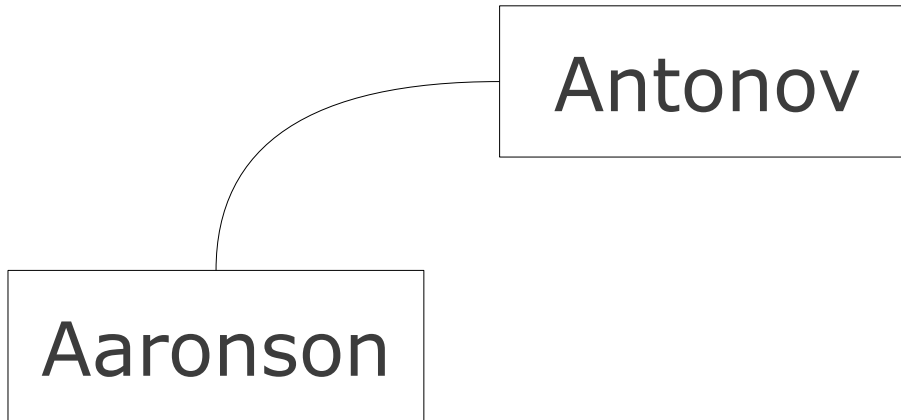
Balanced Tree



Balanced Tree

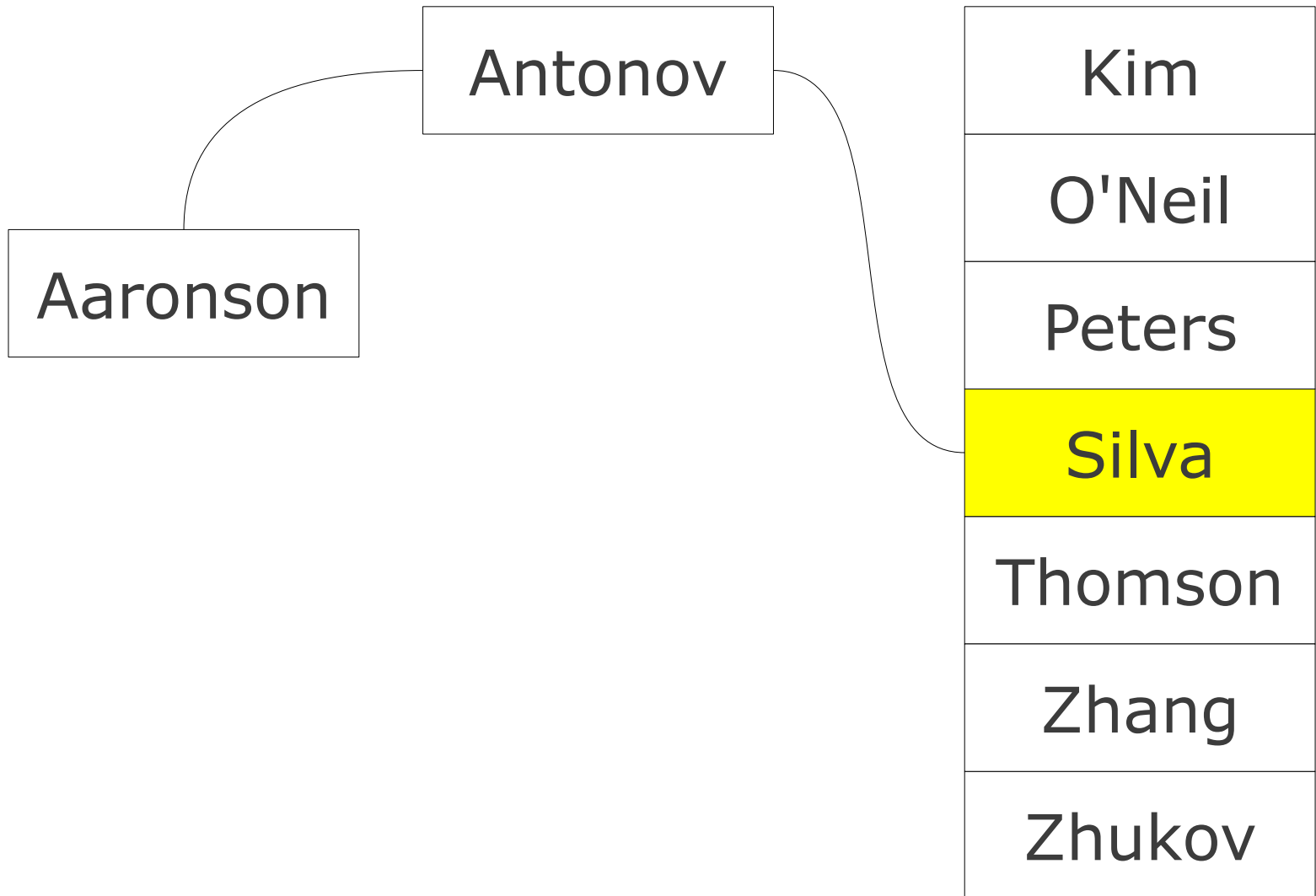


Balanced Tree

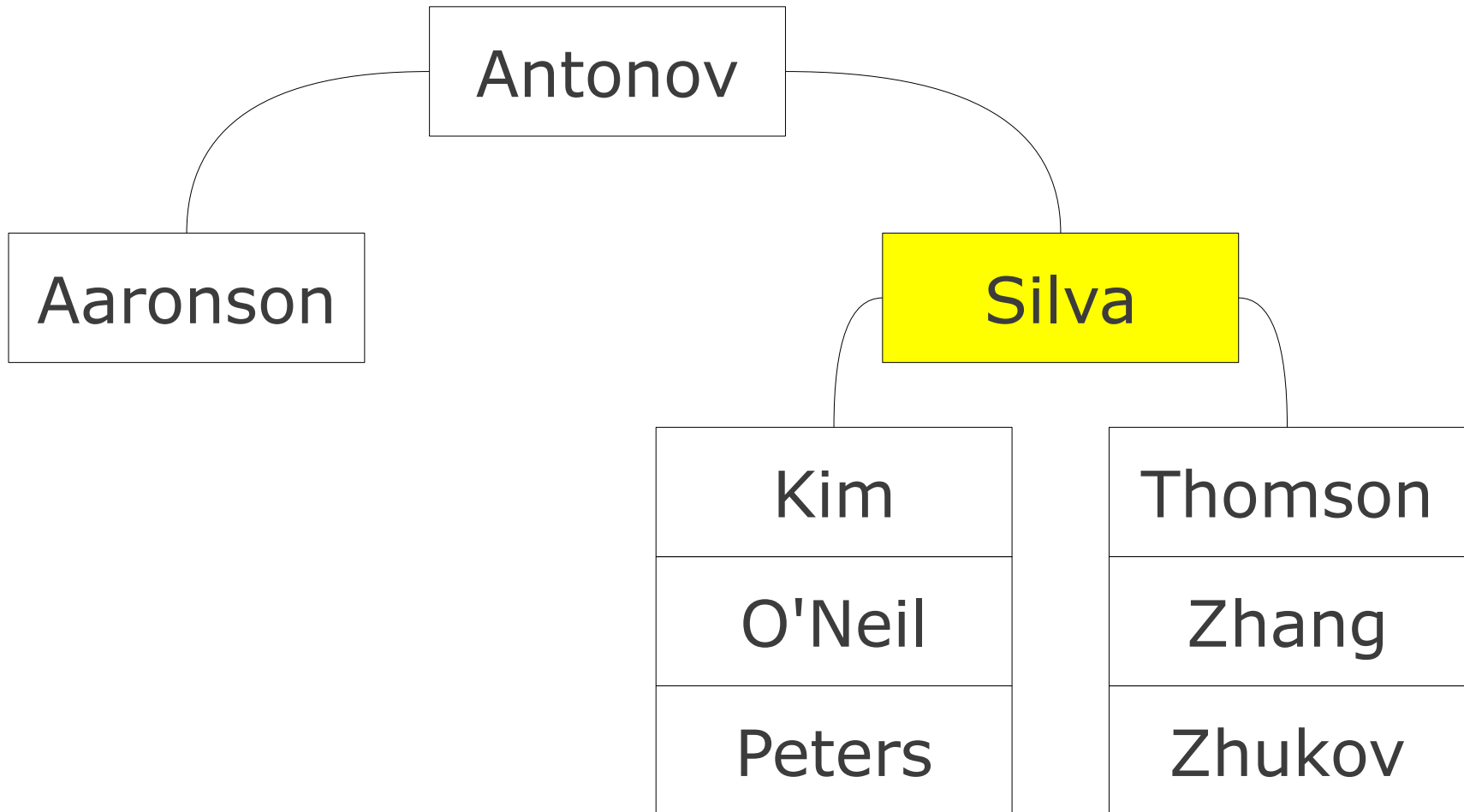


Kim
O'Neil
Peters
Silva
Thomson
Zhang
Zhukov

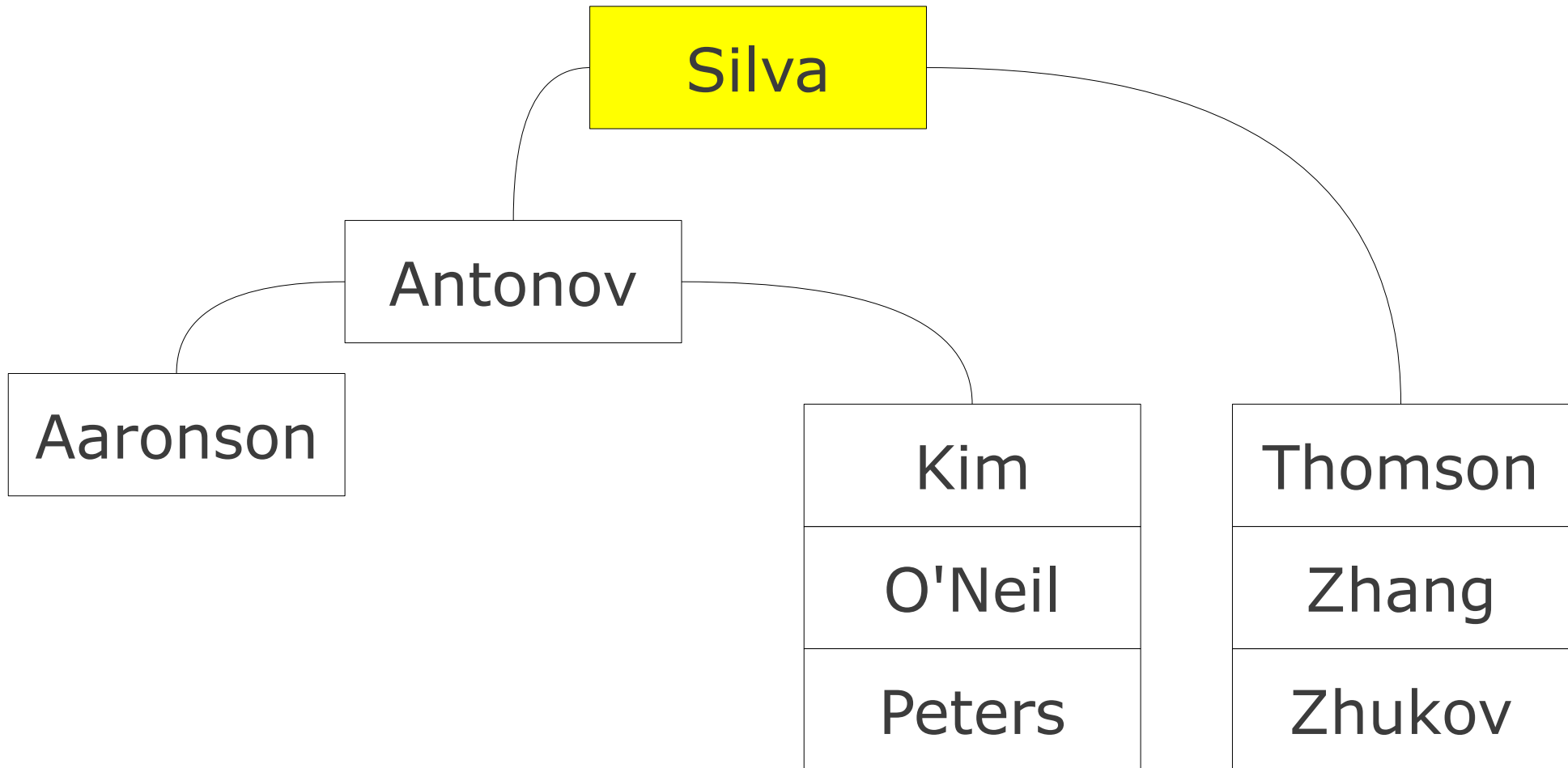
Balanced Tree



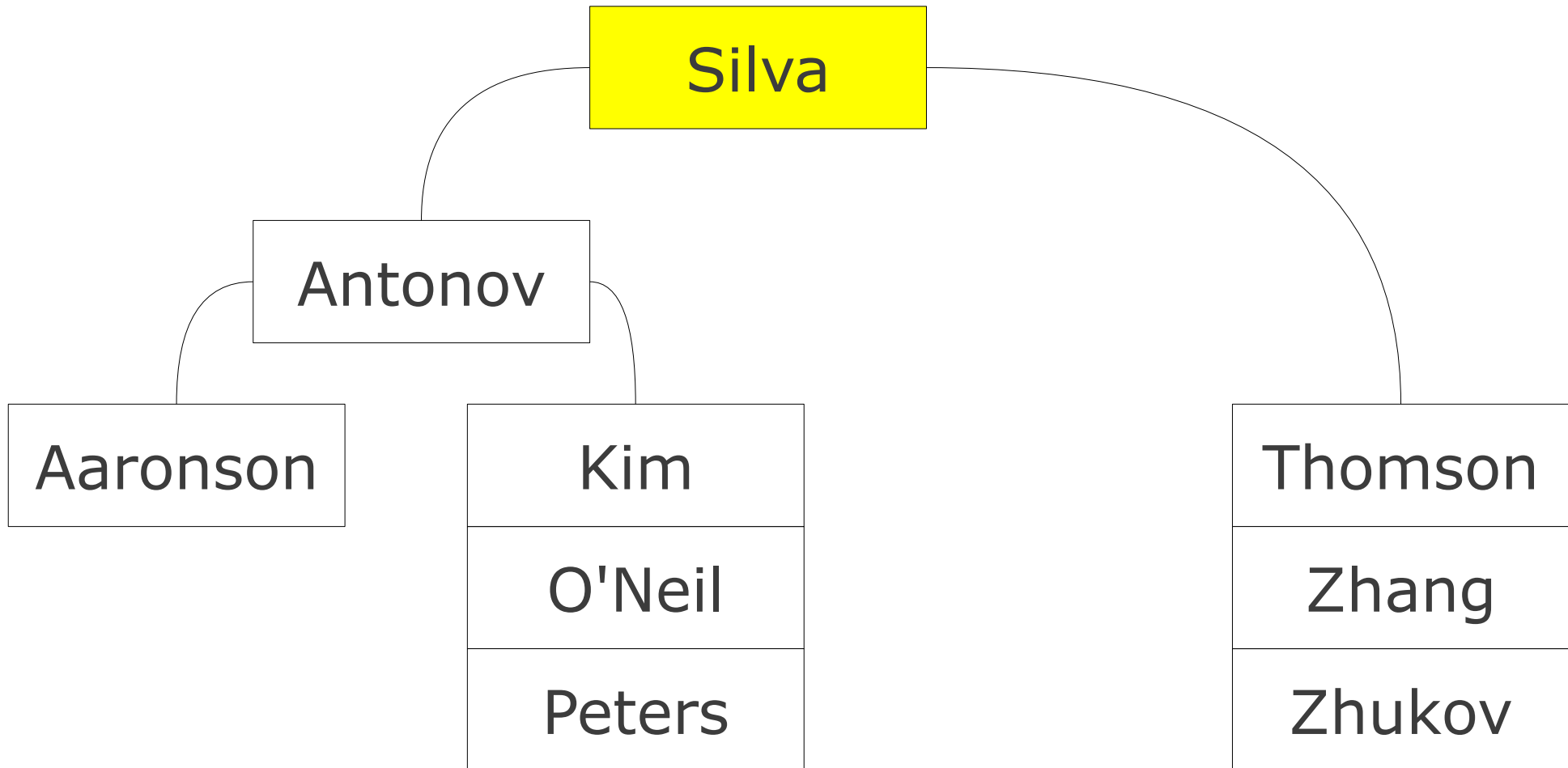
Balanced Tree



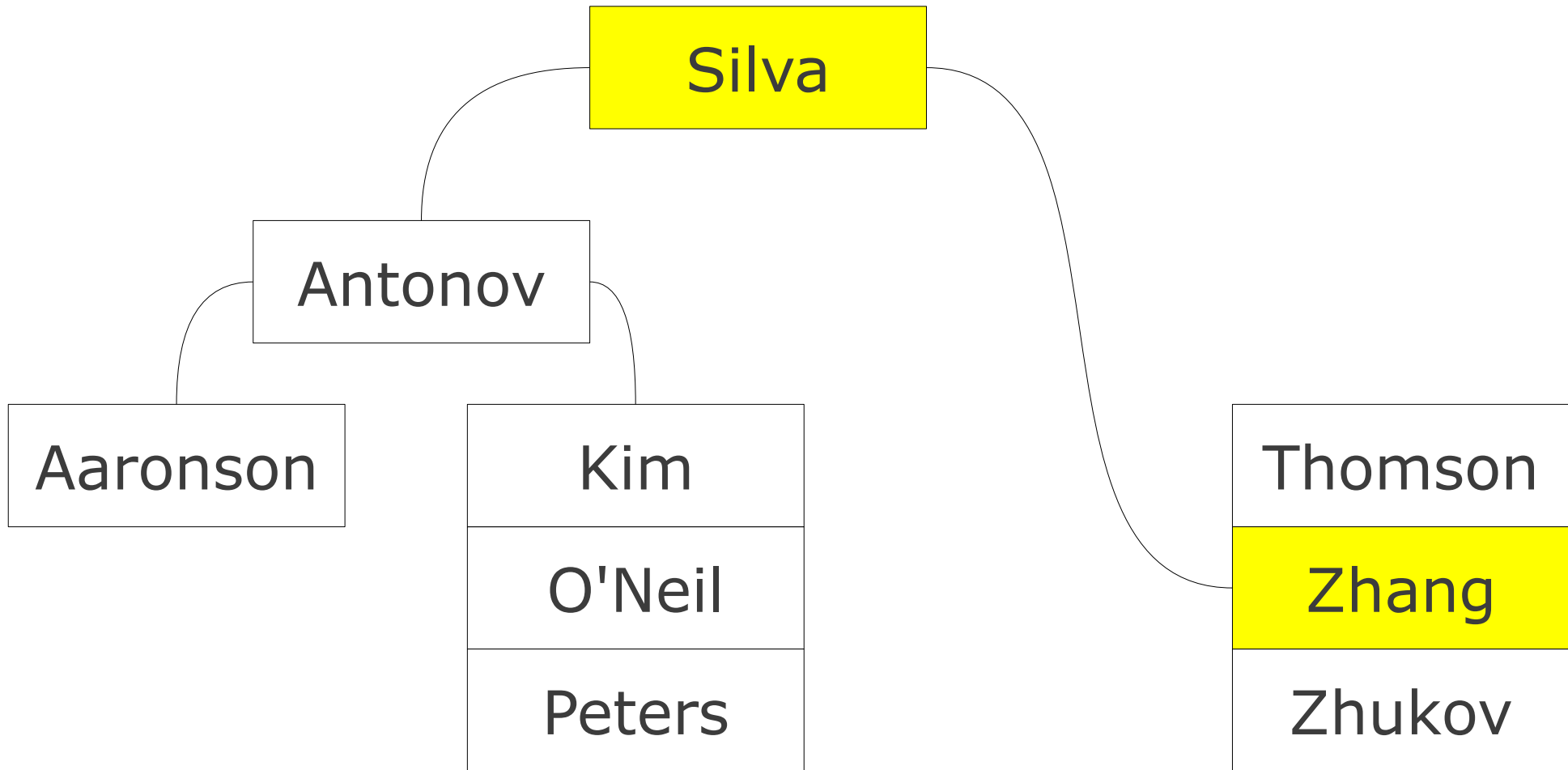
Balanced Tree



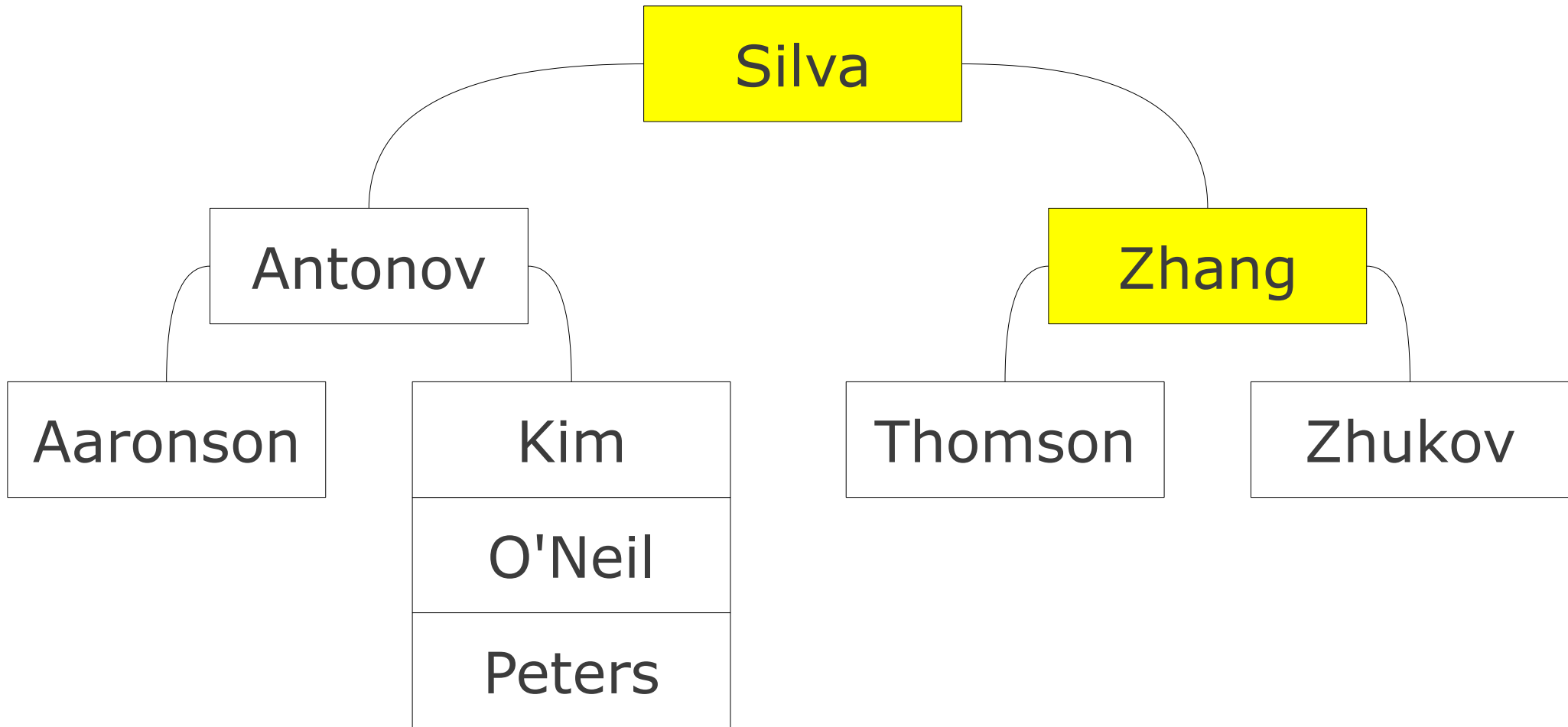
Balanced Tree



Balanced Tree



Balanced Tree



Storage Engines

CVS

yes, it's an option

MyISAM

the default before MySQL 5.5
relatively simple

InnoDB

more features
the new default (since 5.5)

The Extra Features

Foreign Key Constraints
not in MyISAM!

Transactions

```
start transaction;  
update table1 ...;  
update table2 ...;  
commit;
```

Downside: complexity

More Engines

Memory

no harddrive → faster

Blackhole

doesn't actually store anything

Federated

allows remote tables

Etc.

Picking an Engine

```
create table superhero (  
  id integer,  
  primary key (id),  
  name char(100)  
) engine=MyISAM;
```

Creating an Index

Easy retrieval by field

- usually automatic for PK
- optional for other fields
- downsides:
 - additional space
 - harder inserts

```
create index username_index  
on user(username);
```

Distributed Performance Revisited

Distributing the work:

one big machine

vs. many small ones

Some things are easier to split

Distributing RDBMS is hard

(hence the interest in “NoSQL”)

Questions?

(Anything from this semester.)