INF1343, Winter 2012

Data Modeling and Database Design

Yuri Takhteyev
Faculty of Information
University of Toronto

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

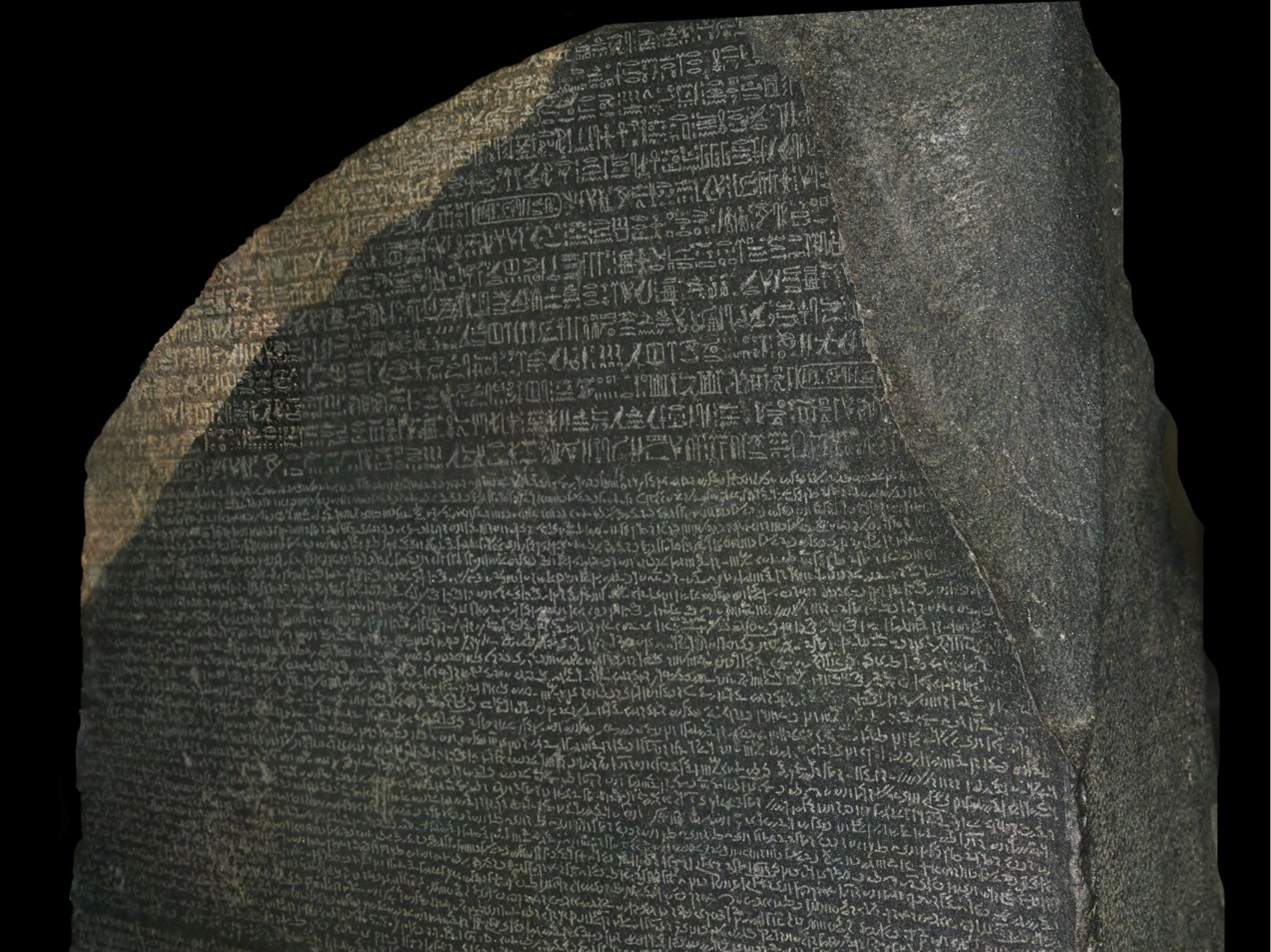
Storage

Persistent

Easy to read

Easy to update

Cost-effective



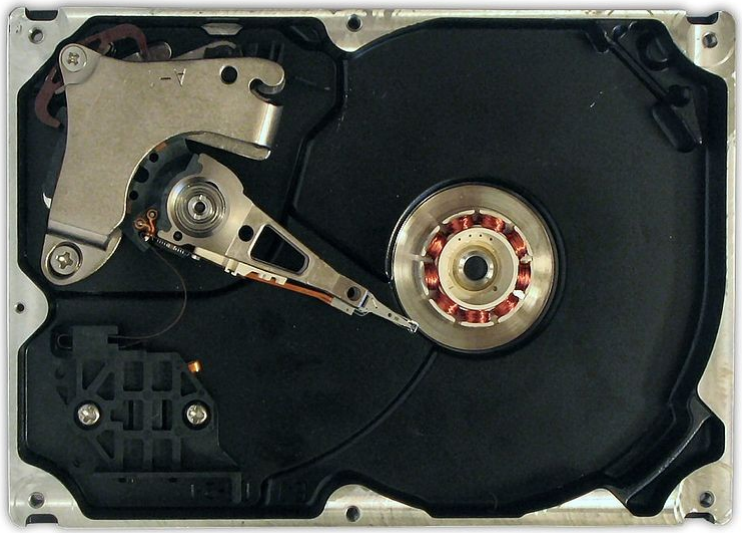
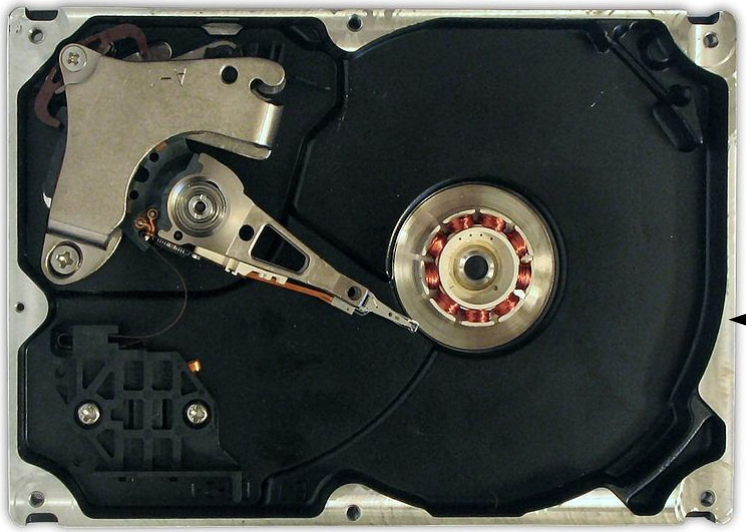
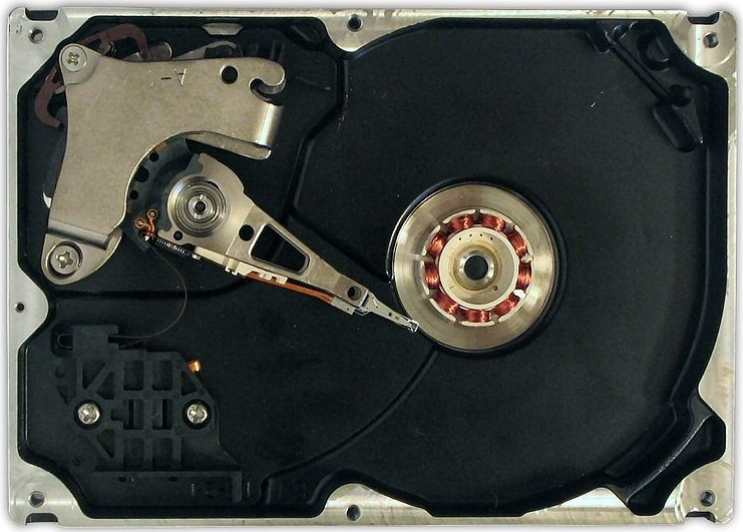


image sources
http://en.wikipedia.org/wiki/File:Hard_disk_dismantled.jpg
http://en.wikipedia.org/wiki/File:Memory_module_DDRAM_20-03-2006.jpg
http://en.wikipedia.org/wiki/File:Largetape.jpg




elixir
M2U51264DS8HC1G-5T
512MB DDR-400MHz/CL3
PC3200U-30331
Warranty Void If Removed
0542.MN05A110TA.5YM0410N08.XX.CN

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⏎755280276,889208095,590.32⏎934625720,459538801,44.66⏎852067113,660539228,1684.77⏎.....

Fixed-Length

65780793855956298 2002347
75528027688920809 5059032
93462572045953880 1004466
85206711366053922 8168477
+ another billion rows

start of Nth row = N * length of row

Fixed-Length

65780793855956298200234 7
75528027688920809505903 2
93462572045953880100446 6
85206711366053922816847 7
+ another billion rows

How do we **add** a row?
How do we **delete** a row?
How do we **find** a row?

Sort it?

75528027	00291787	99023934
75528027	68892080	95059032
75528027	98900892	34000020
75529012	39391917	21000129

+ 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))

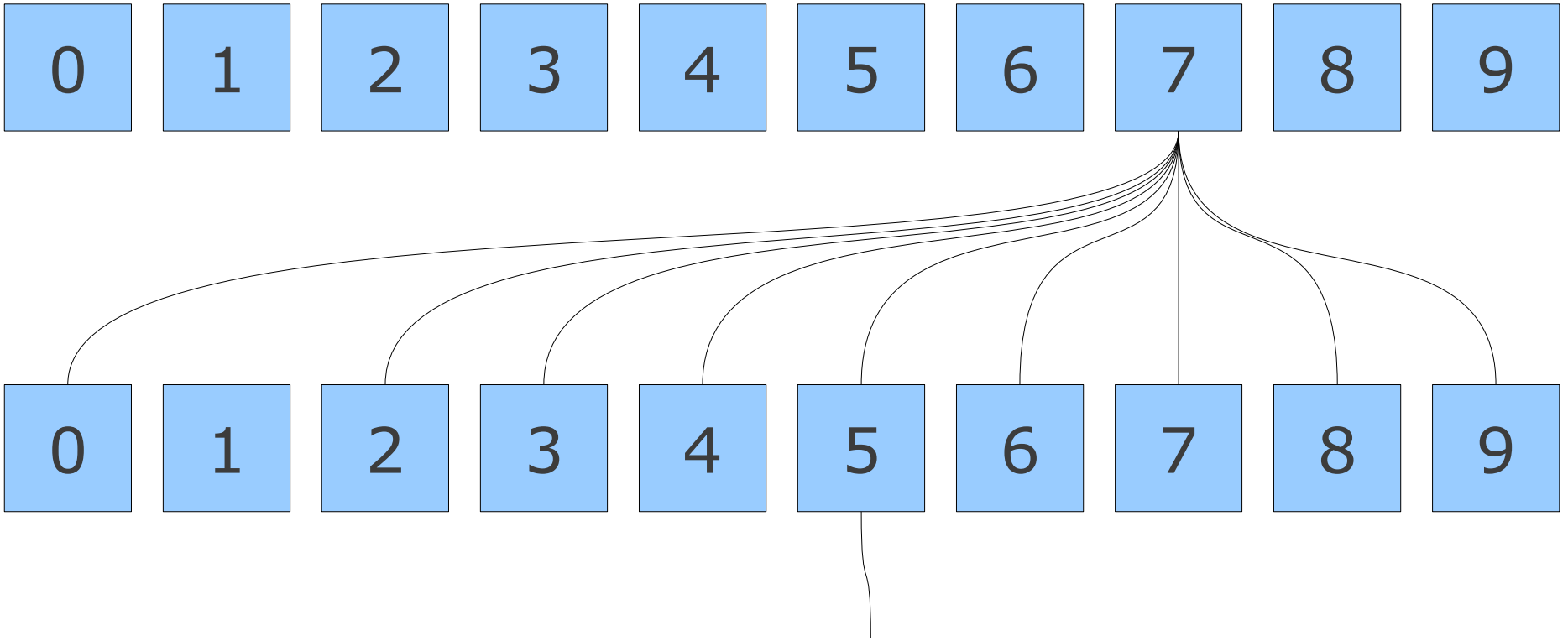
Inserting into a sorted list: O(n)

And the second field?

75528027002917879902393 4
75528027688920809505903 2
75528027900892340000020
75529012393919172100012 9
+ 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:

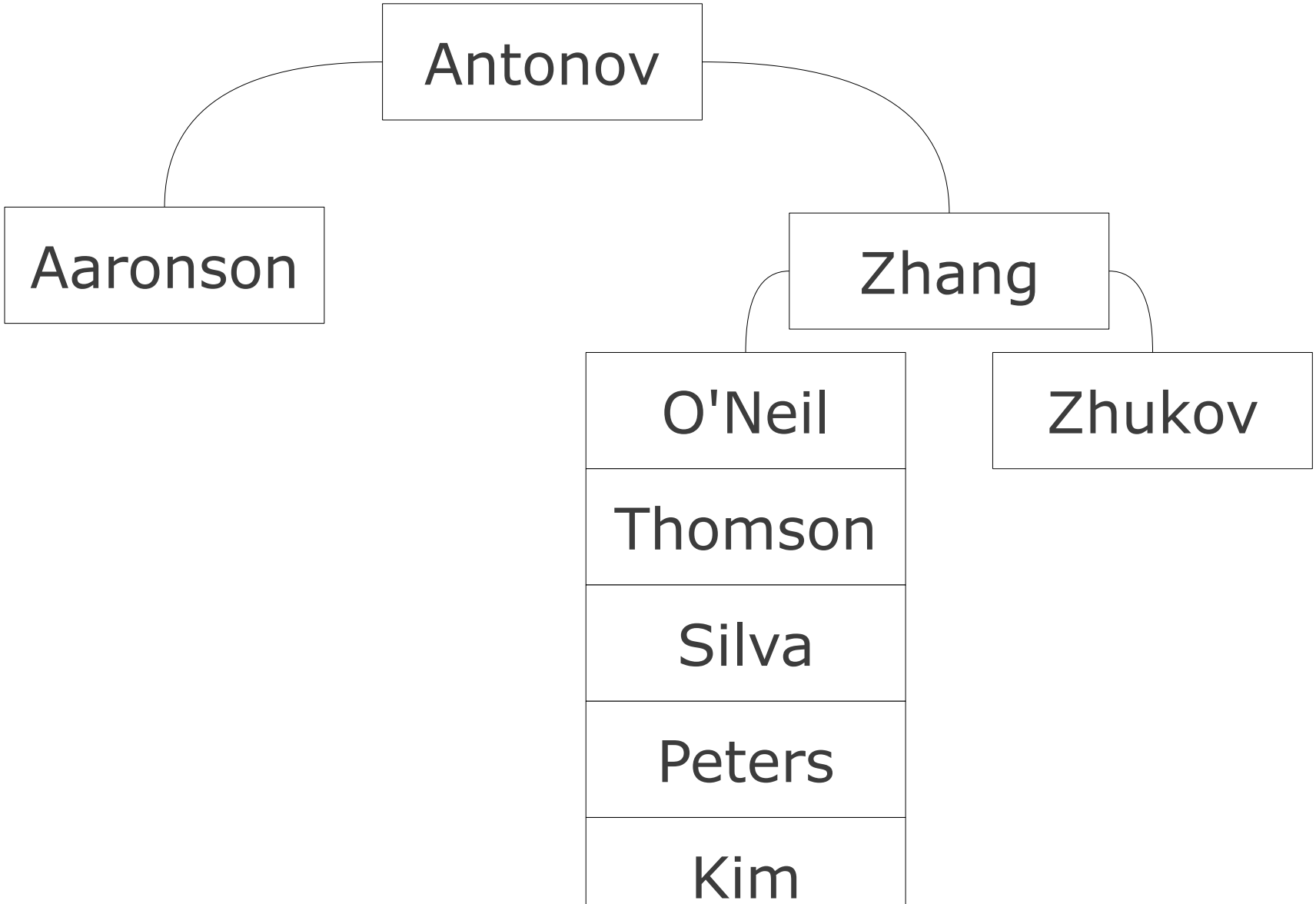
E.g., ISAM

Balanced:

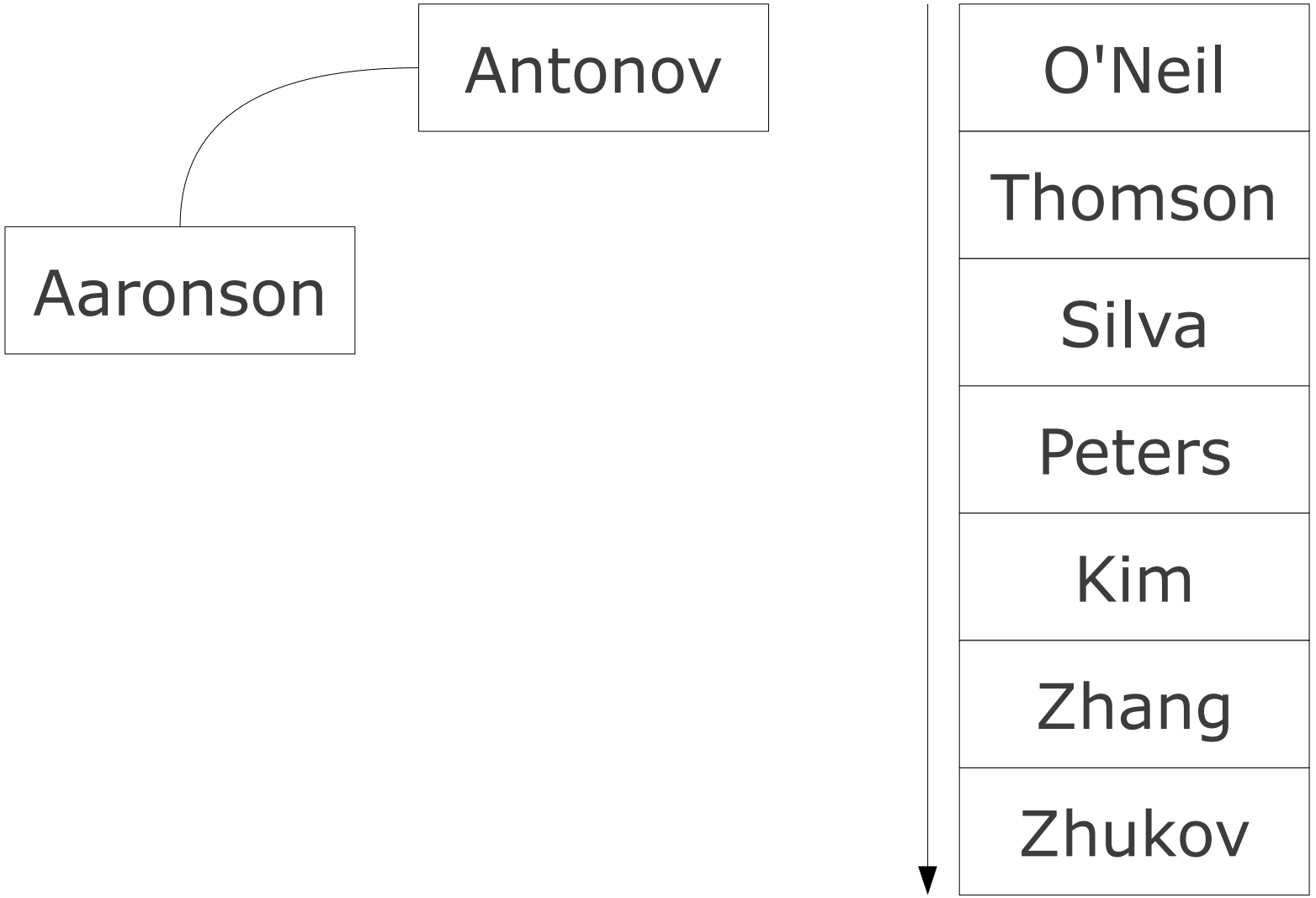
E.g., B+ Trees

A: 4%
M: 9.5%
Q: 0

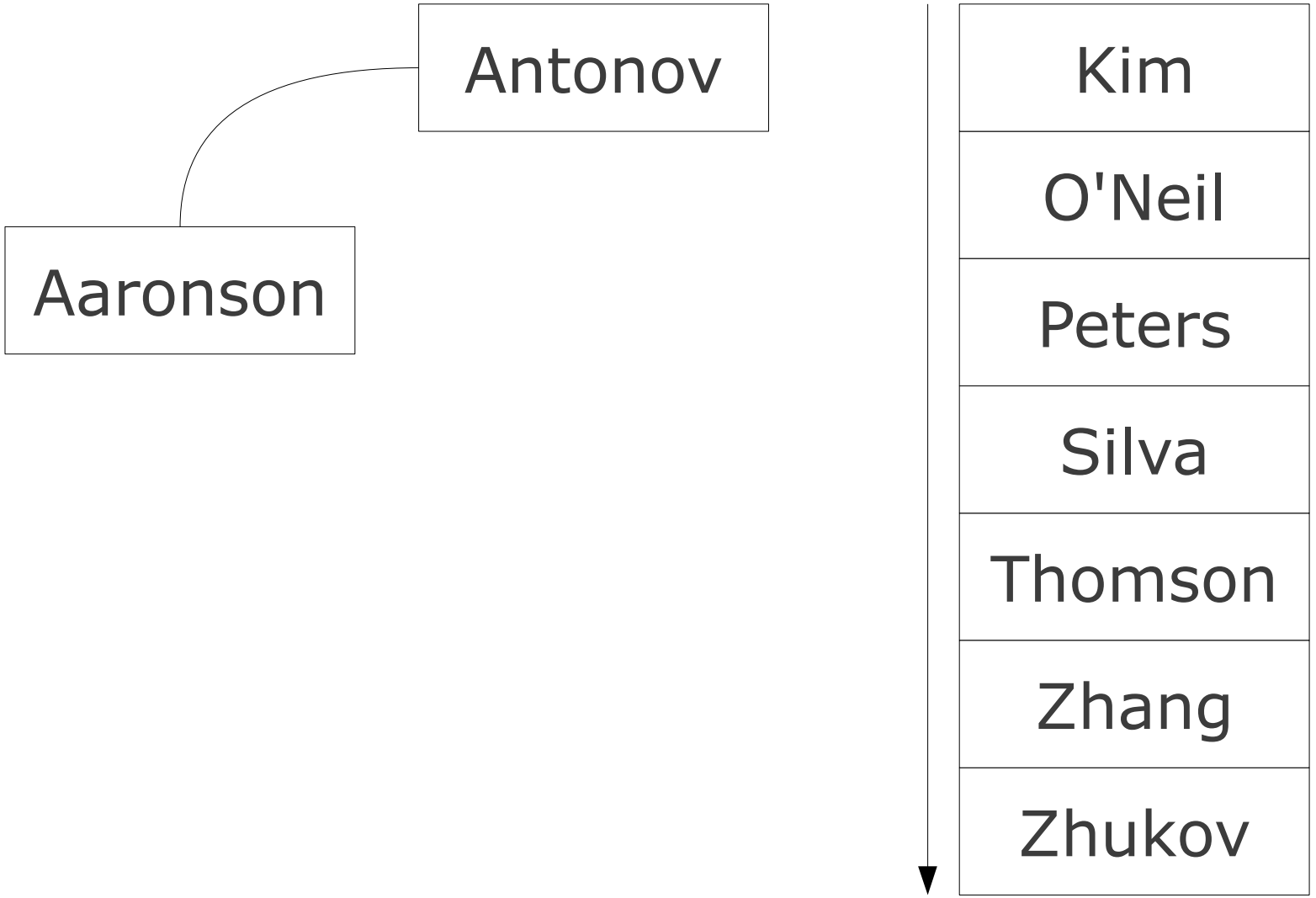
Balanced Tree



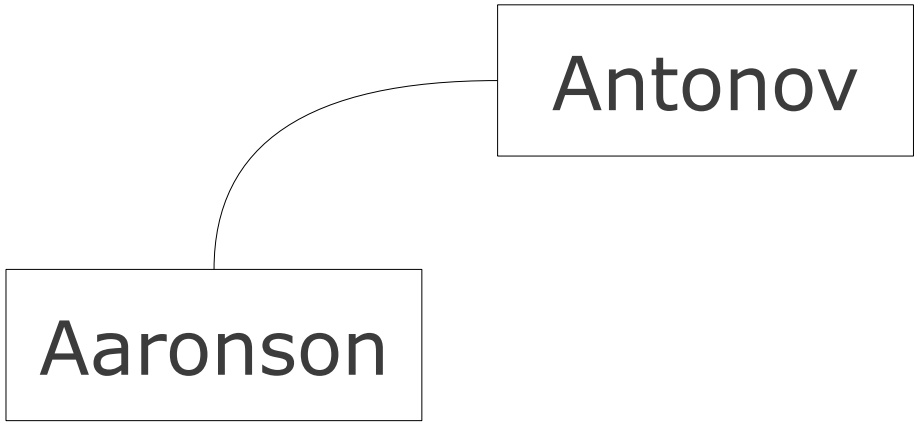
Balanced Tree



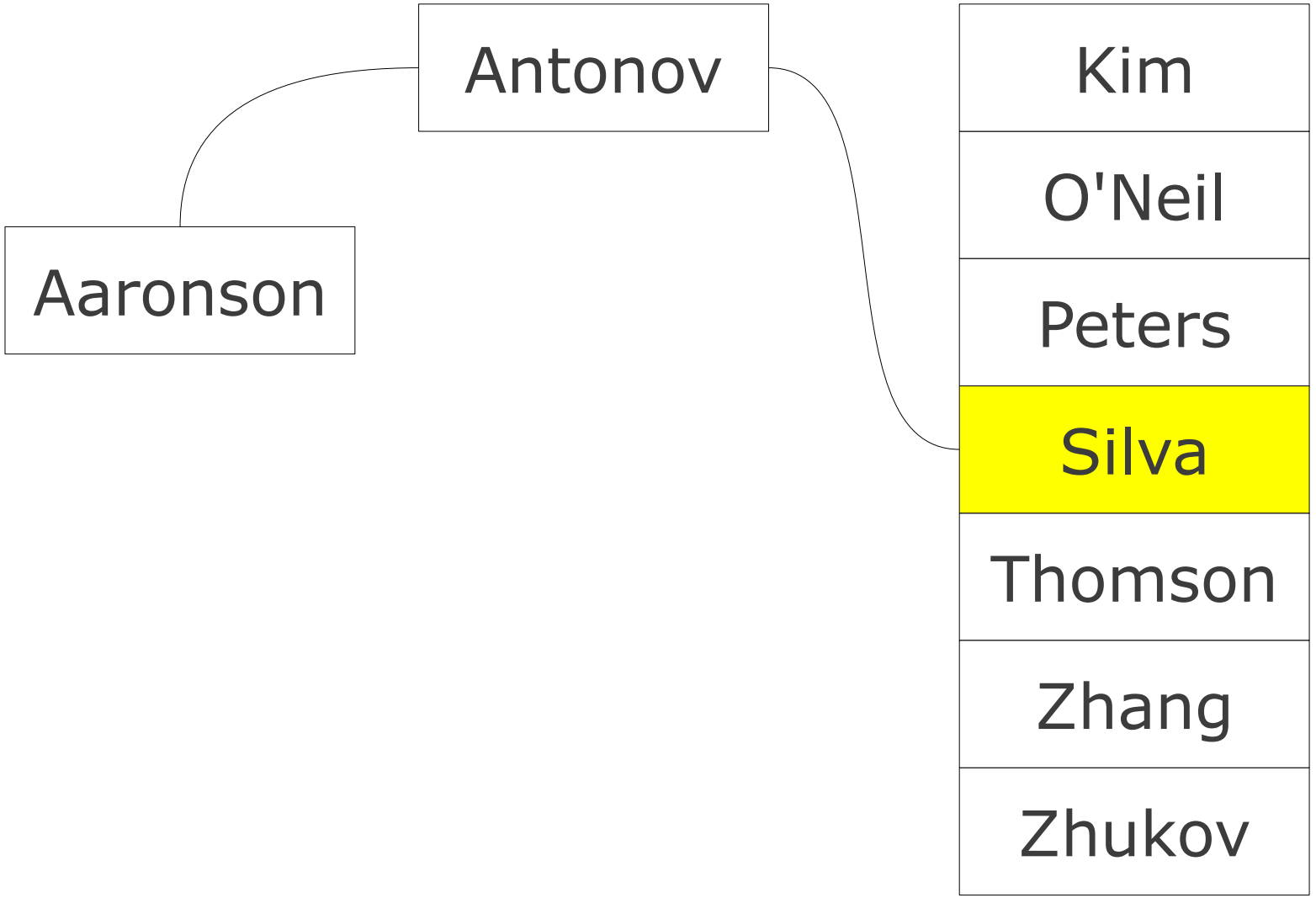
Balanced Tree



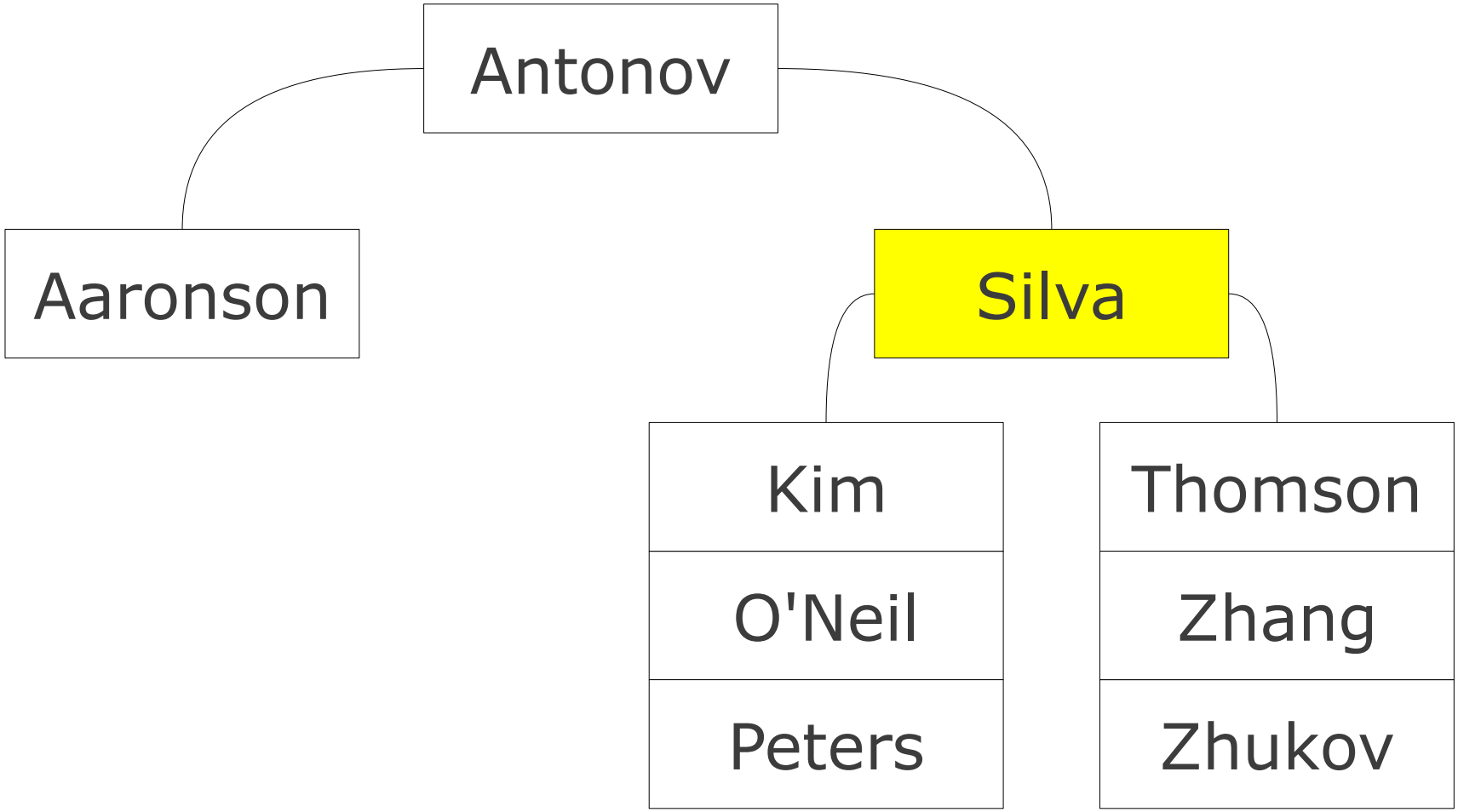
Balanced Tree



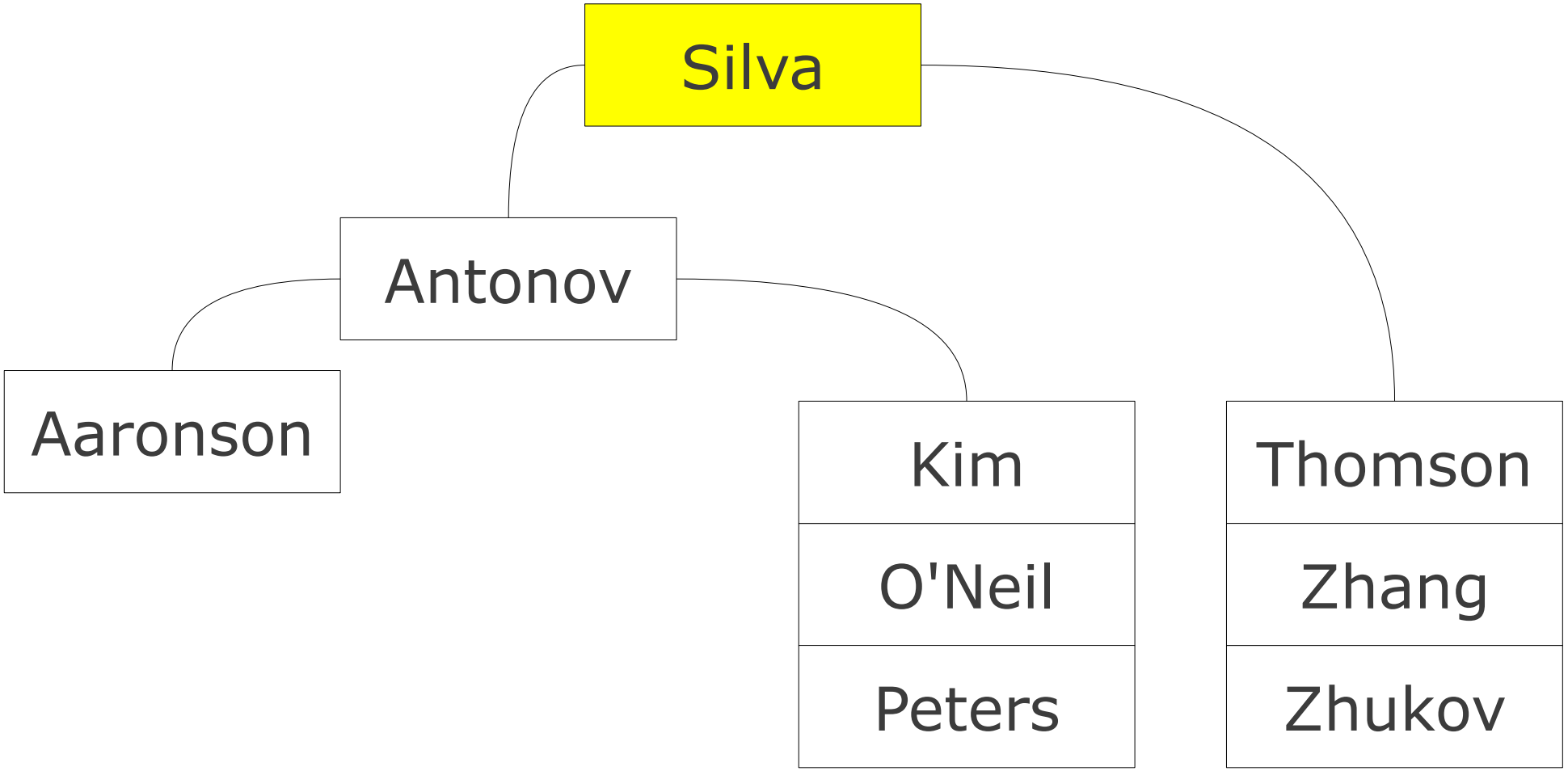
Balanced Tree



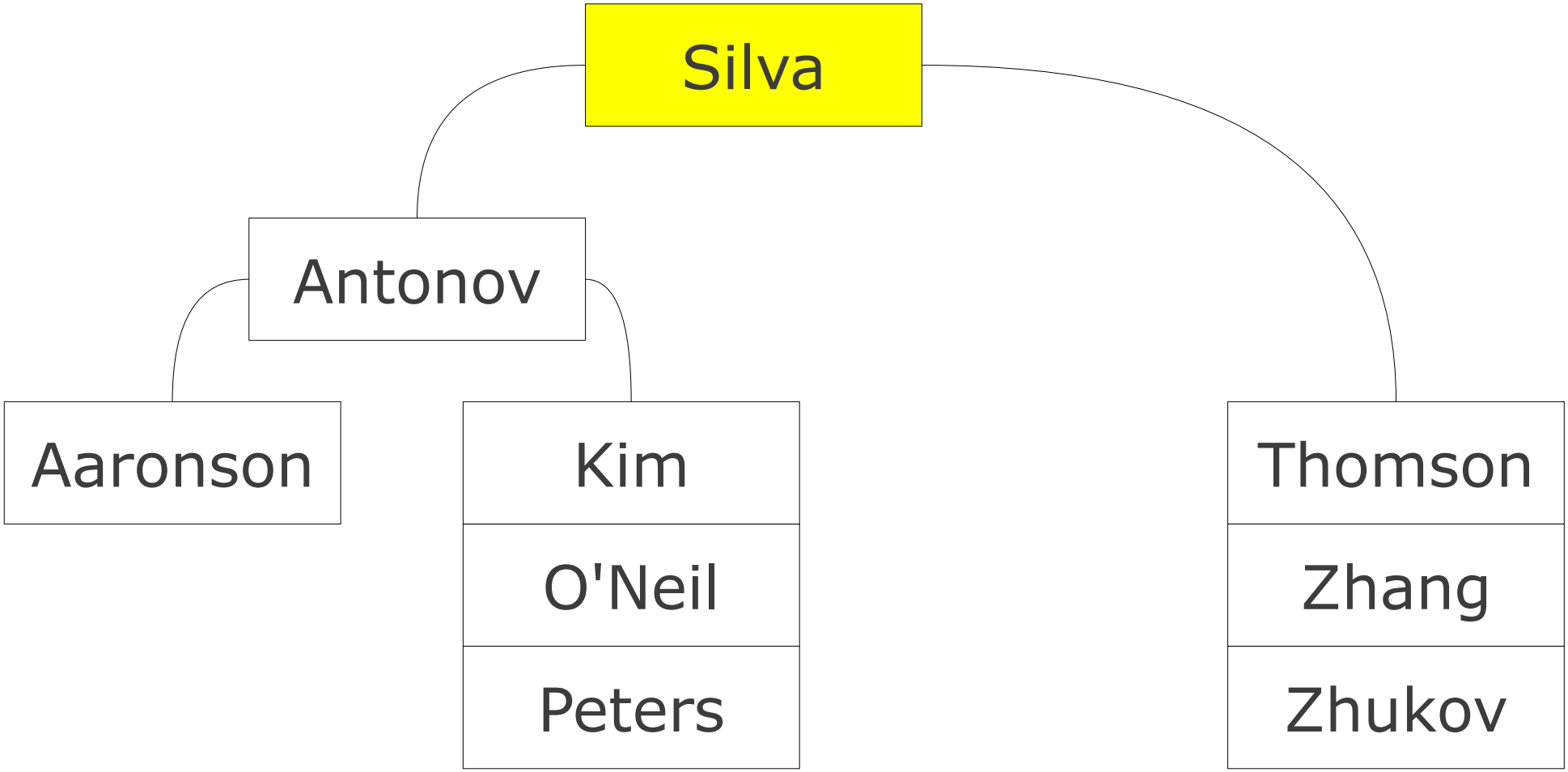
Balanced Tree



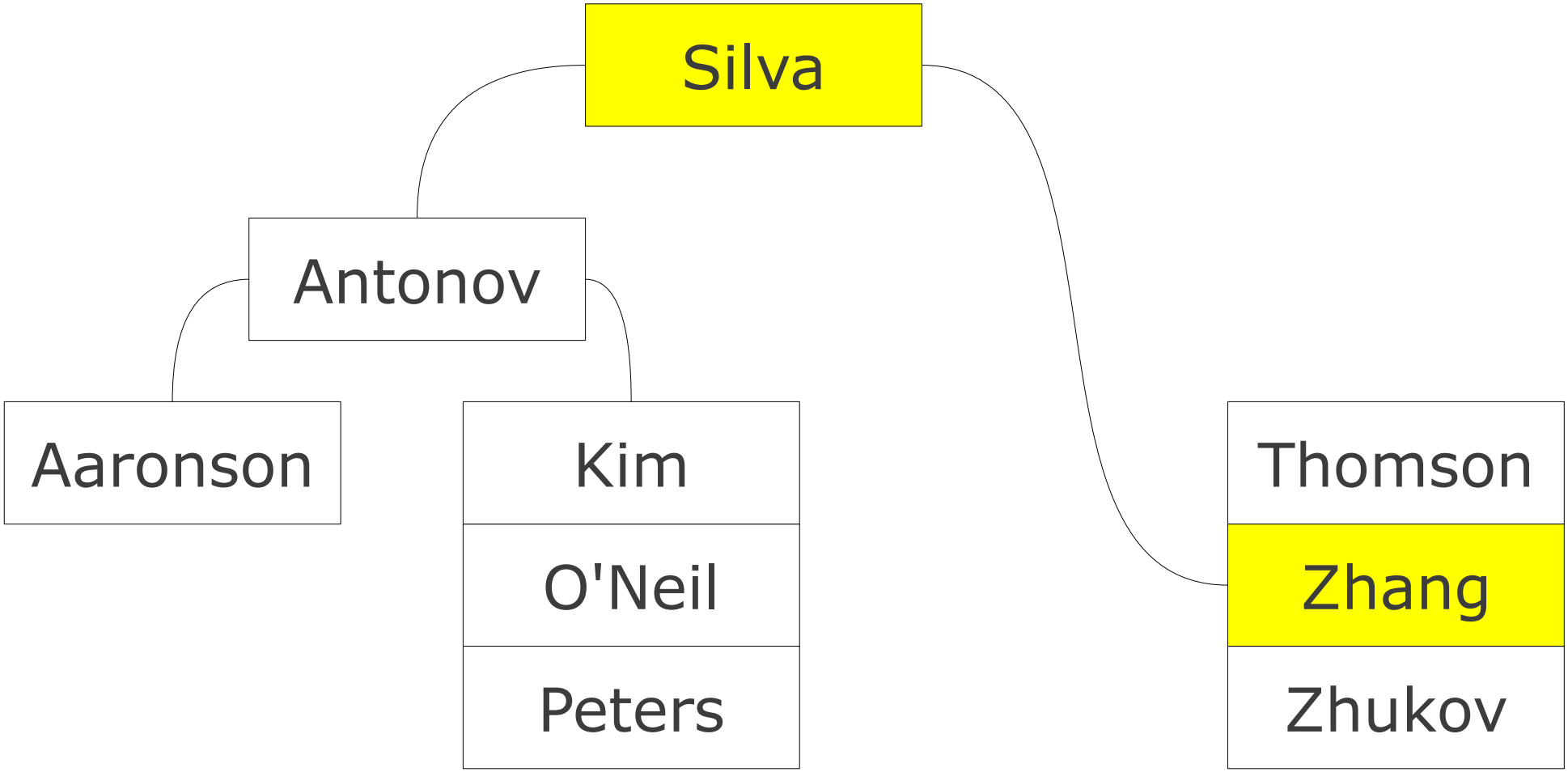
Balanced Tree



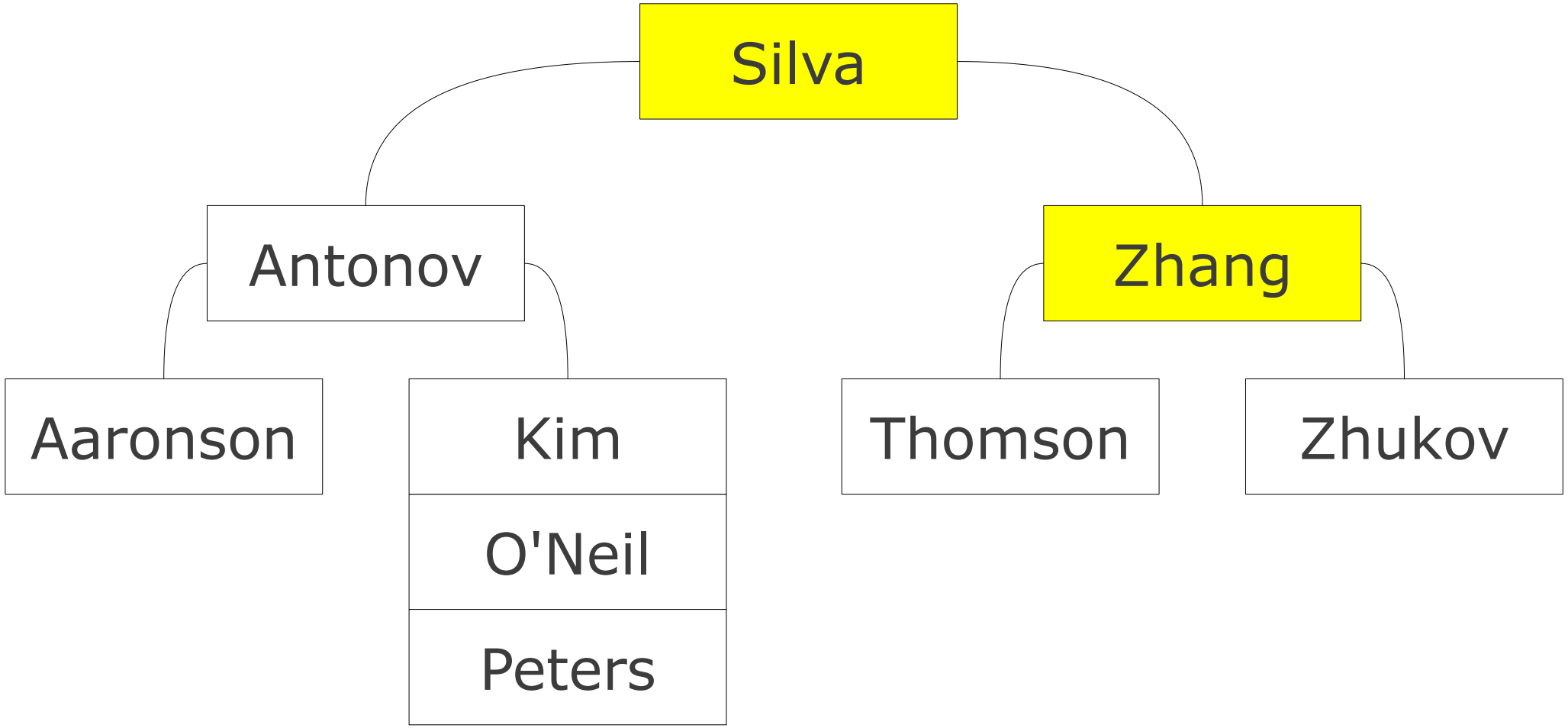
Balanced Tree



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.)