

# CCT395, Week 11

## Review, Permissions Physical Storage and Performance

Yuri Takhteyev  
University of Toronto  
November 17, 2010



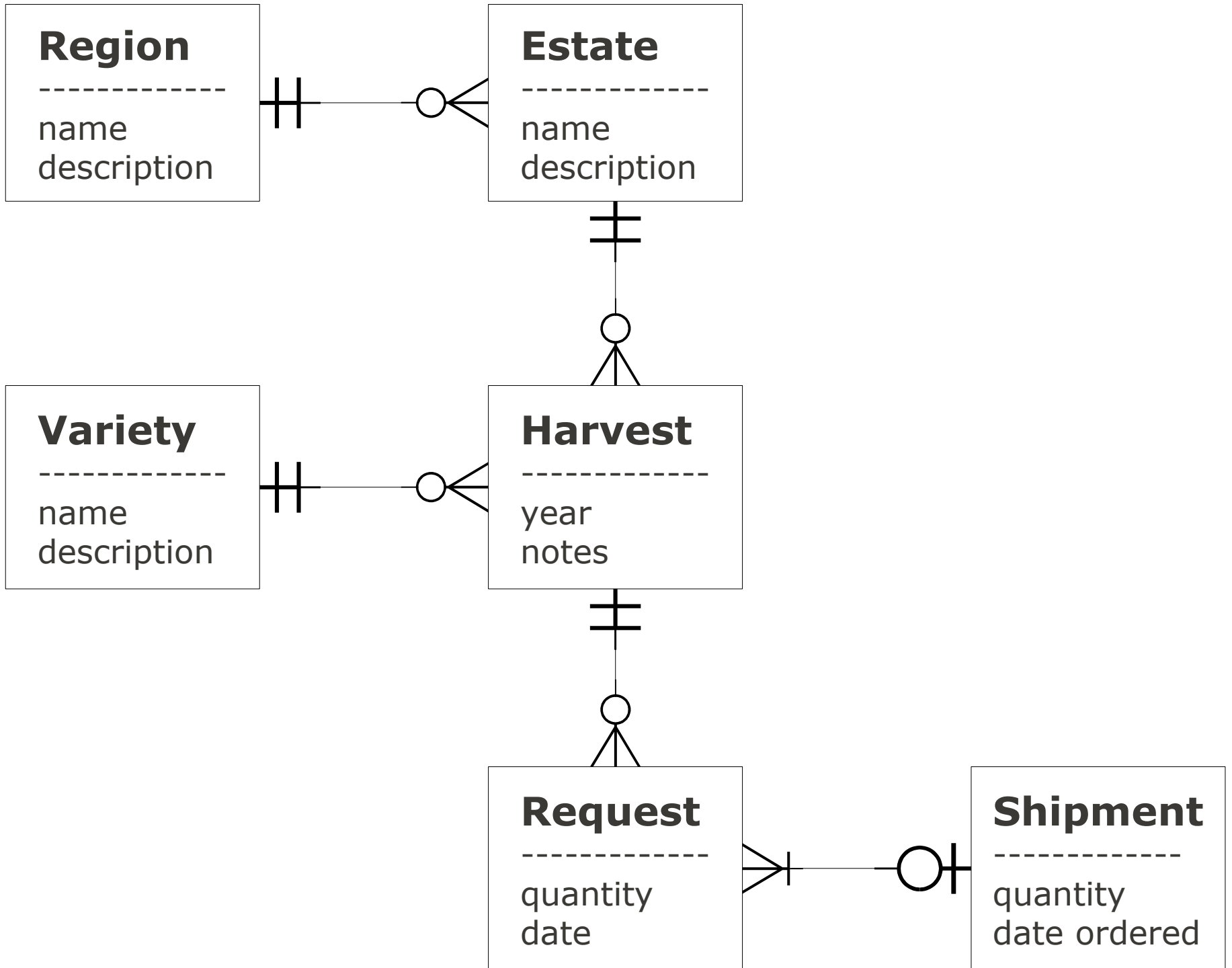
This presentation is licensed under Creative Commons Attribution License, v. 3.0. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/>. This presentation incorporates images from the Crystal Clear icon collection by Everaldo Coelho, available under LGPL from <http://everaldo.com/crystal/>.

# The Exam

25%: ER + Normalization

50%: Fill-in-the-blanks SQL

25%: Conceptual questions



```
select superhero.name from superhero
```

```
join _____
```

```
on _____
```

```
where _____ != "Lex Luthor"
```

```
_____ by _____
```

# Conceptual questions

- Main ideas
- Things from the last 5 weeks  
(check the updated outline)

Questions?

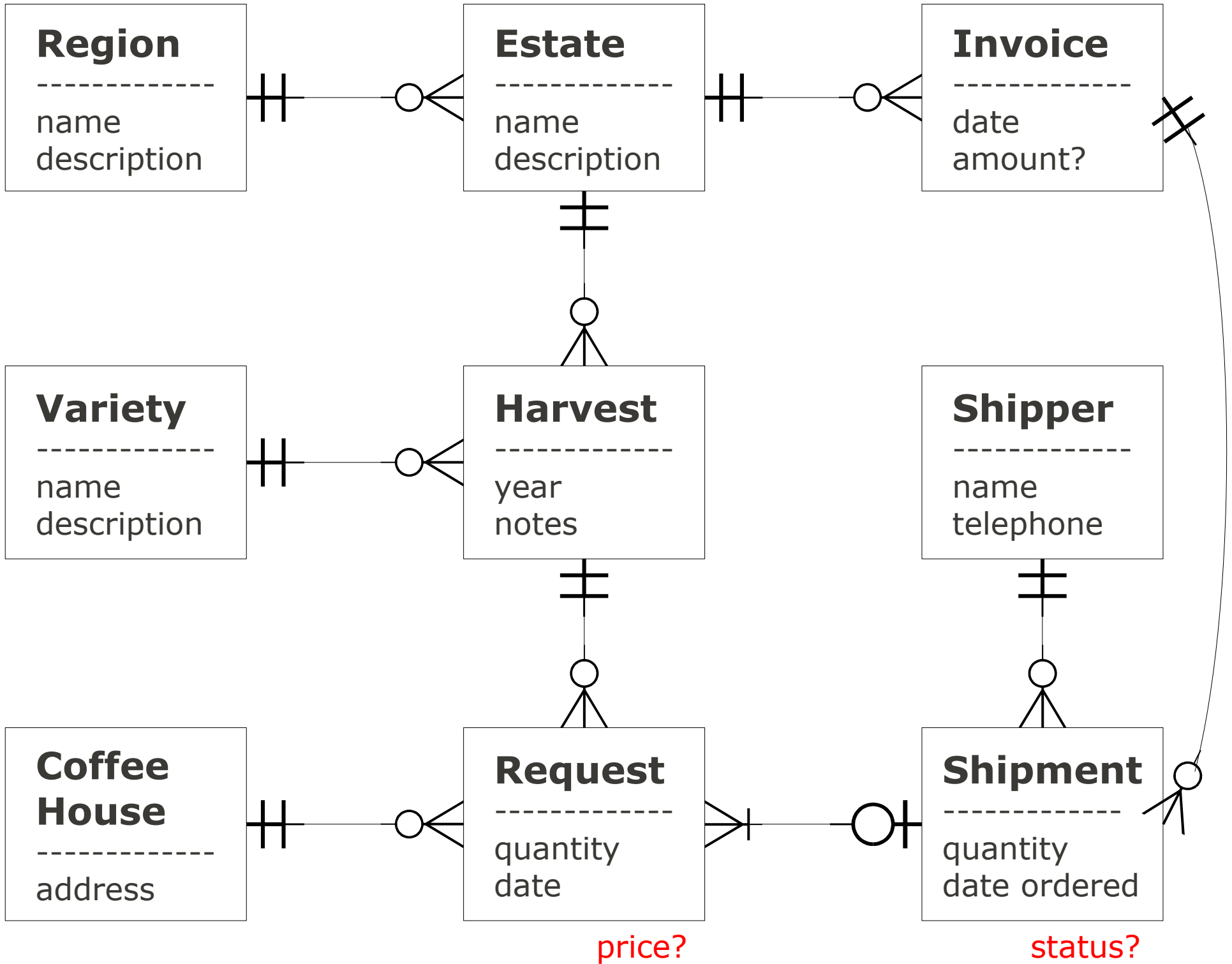
João's Coffee

# Divide and Conquer

Getting the customers the right coffee

- Ordering the right coffee
- Delivering the coffee to the right customers





# SQL

```
create table harvest (  
    harvest_id int,  
    variety_id int,  
    foreign key (variety_id)  
        references variety(variety_id) ,  
    estate_id int,  
    foreign key (estate_id)  
        references estate(estate_id) ,  
    year year,  
    notes varchar(200) ,  
    primary key (harvest_id) ,  
);
```

# Storage

## **Persistent storage**

usually a harddrive(s)

## **Challenges**

finding a representation

not losing information

storage space

performance

# CSV

simple!

relatively small

but what to do with NULLs?

and how would it perform?

# CSV

657807938,559562982,223.47

755280276,889208095,590.32

934625720,459538801,344.66

852067113,660539228,684.77

+ another billion rows

How do we **add** a row?

How do we **delete** a row?

How do we **find** a row?

# What if we sort it?

755280270,989809237,910.21

755280276,889208095,590.32

755280276,889909232,293.12

755280281,198234234,120.56

755280291,782737671,123.23

Finding is **easier!**

But adding is **harder...**

# And the second field?

479837218,889208091,345.29

755280276,889208095,590.32

879873454,889208098,938.29

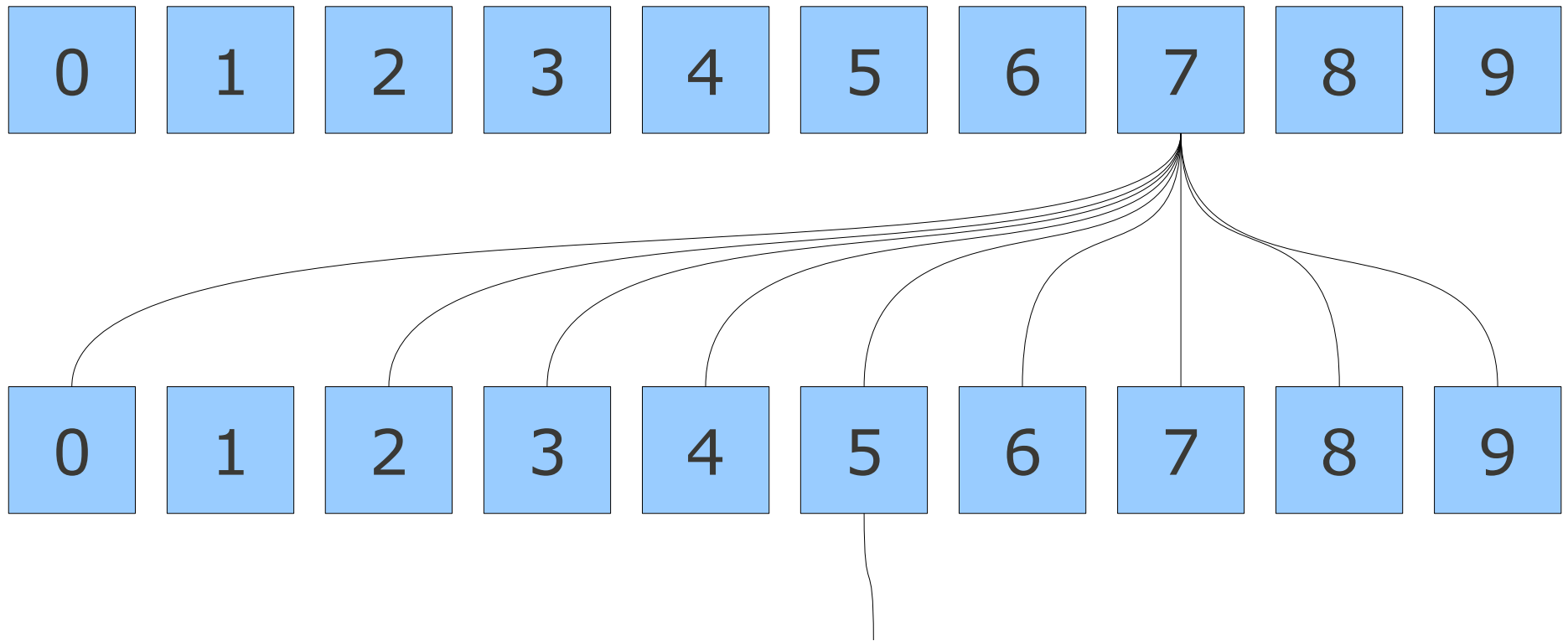
767983465,889208101,389.02

574983342,889208121,291.73

Essentially an **index**.

Easier to find, more stuff to store, more work to insert.

# Trees



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



# Fixed vs Dynamic

## **ISAM:**

The structure is fixed

## **B+ trees:**

The structure changes

**A:** 4%

**O:** <0.5%

**M:** 9.5%

# CCT395, Week 9

## Review, Permissions Physical Storage and Performance

Yuri Takhteyev  
University of Toronto  
November 17, 2010



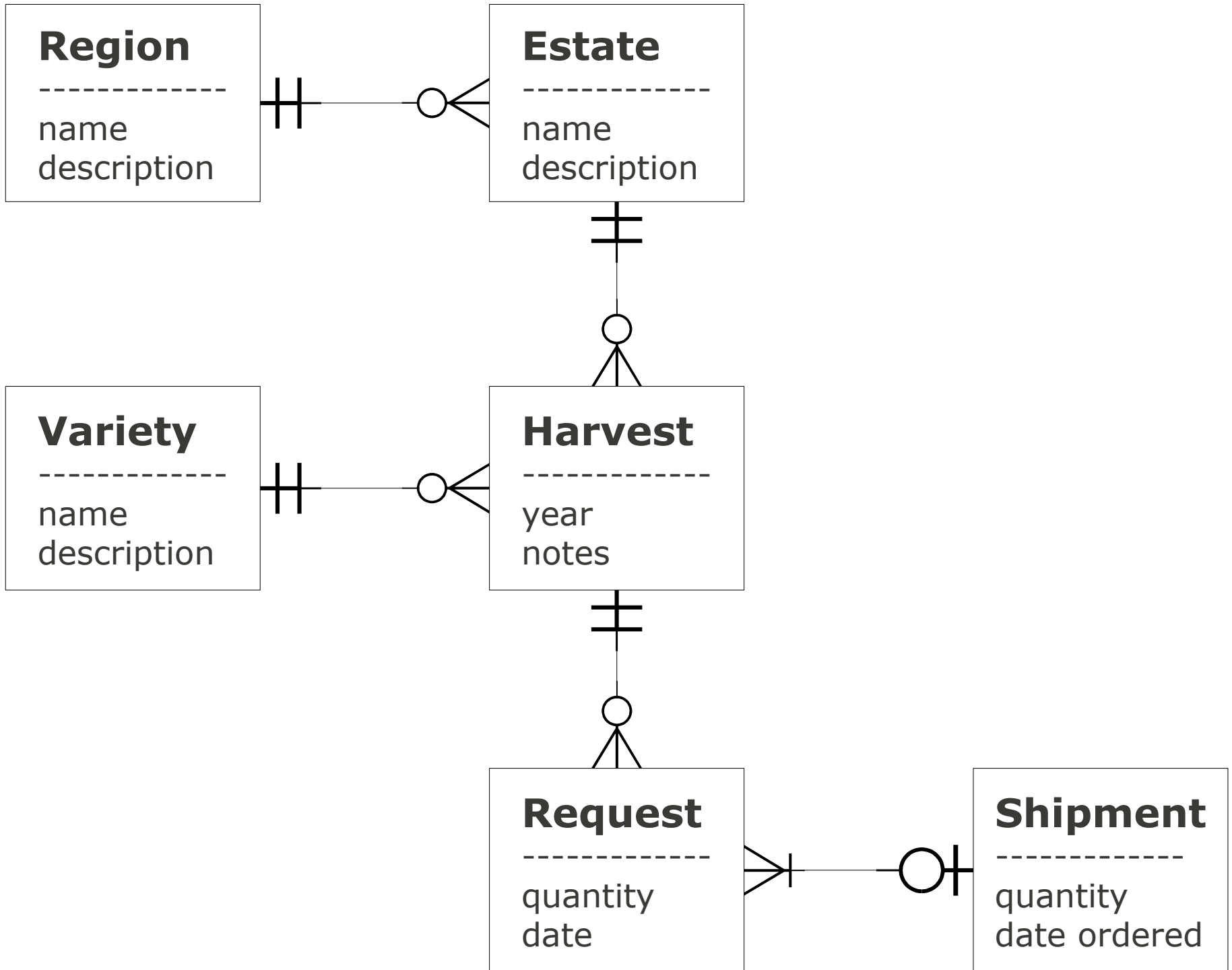
This presentation is licensed under Creative Commons Attribution License, v. 3.0. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/>. This presentation incorporates images from the Crystal Clear icon collection by Everaldo Coelho, available under LGPL from <http://everaldo.com/crystal/>.

# The Exam

25%: ER + Normalization

50%: Fill-in-the-blanks SQL

25%: Conceptual questions



```
select superhero.name from superhero
```

```
join _____
```

```
on _____
```

```
where _____ != "Lex Luthor"
```

```
_____ by _____
```

# Conceptual questions

- Main ideas
- Things from the last 5 weeks  
(check the updated outline)

Questions?

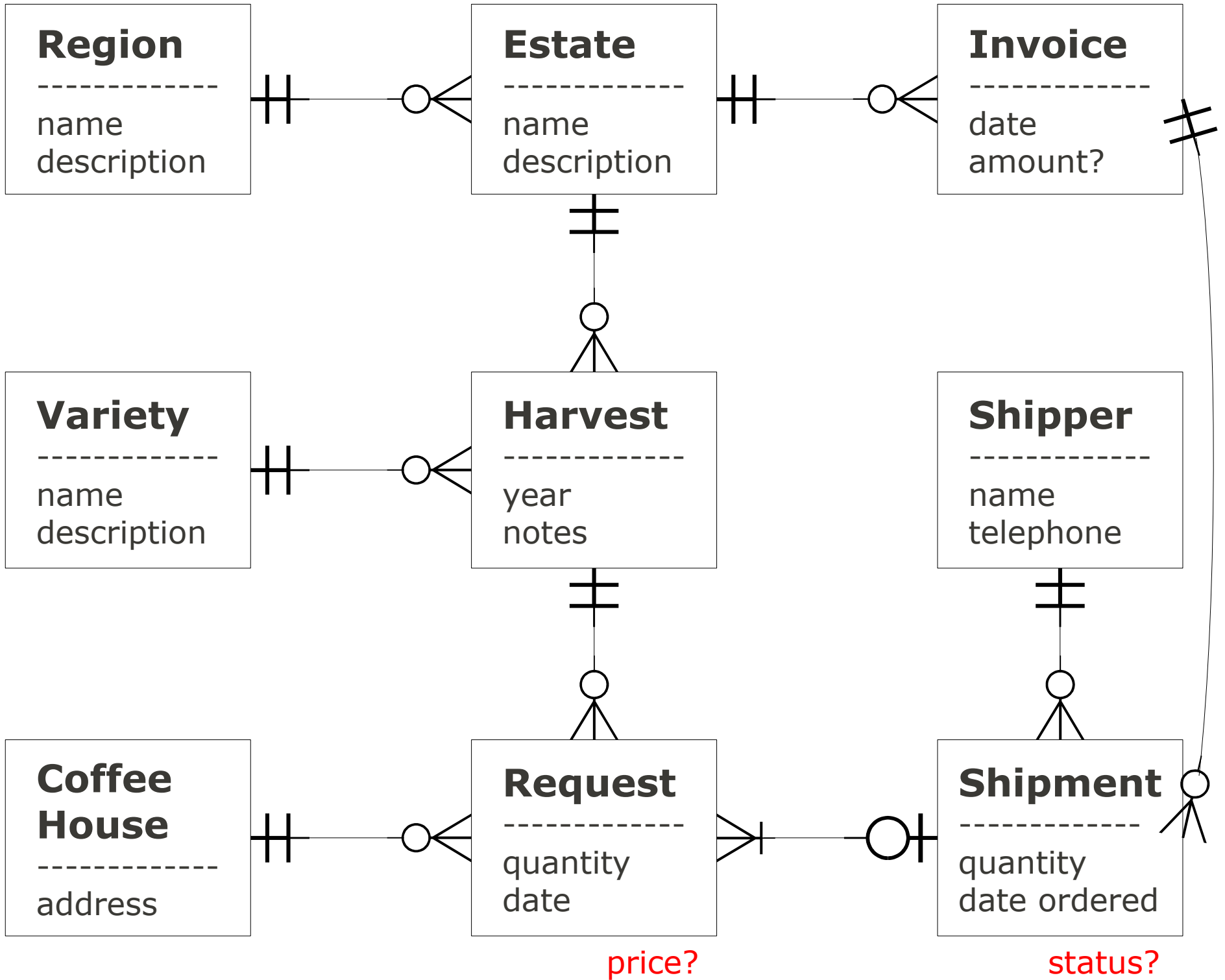
João's Coffee



# Divide and Conquer

Getting the customers the right coffee

- Ordering the right coffee
- Delivering the coffee to the right customers



# SQL

```
create table harvest (  
    harvest_id int,  
    variety_id int,  
    foreign key (variety_id)  
        references variety(variety_id) ,  
    estate_id int,  
    foreign key (estate_id)  
        references estate(estate_id) ,  
    year year,  
    notes varchar(200) ,  
    primary key (harvest_id) ,  
);
```

# Storage

## **Persistent storage**

usually a harddrive(s)

## **Challenges**

finding a representation

not losing information

storage space

performance

# 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 =  $N * \text{length of row}$

# CSV

657807938,559562982,23.47

755280276,889208095,590.32

934625720,459538801,44.66

852067113,660539228,1684.77

+ another billion rows

How do we **add** a row?

How do we **delete** a row?

How do we **find** a row?

# What if we sort it?

755280270,989809237,10.21

755280276,889208095,590.32

755280276,889909232,293.12

755280281,198234234,3120.56

755280291,782737671,13.23

Finding is **easier!**

But adding is **harder...**

# And the second field?

479837218,889208091,12345.29

755280276,889208095,590.32

879873454,889208098,38.29

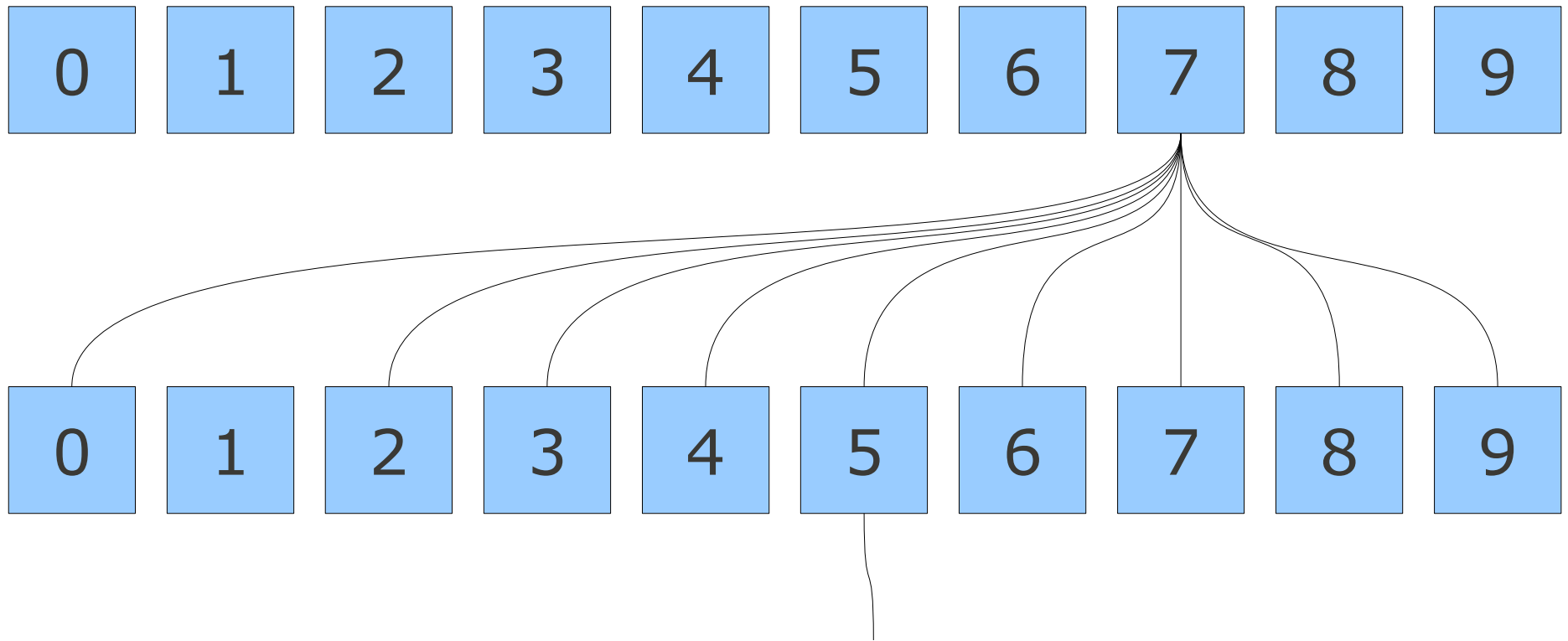
767983465,889208101,9.02

574983342,889208121,1.73

Essentially an **index**.

Easier to find, more stuff to store, more work to insert.

# 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:**

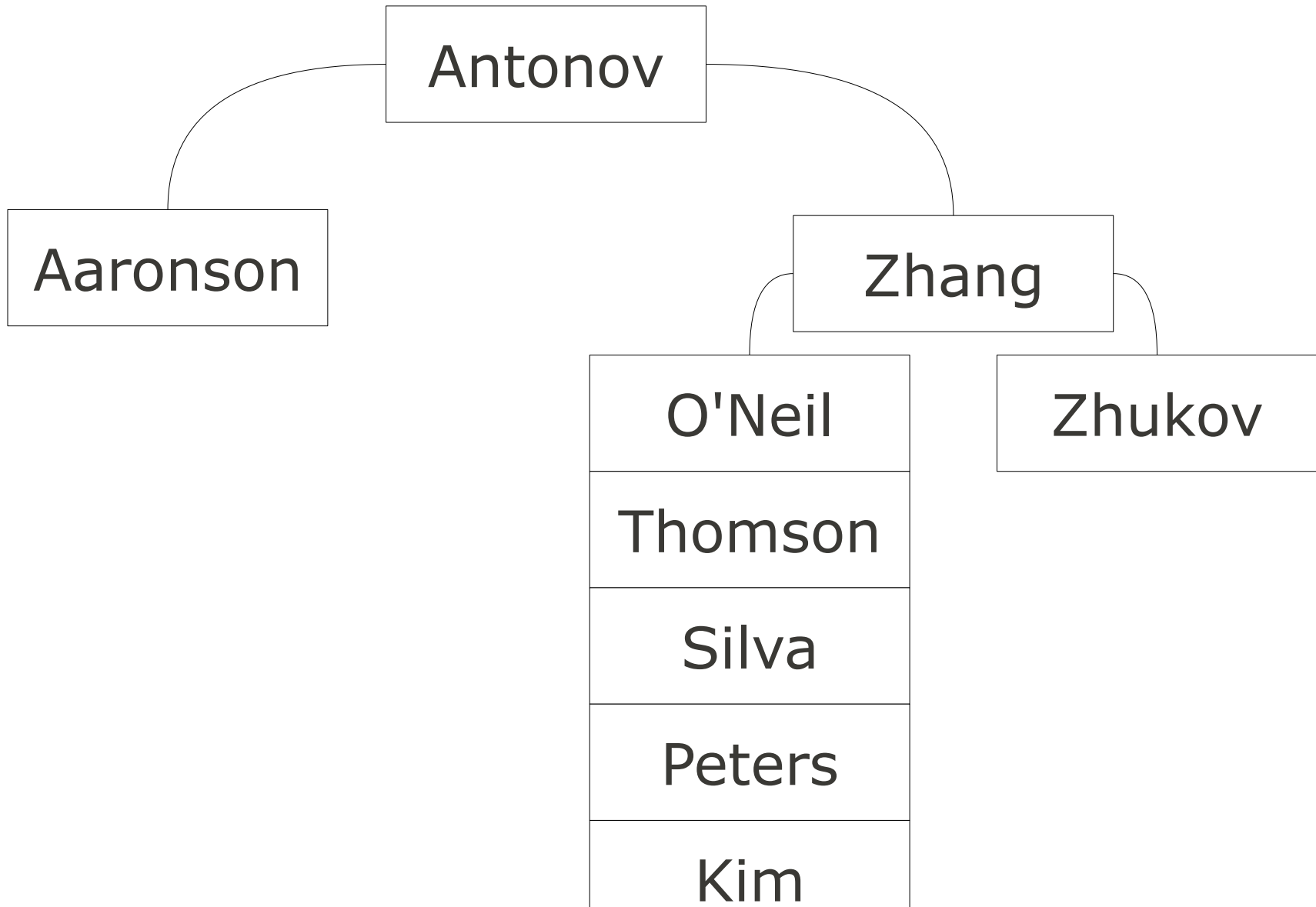
E.g., B+ Trees

**A:** 4%

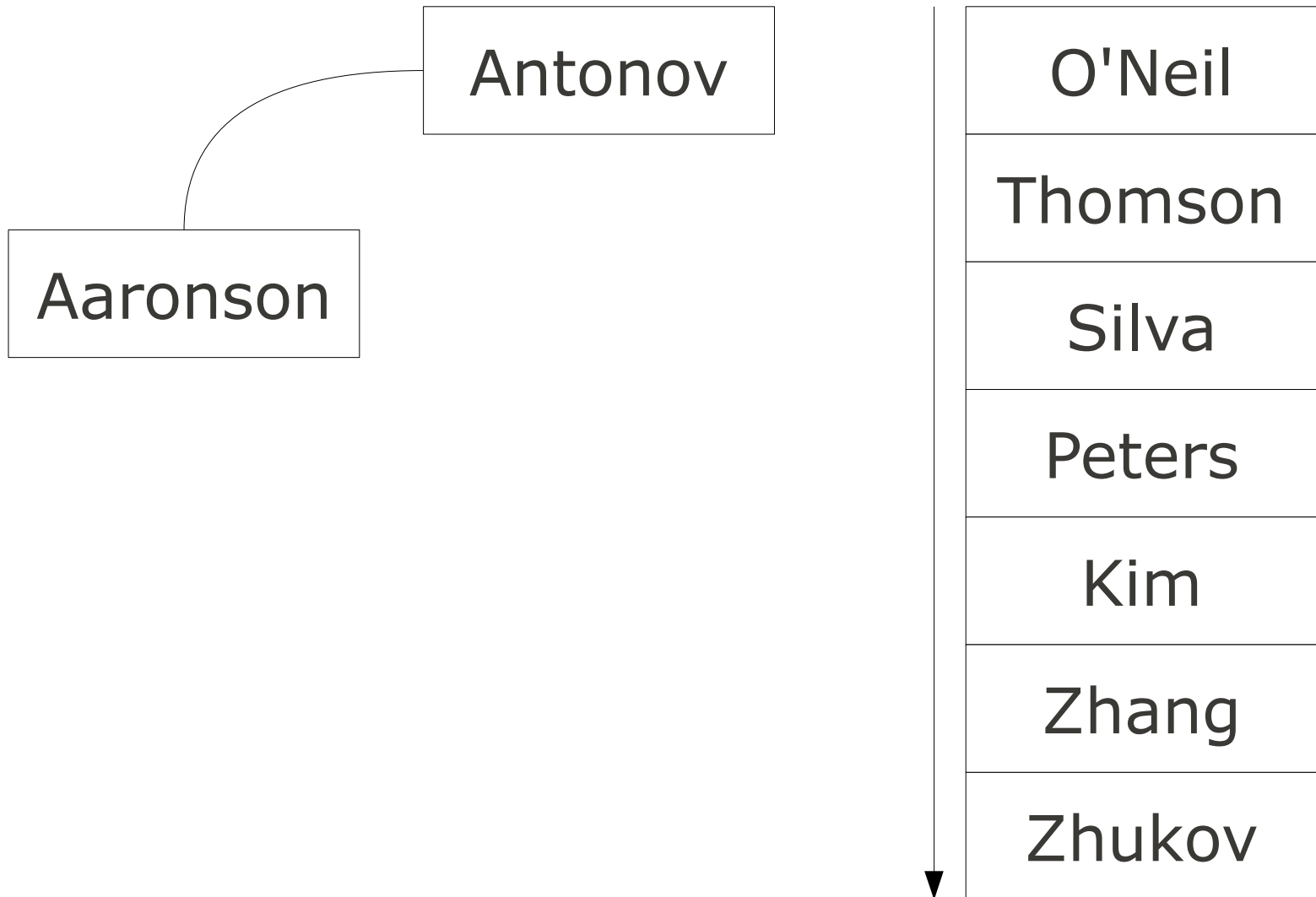
**M:** 9.5%

**Q:** 0

# Balanced Tree

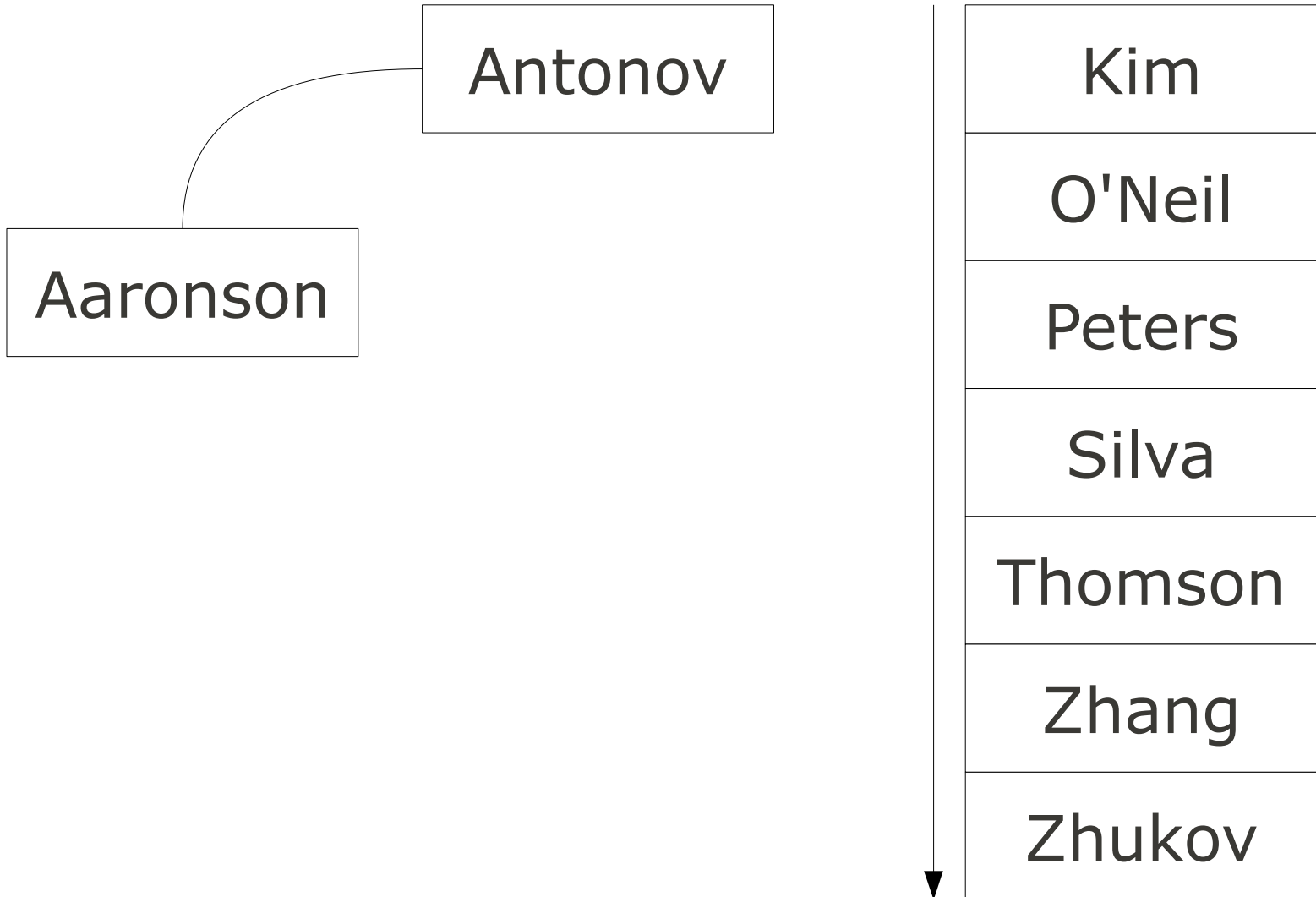


# Balanced Tree

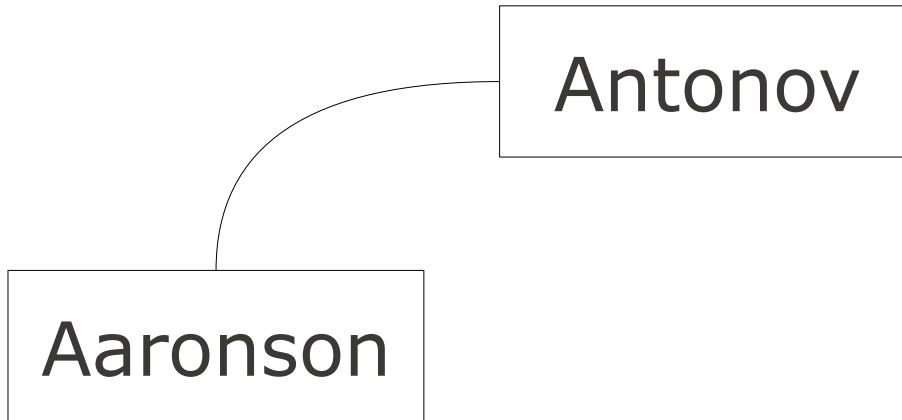




# Balanced Tree

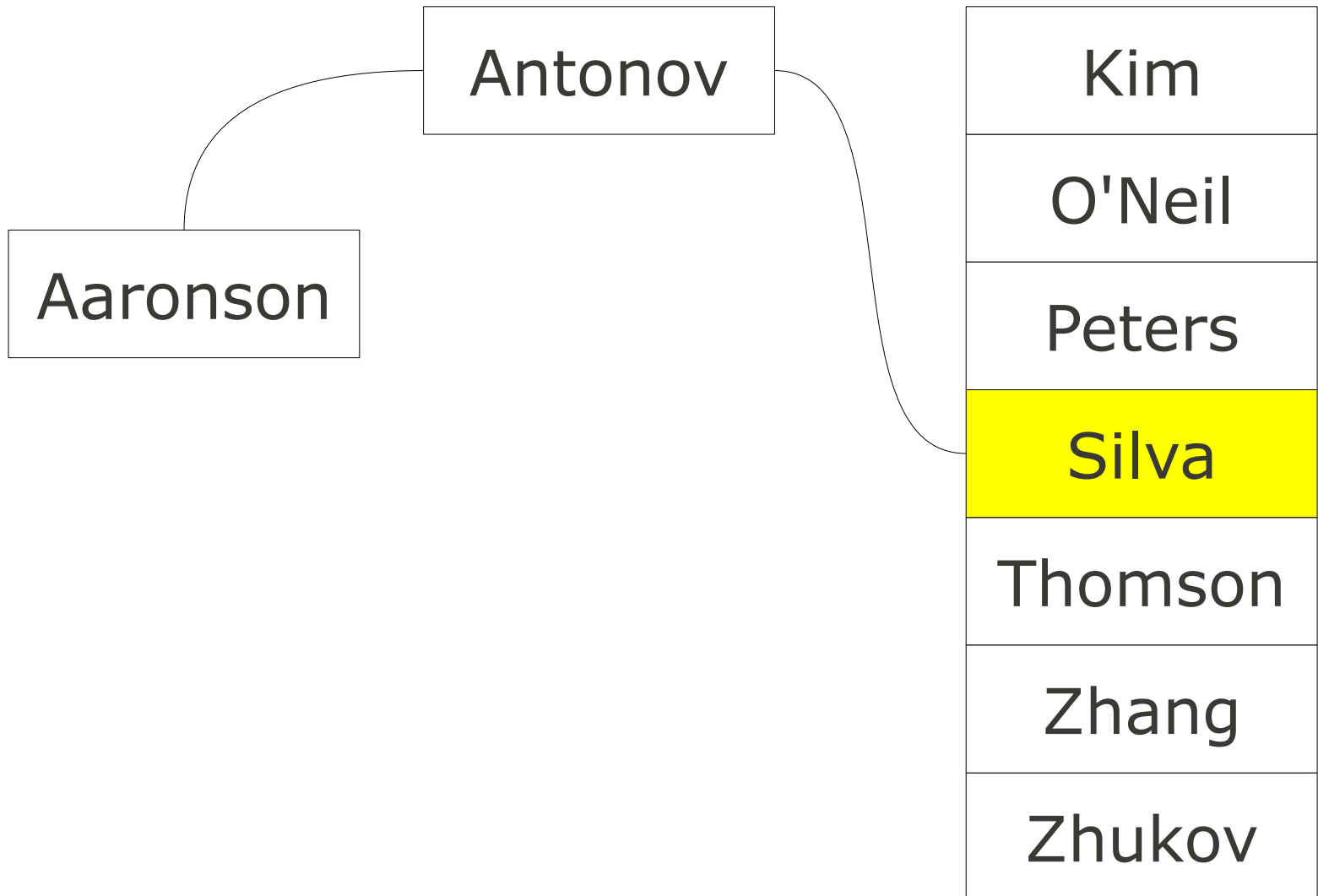


# Balanced Tree

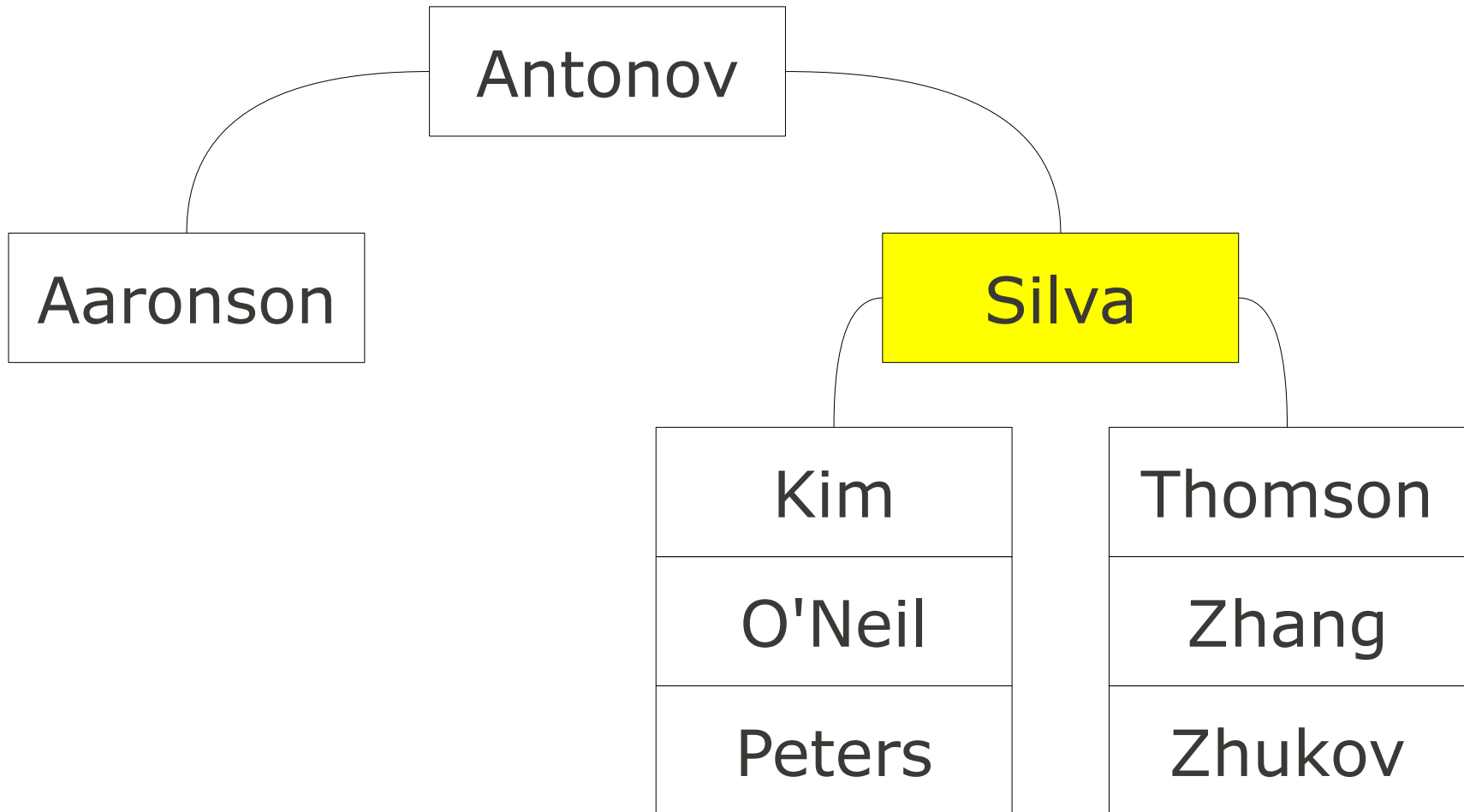


Kim
O'Neil
Peters
Silva
Thomson
Zhang
Zhukov

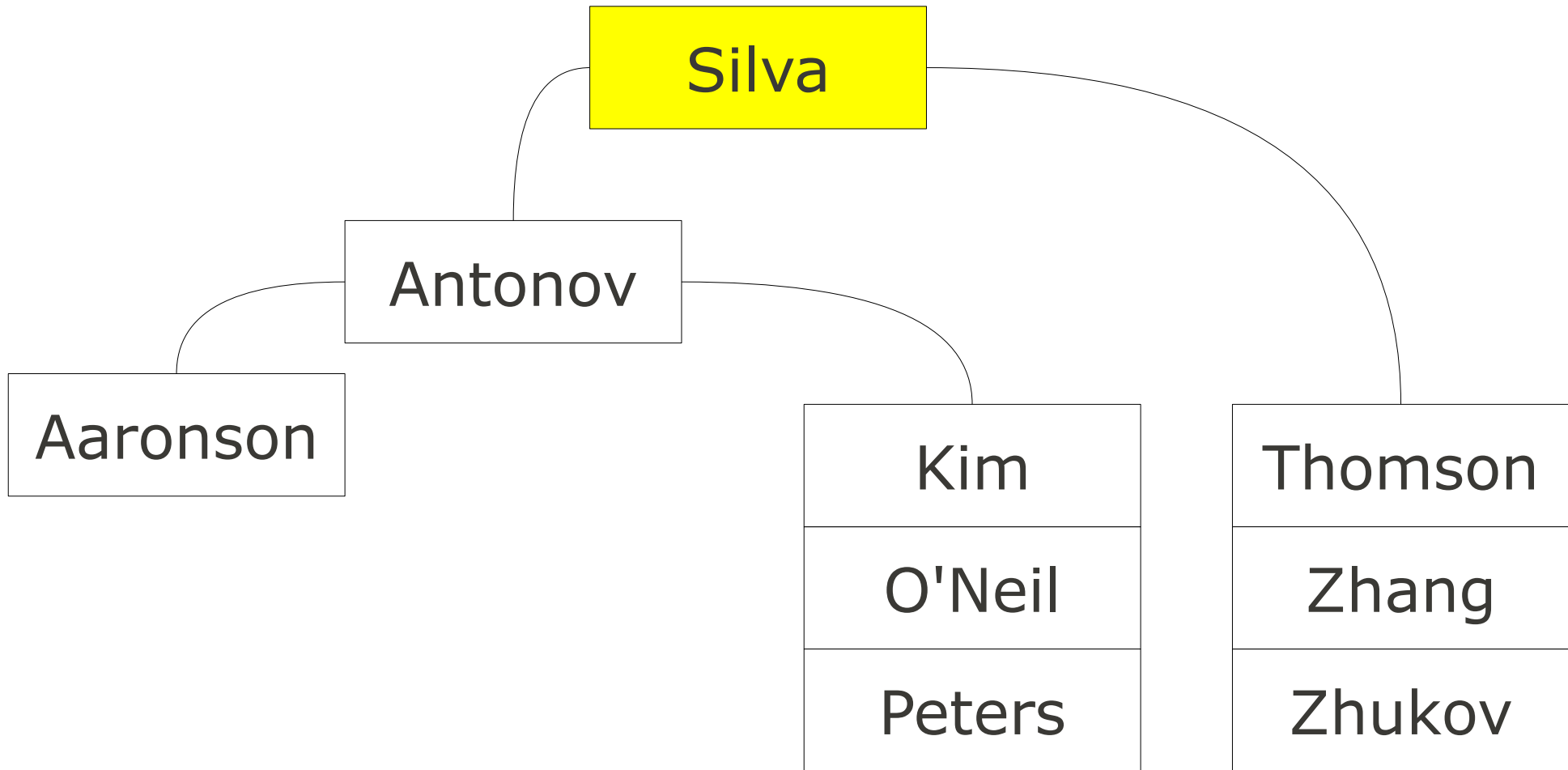
# Balanced Tree



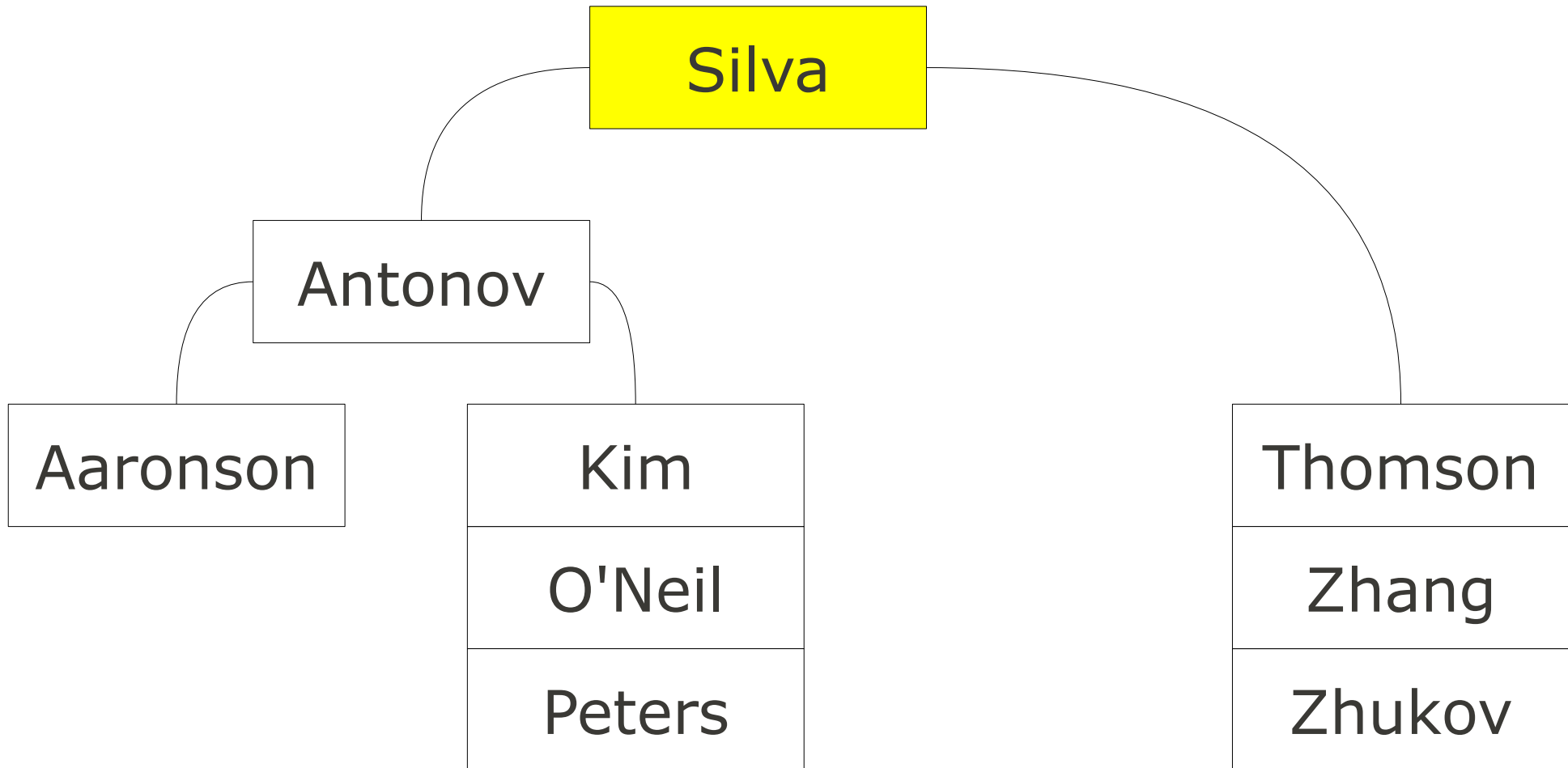
# Balanced Tree



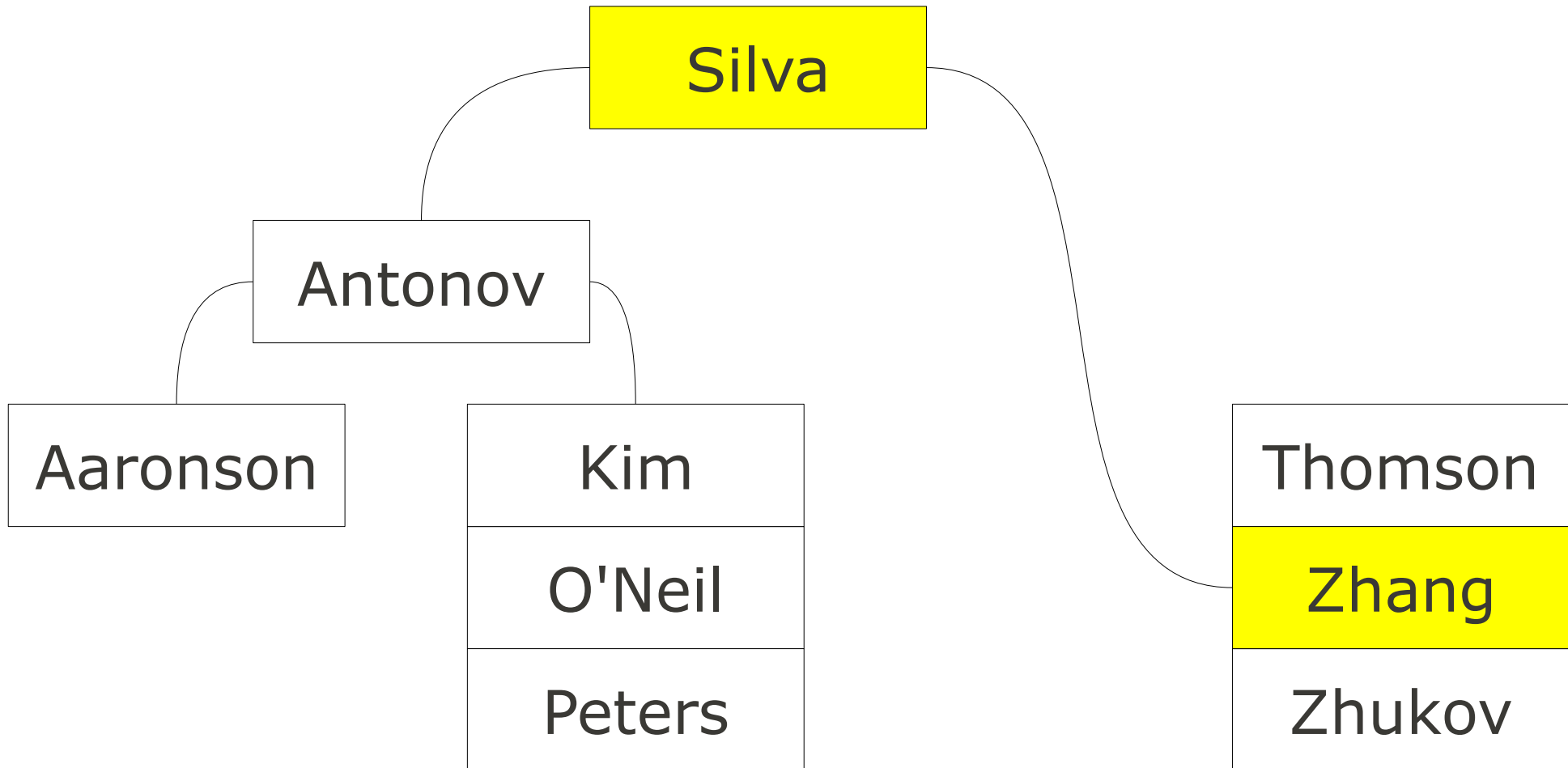
# Balanced Tree



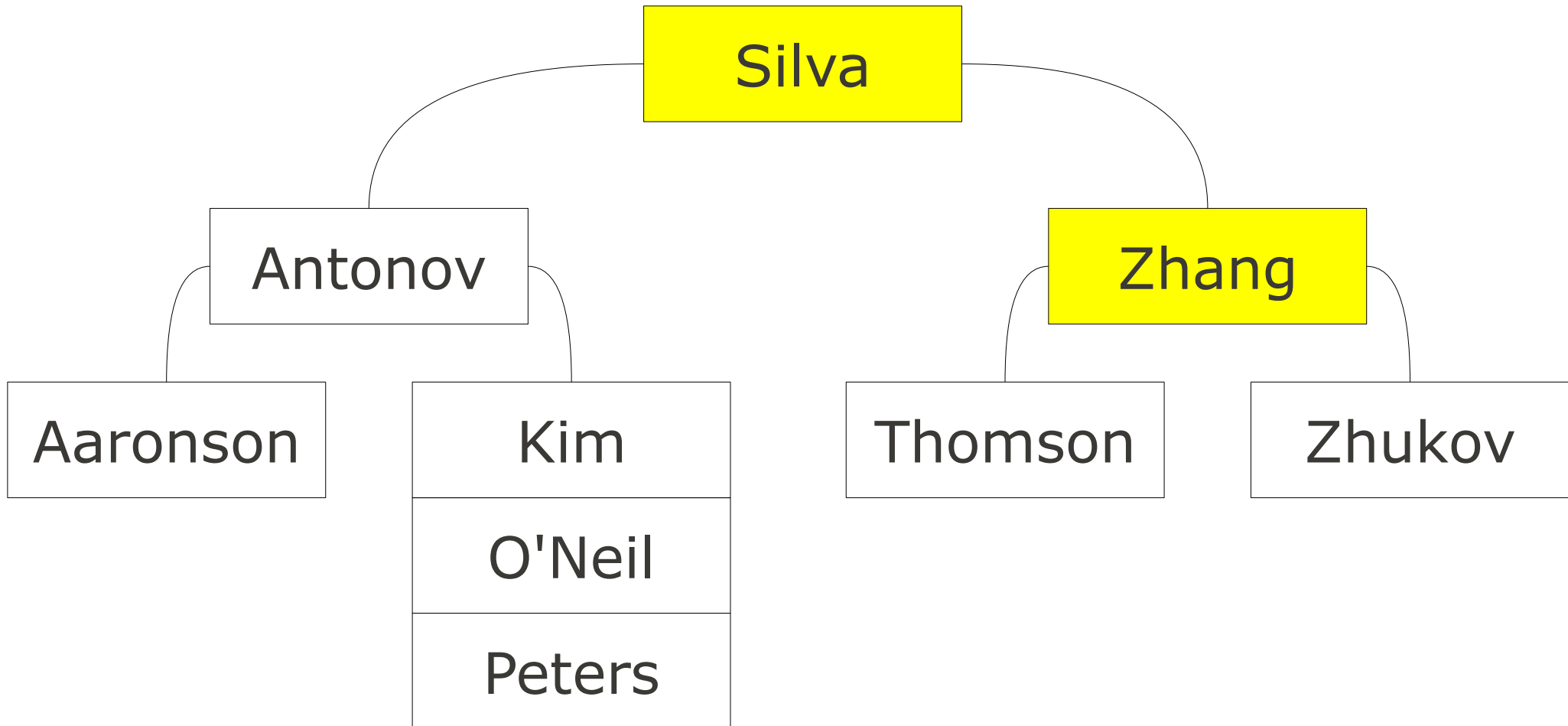
# Balanced Tree



# Balanced Tree



# Balanced Tree





# Storage Engines

## **CVS**

yes, it's an option

## **MyISAM**

the default  
relatively simple

See: `/var/lib/mysql/menagerie/`

## **InnoDB**

more features

# Storage Engines

## **CVS**

yes, it's an option

## **MyISAM**

the default  
relatively simple

## **InnoDB**

more features

# Storage Engines

## **CVS**

yes, it's an option

## **MyISAM**

the default  
relatively simple

## **InnoDB**

more features

# The Extra Features

## Foreign Key Constraints

no, 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=InnoDB;
```

# Index

## Easy retrieval by field

- usually automatic for PK
- optional for other fields

```
create index name_index  
on superhero (name) ;
```

to be continued