Database Storage Engines

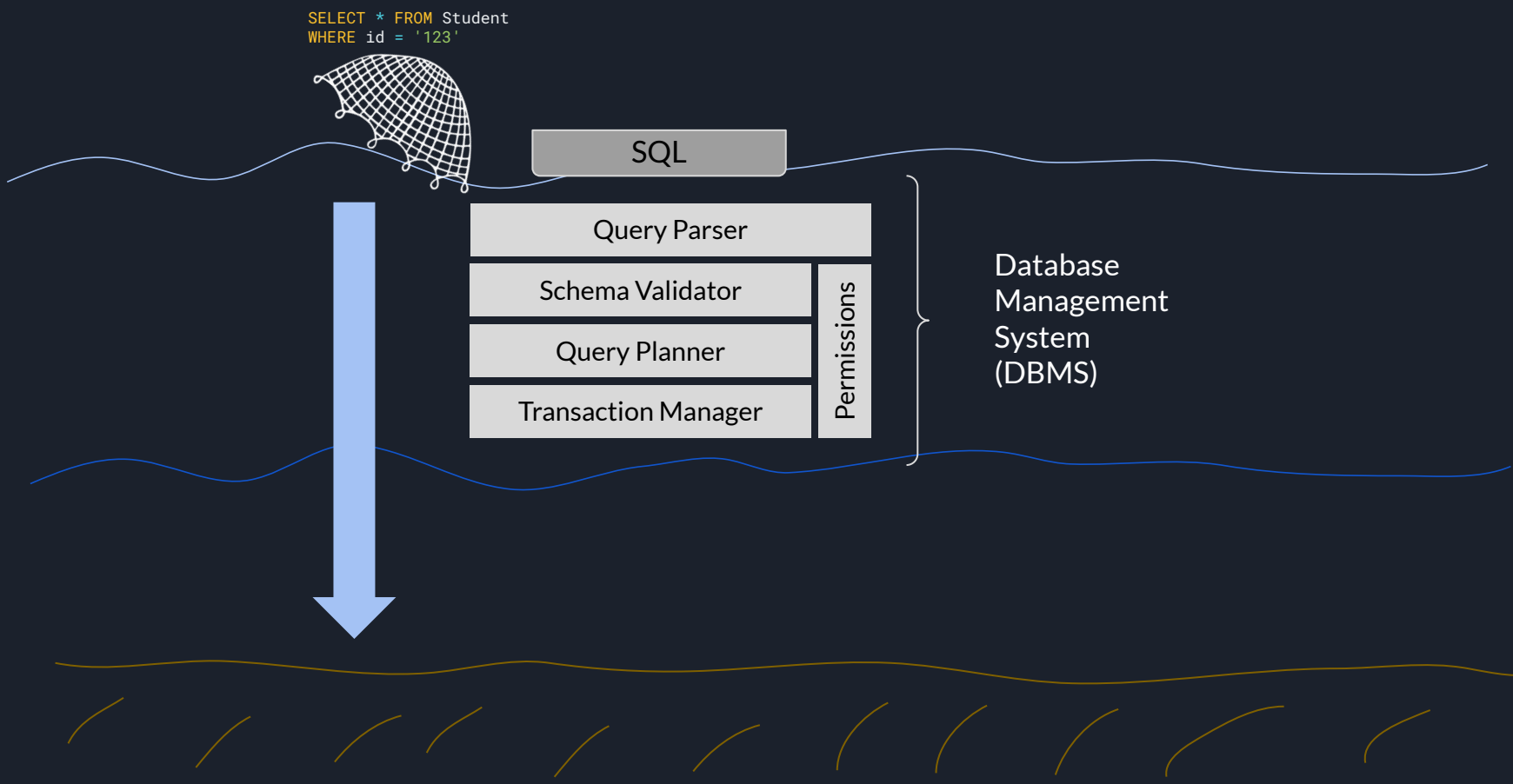
Martin Häusler, PhD
Consulting Architect @ IBM
Engineering Kiosk Alps, May 2026

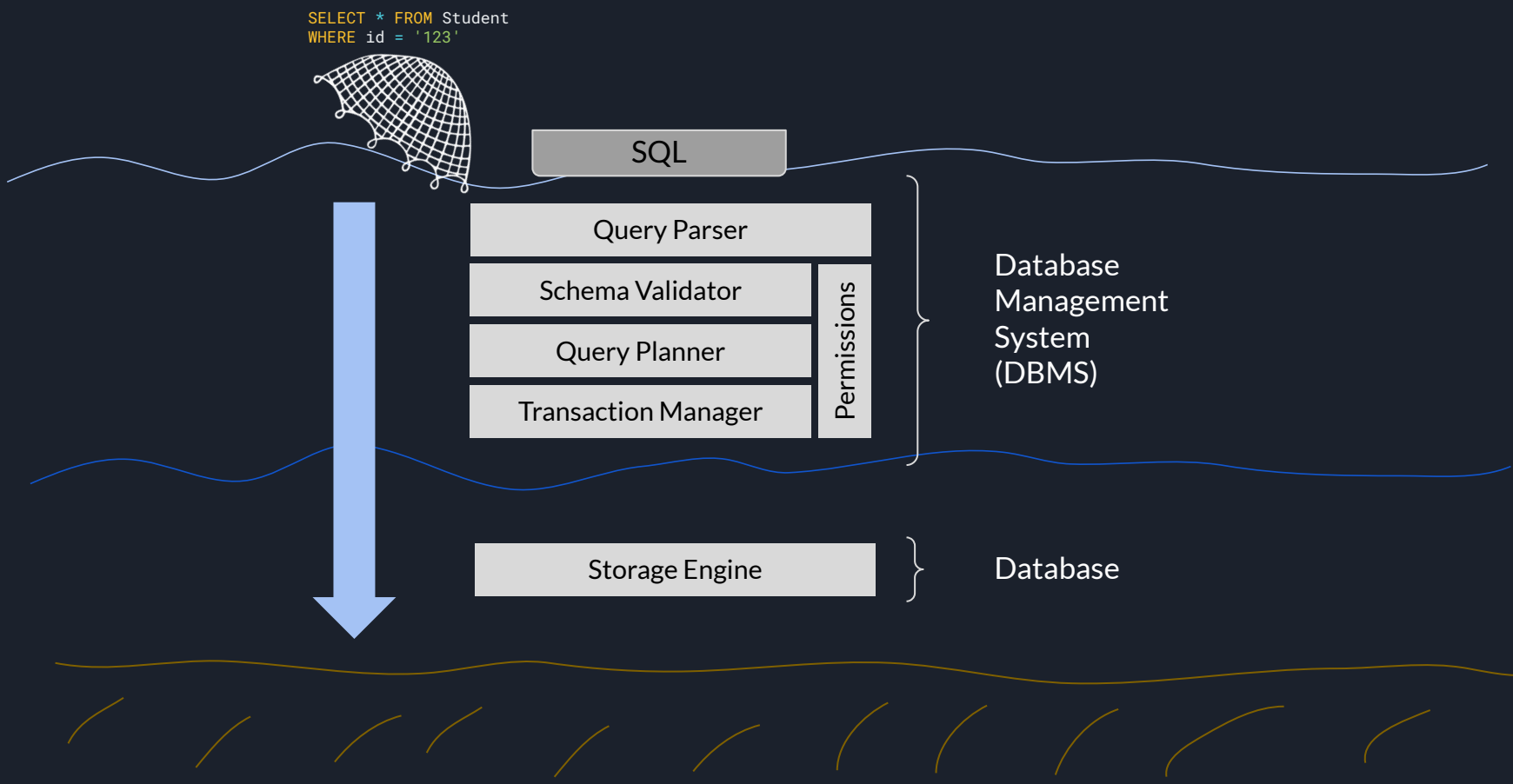
martin.haeusler89@gmail.com https://github.com/MartinHaeusler

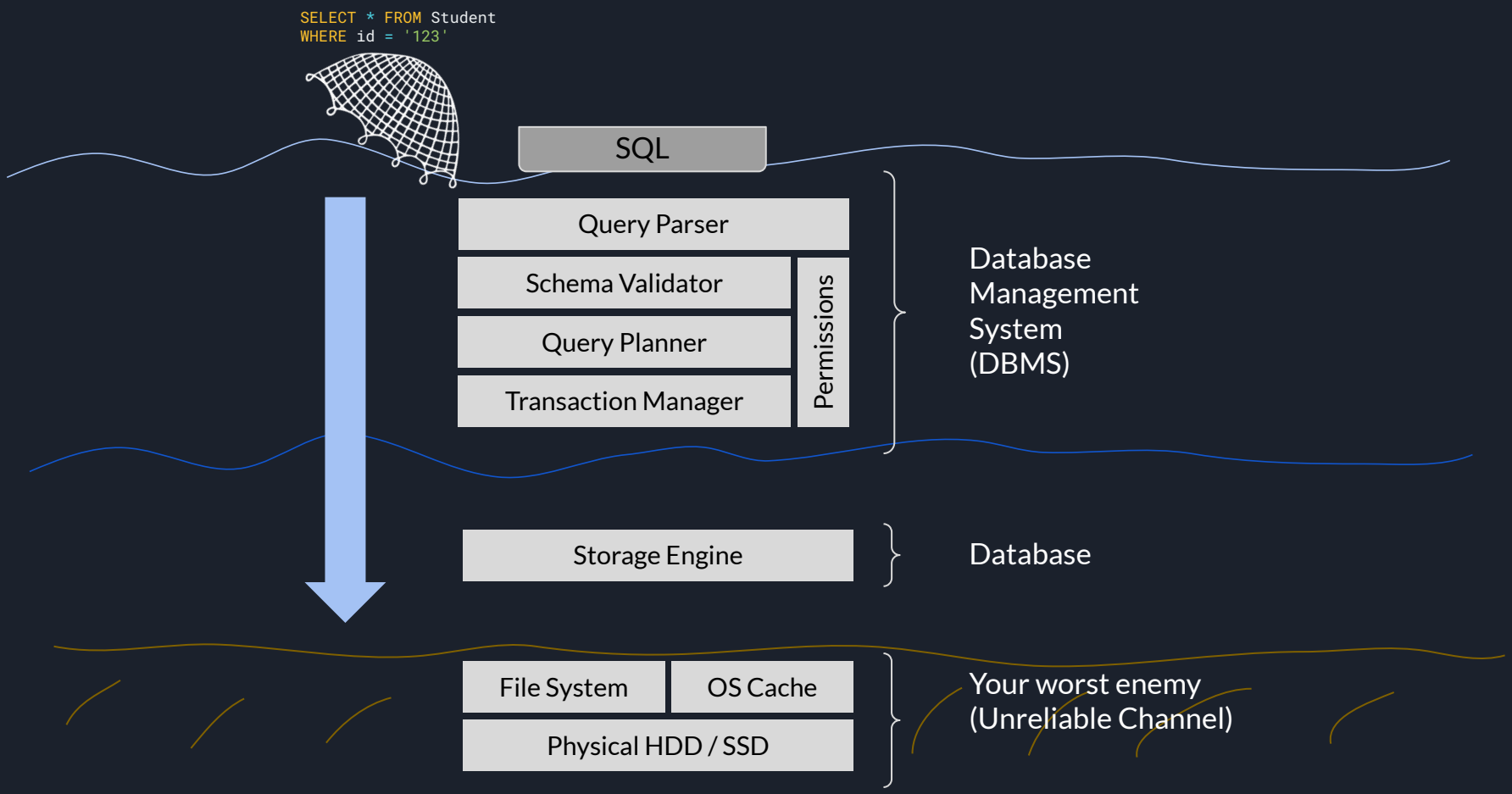

SQL

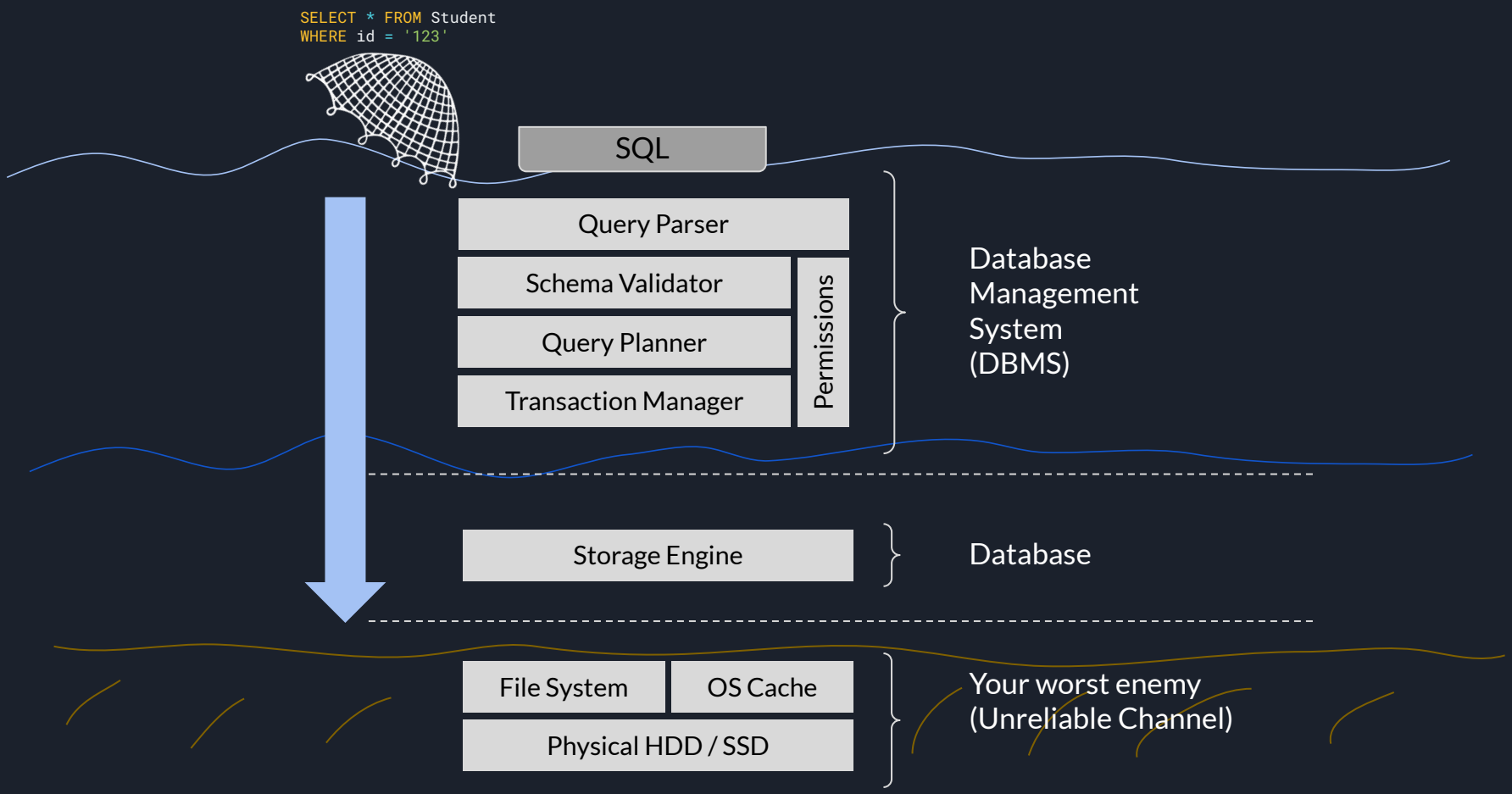

SELECT * FROM Student
WHERE id = '123'
SQL

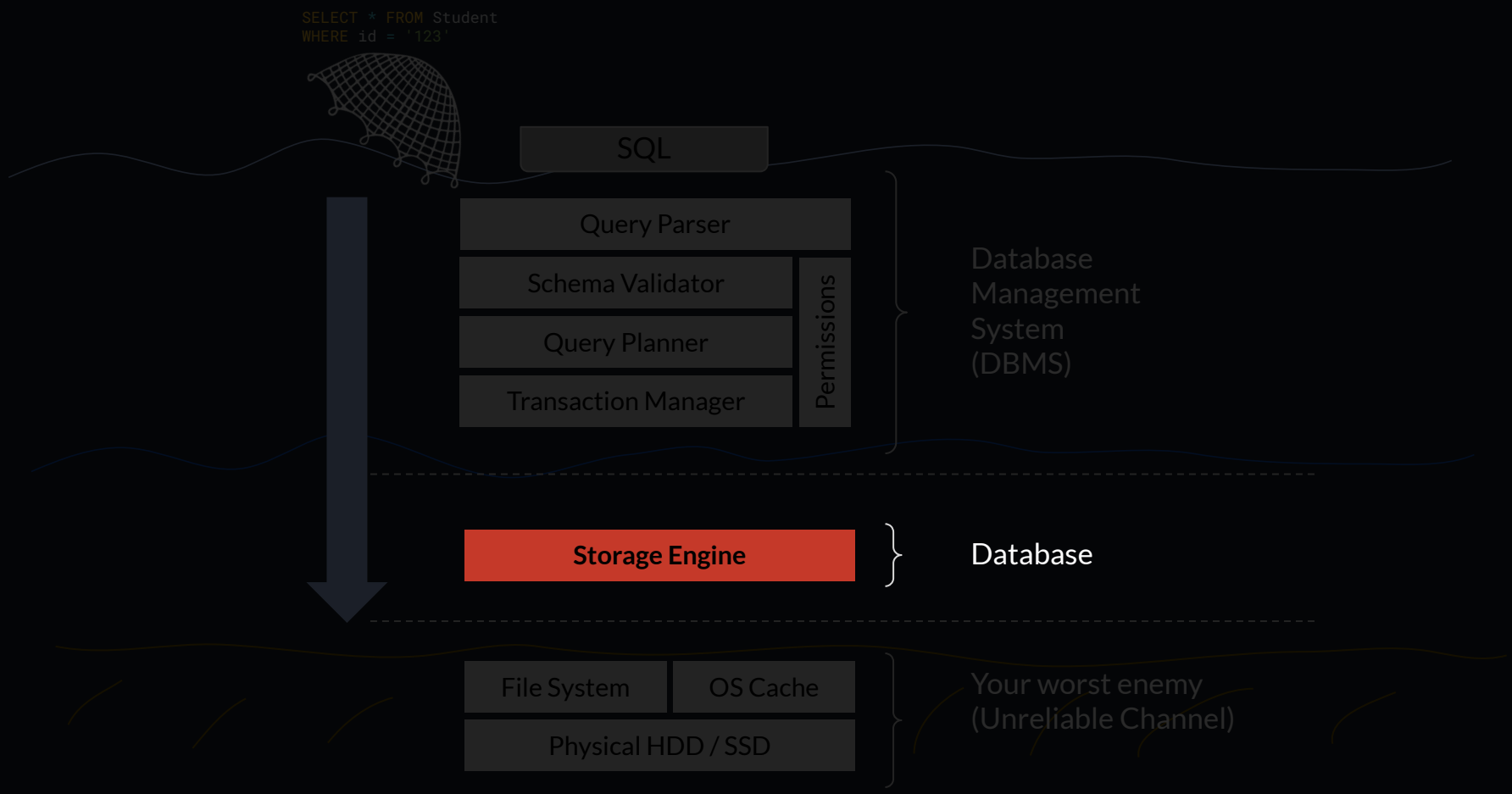

SELECT * FROM Student
WHERE id = '123'
SQL


SELECT * FROM Student
WHERE id = '123'
SQL
Query Parser
Schema Validator
Query Planner
Transaction Manager
Permissions
Database
Management
System
(DBMS)


SELECT * FROM Student
WHERE id = '123'
SQL
Query Parser
Schema Validator
Query Planner
Transaction Manager
Permissions
Database Management System (DBMS)
Storage Engine
Database


SELECT * FROM Student
WHERE id = '123'
SQL
Query Parser
Schema Validator
Query Planner
Transaction Manager
Permissions
Database Management System (DBMS)
Storage Engine
Database
File System
OS Cache
Physical HDD / SSD
Your worst enemy (Unreliable Channel)


SELECT * FROM Student
WHERE id = '123'
SQL
Query Parser
Schema Validator
Query Planner
Transaction Manager
Permissions
Database Management System (DBMS)
Storage Engine
Database
File System
OS Cache
Physical HDD / SSD
Your worst enemy (Unreliable Channel)


SELECT * FROM Student
WHERE id = '123'
SQL
Query Parser
Schema Validator
Query Planner
Transaction Manager
Permissions
Database Management System (DBMS)
Storage Engine
Database
File System
OS Cache
Physical HDD / SSD
Your worst enemy (Unreliable Channel)

Tables and other pretty little lies

id	firstname	lastname	birthdate
1	John	Doe	01.01.1970
2	Jane	Doe	23.02.1972
3	Jack	Smith	12.05.2006
4	Jill	Valentine	13.03.1998

Tables and other pretty little lies

id	firstname	lastname	birthdate
1	John	Doe	01.01.1970
2	Jane	Doe	23.02.1972
3	Jack	Smith	12.05.2006
4	Jill	Valentine	13.03.1998

Table: 2 Dimensions

Tables and other pretty little lies

id	firstname	lastname	birthdate
1	John	Doe	01.01.1970
2	Jane	Doe	23.02.1972
3	Jack	Smith	12.05.2006
4	Jill	Valentine	13.03.1998

Table: 2 Dimensions

Disk: 1 Dimension

Tables and other pretty little lies

id	firstname	lastname	birthdate
1	John	Doe	01.01.1970
2	Jane	Doe	23.02.1972
3	Jack	Smith	12.05.2006
4	Jill	Valentine	13.03.1998

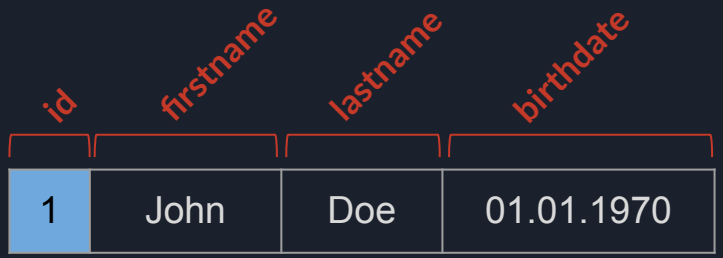
Table: 2 Dimensions

Disk: 1 Dimension

Record

Record

Record



1	{ "firstname": "John", "lastname": "Doe", "birthdate": "01.01.1970" }

Record

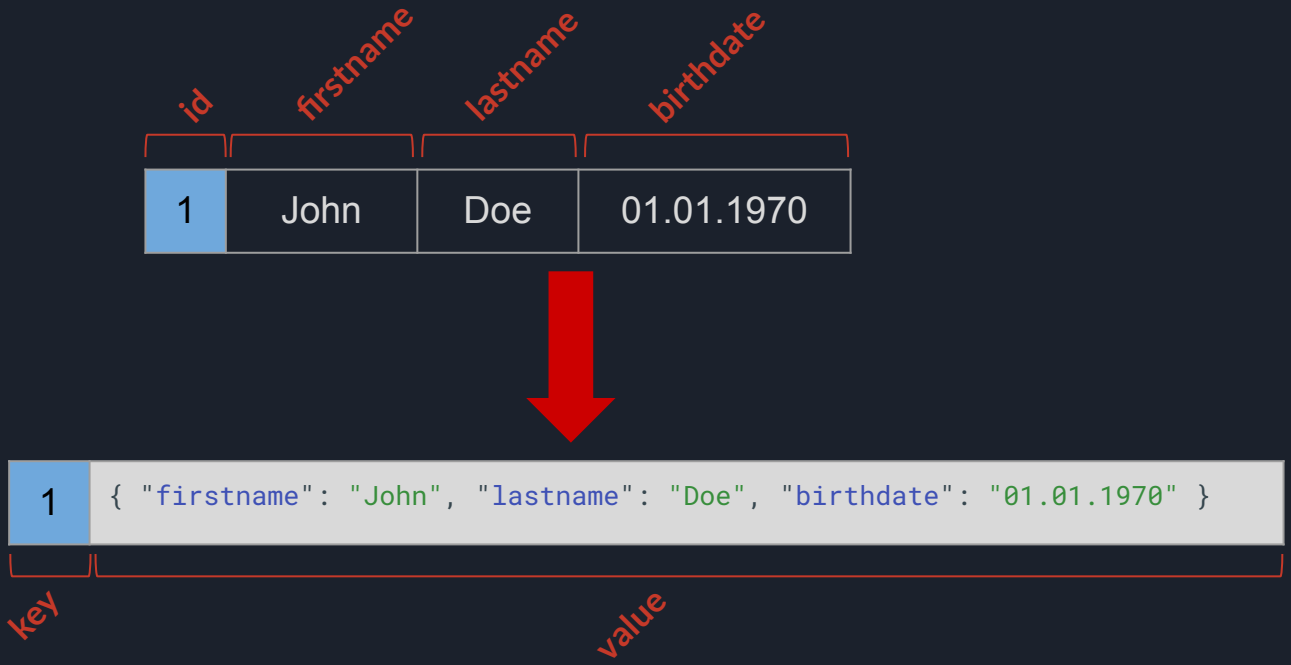
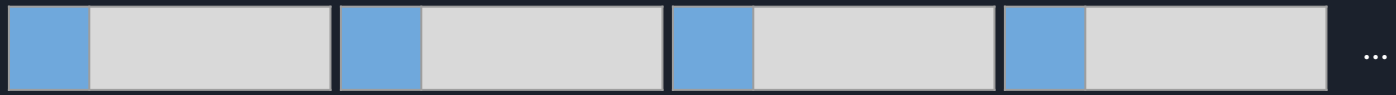

id
firstname
lastname
birthdate
1
John
Doe
01.01.1970
1
{ "firstname": "John", "lastname": "Doe", "birthdate": "01.01.1970" }
key
value

Table = List of Records



Every Database comes down to a list of Key-Value pairs.


SO IT'S ALL JUST
LISTS OF KEY-VALUE PAIRS?
ALWAYS HAS BEEN.

Key Value Store: Where Key-Value Pairs live

```
interface KeyValueStore {



}
```

Key Value Store: Where Key-Value Pairs live

```
interface KeyValueStore {

    void put(byte[] key, byte[] value);



}
```

Key Value Store: Where Key-Value Pairs live

```
interface KeyValueStore {

    void put(byte[] key, byte[] value);

    byte[] get(byte[] key);


}
```

Key Value Store: Where Key-Value Pairs live

```
interface KeyValueStore {
    void put(byte[] key, byte[] value);
    byte[] get(byte[] key);
    Iterator<byte[]> iterator();
}
```

Key Value Store: Where Key-Value Pairs live

```
interface KeyValueStore {
    void put(byte[] key, byte[] value);
    byte[] get(byte[] key);
    Iterator<byte[]> iterator();
}
```

* This is the BARE minimum interface. Practical implementations have more methods.

Simplifying Assumptions on Slides:

- Slides only show keys to reduce visual clutter
- Values can always be “carried along” (e.g. via Suffix Encoding)
- Keys are *actually* byte arrays…
- … but we will use strings or numbers for visualization


LET'S
PLAY

Is “walk” in the list?

```
able
time
walk
bake
```

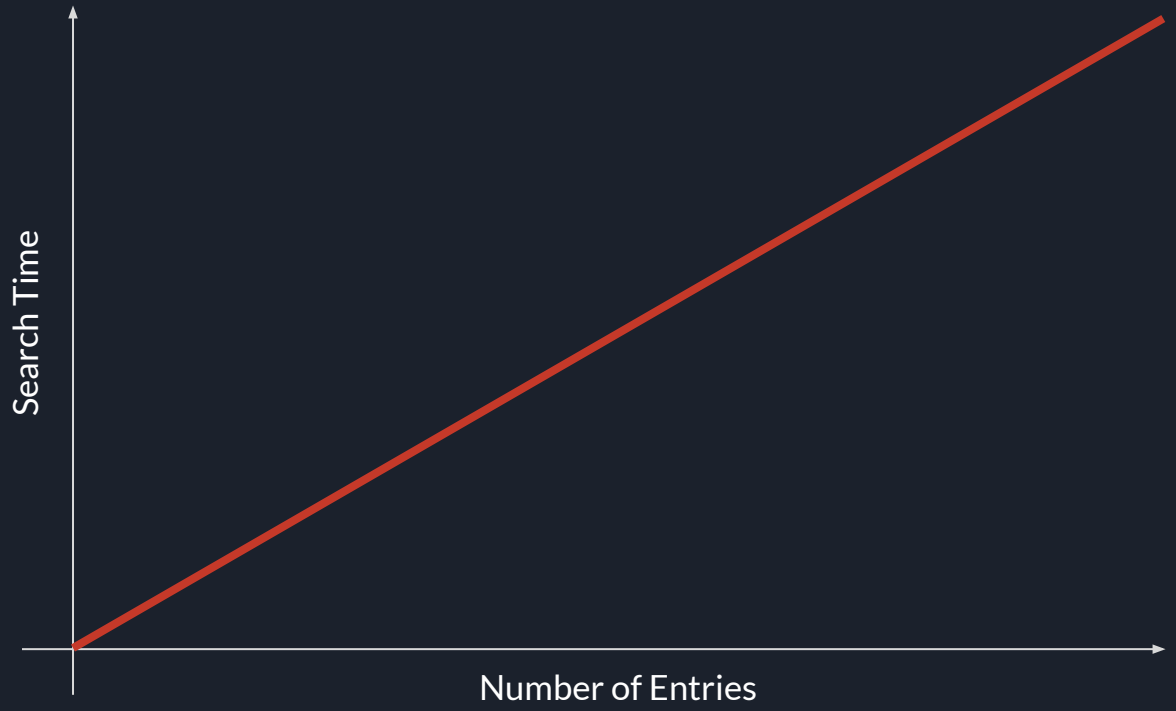
Is “walk” in the list?

```
hand
gold
xray
unit
keen
face
walk
idea
zero
over
```

Is “walk” in the list?

same	each	xmas	yoga	mark	hand
upon	unit	area	ship	jazz	user
ball	fact	main	wind	vast	keen
year	true	time	xray	name	fair
rain	cake	join	quad	bake	city
path	gear	team	leaf	view	pack
said	dial	safe	high	lamp	acid
door	fast	vote	jump	iron	kick
oven	help	gold	neck	xyst	icon
desk	blue	zero	baby	idle	oral
over	gate	card	real	edge	data
wave	made	zinc	node	joke	idea
wall	tall	vain	easy	page	road
open	near	kind	keep	read	lake
tall	calm	walk	echo	part	quiz
urge	yard	quit	mind	zone	life
game	hard	face	atom	able	

General Problem: Searching is O(N)



WORK SMARTER, NOT HARDER

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	**oven**	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “oven” before or after “walk”?

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	**oven**	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “oven” before or after “walk”?

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	**tall**	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “tall” before or after “walk”?

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	**tall**	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “tall” before or after “walk”?

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	**wave**
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “wave” before or after “walk”?

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	**wave**
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “wave” before or after “walk”?

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	**user**	zone
each	idea	mark	rain	vain	

Is “user” before or after “walk”?

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	**user**	zone
each	idea	mark	rain	vain	

Is “user” before or after “walk”?

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “vote” before or after “walk”?

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	**vote**
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “vote” before or after “walk”?

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	**wall**
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “wall” before or after “walk”?

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “walk” in the list?

able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	**walk**
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Is “walk” equal to “walk”?

Is “walk” in the list?

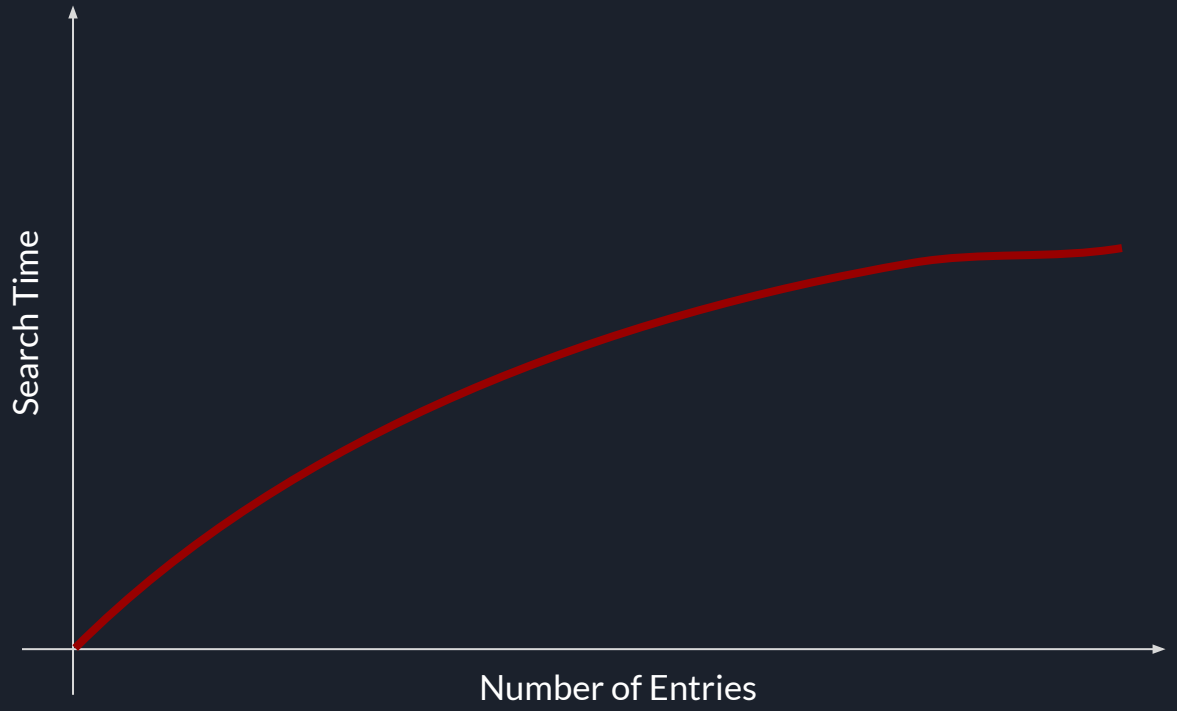
able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	**walk**
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Our Search

100 List Items

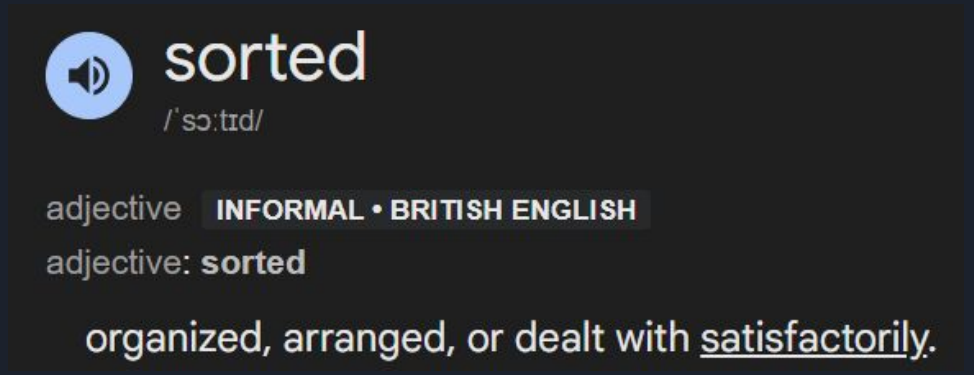
7 Comparisons

Searching in *sorted* list is O(log(N))



sorted

sorted


sorted
/ˈsɔːtɪd/
adjective
INFORMAL • BRITISH ENGLISH
adjective: sorted
organized, arranged, or dealt with satisfactorily.

So we want our data **sorted**...

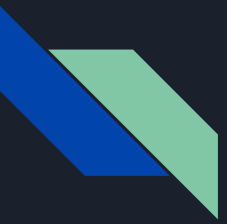
So we want our data **sorted**...

... to be able to search in **O(log(N))**...

So we want our data **sorted**...

... to be able to search in **O(log(N))**...

... but **sorting** the data in the first place...

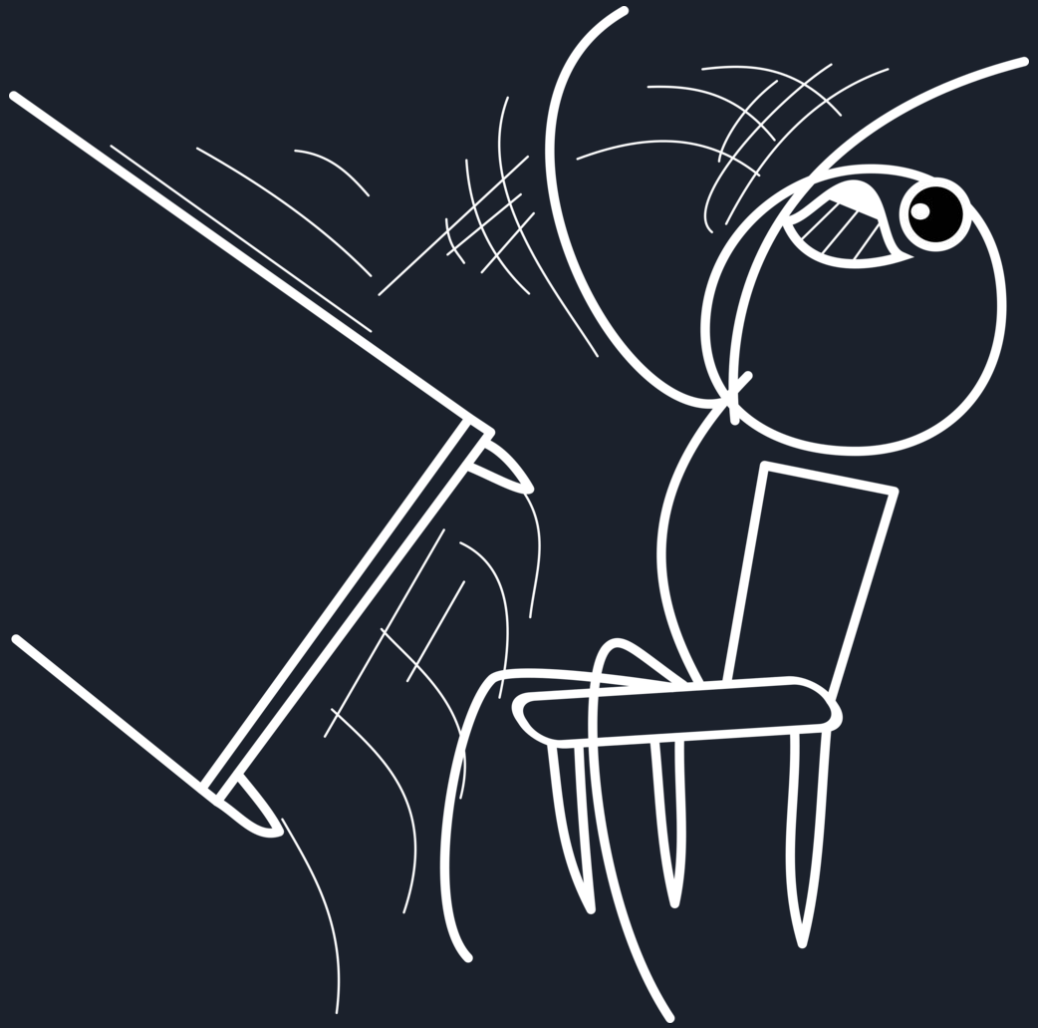


So we want our data **sorted**...

... to be able to search in **O(log(N))**...

... but **sorting** the data in the first place...

... is **O(n * log(N))**



The Saving Grace

The Saving Grace

The Saving Grace

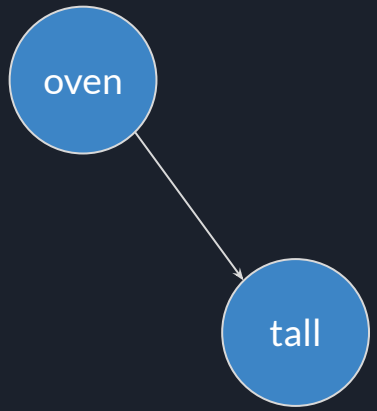
Idea:

- Writes pay the cost of organization
- Reads should be as cheap as possible

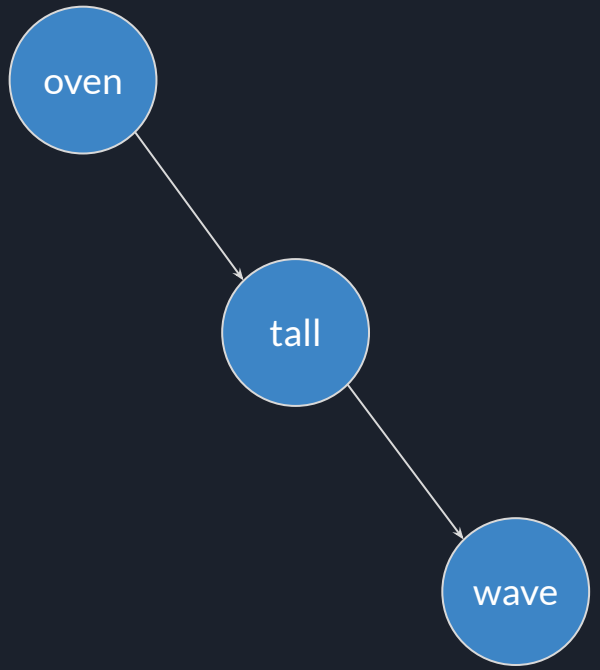
Idea:

- Start with an empty list
- As new data comes in…
- …we sort it in…
- … so that the new state is again sorted!

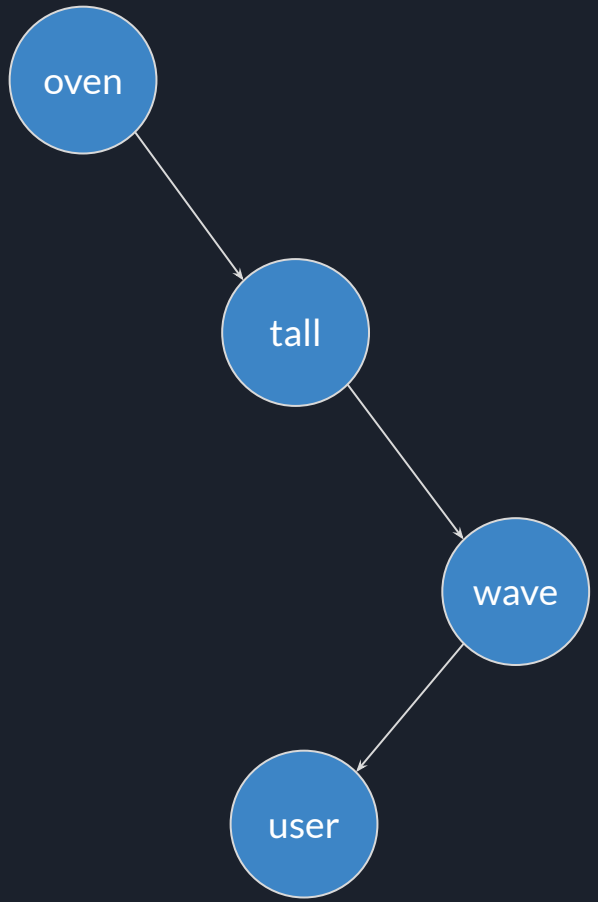
able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	**oven**	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	



able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	**tall**	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

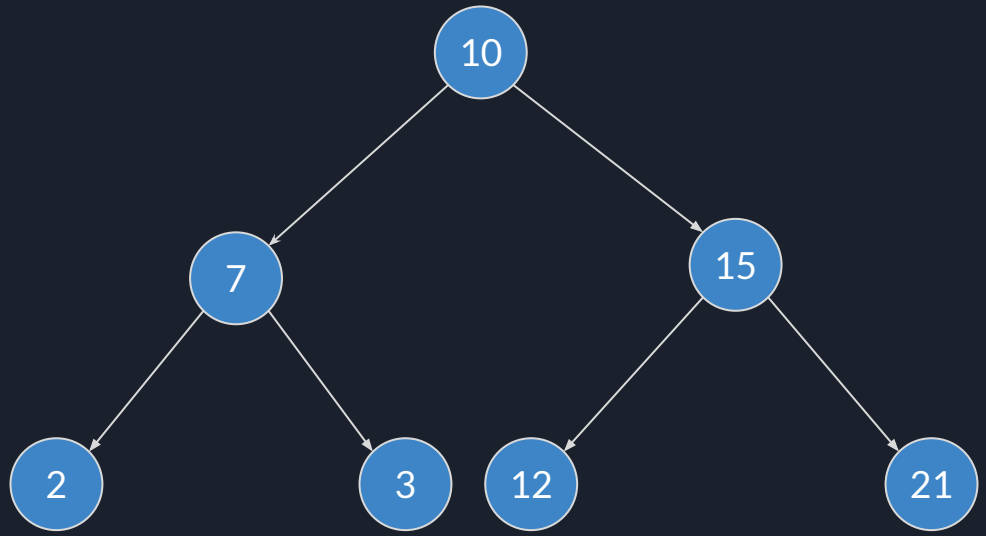


able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

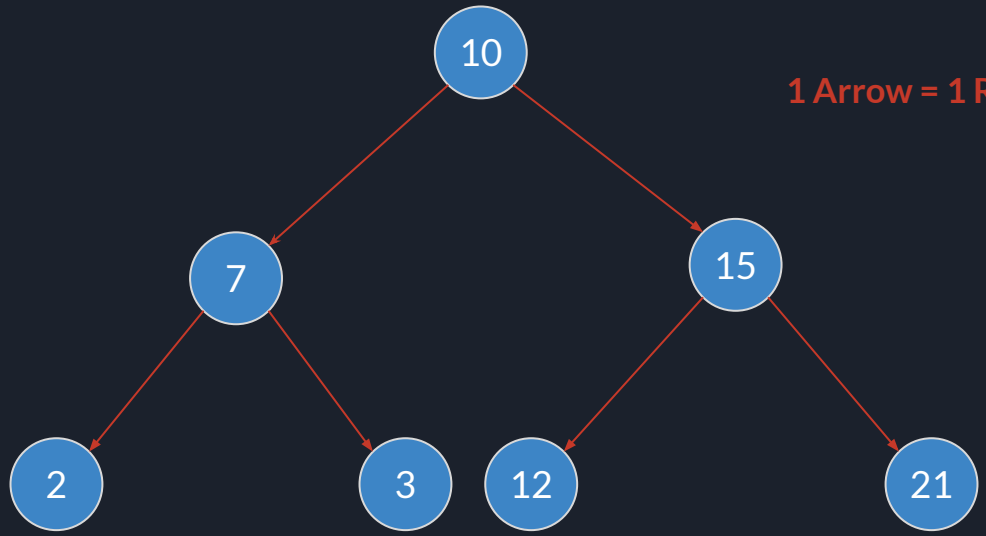


able	easy	idle	mind	read	vast
acid	echo	iron	name	real	view
area	edge	jazz	near	road	vote
atom	face	join	neck	safe	walk
baby	fact	joke	node	said	wall
bake	fair	jump	open	same	wave
ball	fast	keen	oral	ship	wind
blue	game	keep	oven	tail	xmas
cake	gate	kick	over	tall	xray
calm	gear	kind	pack	team	xyst
card	gold	lake	page	time	yard
city	hand	lamp	part	true	year
data	hard	leaf	path	unit	yoga
desk	help	life	quad	upon	zero
dial	high	made	quit	urge	zinc
door	icon	main	quiz	user	zone
each	idea	mark	rain	vain	

Binary Search Tree



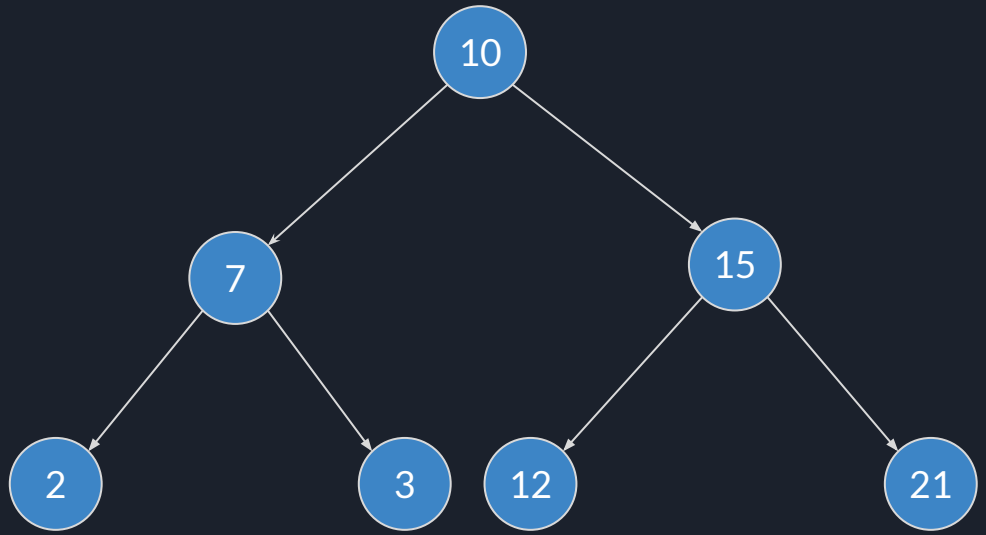
Binary Search Tree



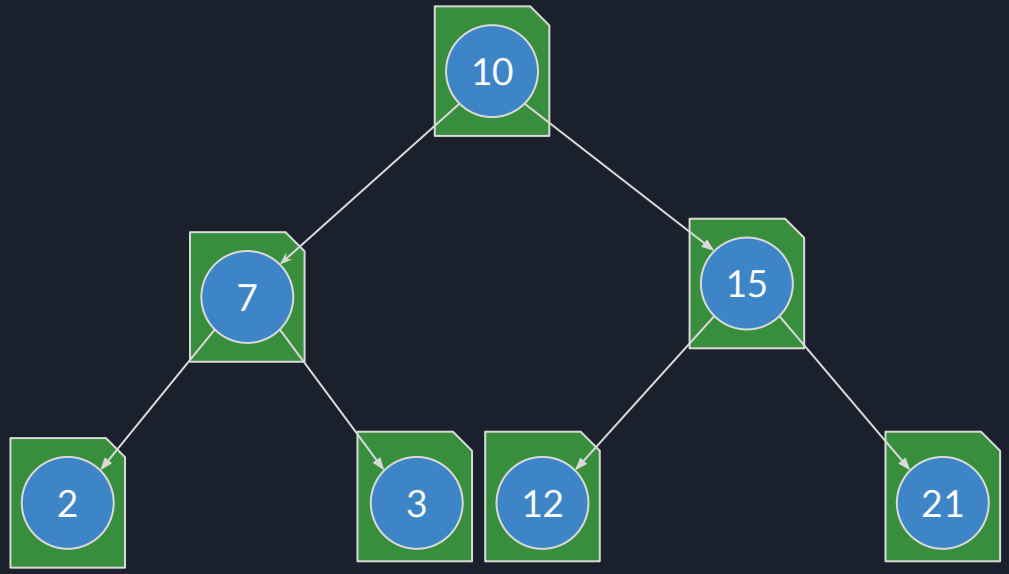
1 Arrow = 1 Random Lookup on disk

How do you save a tree structure to disk?

How do you save a tree structure to disk?

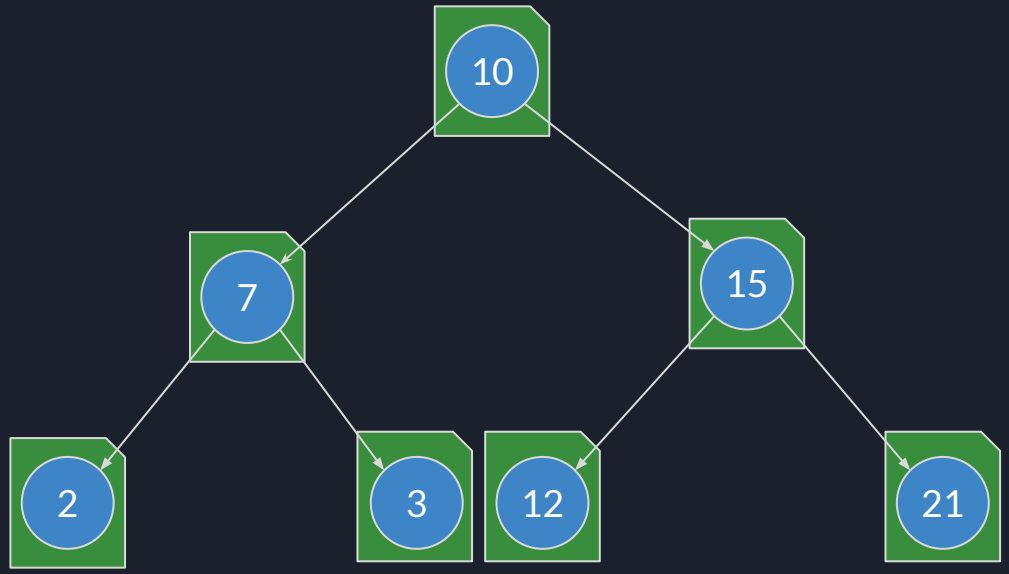


How do you save a tree structure to disk?



“Page” of Memory
Usually 1 KB ~ 4 KB

How do you save a tree structure to disk?

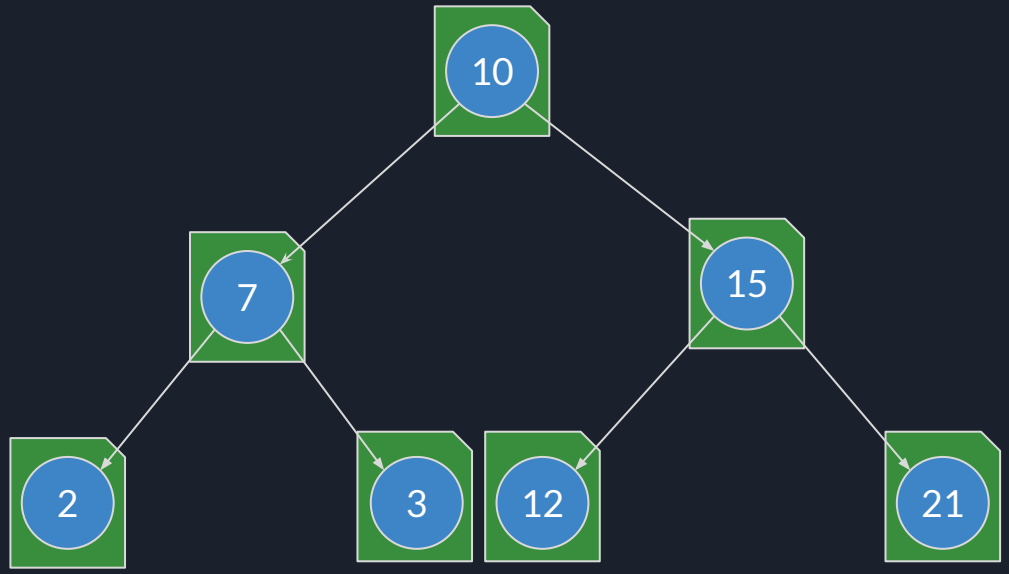


"Page" of Memory
Usually 1 KB ~ 4 KB

Simplifying Assumptions:

- 1 Page = 1 file on disk
- 1 Page = 1 node in the tree

How do you save a tree structure to disk?



"Page" of Memory
Usually 1 KB ~ 4 KB

Simplifying Assumptions:

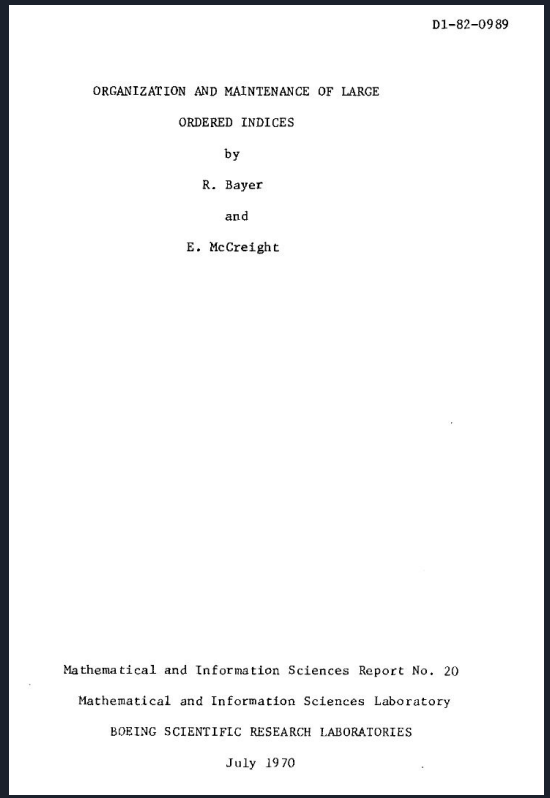
- 1 Page = 1 file on disk
- 1 Page = 1 node in the tree

Page Manager keeps track of loaded pages and caches them in memory.

Modified pages become "pinned" (flagged as "dirty") and need to be written to disk before unpinning.

1970

1970: Rudolf Bayer hits the stage



D1-82-0989

ORGANIZATION AND MAINTENANCE OF LARGE
ORDERED INDICES

by

R. Bayer

and

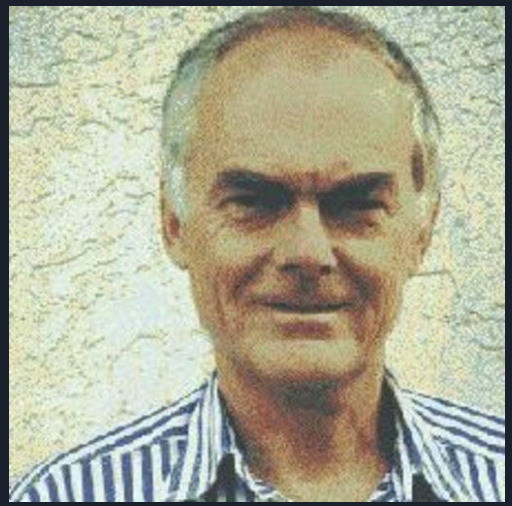
E. McCreight

Mathematical and Information Sciences Report No. 20

Mathematical and Information Sciences Laboratory

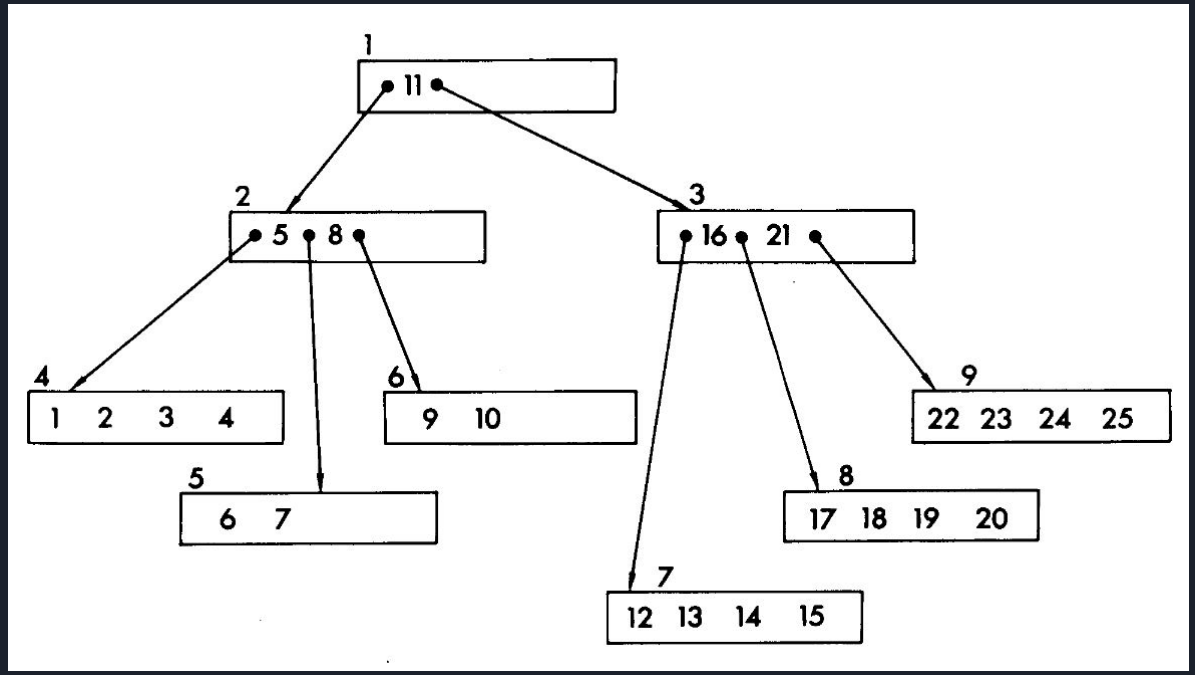
BOEING SCIENTIFIC RESEARCH LABORATORIES

July 1970



Rudolf Bayer
Technical University of Munich

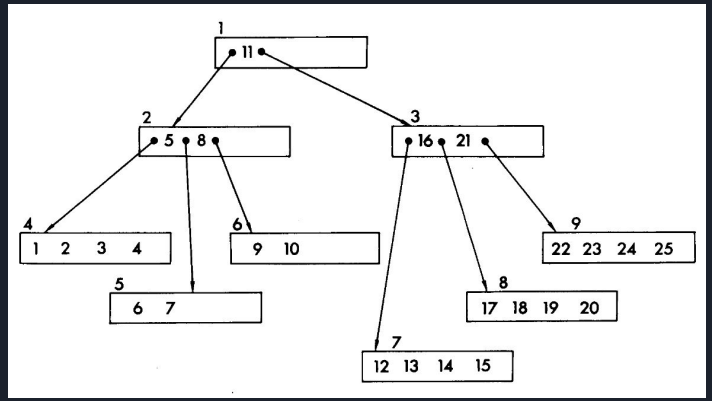
The B-Tree


1
11
2
5 8
3
16 21
4
1 2 3 4
6
9 10
9
22 23 24 25
5
6 7
8
17 18 19 20
7
12 13 14 15

The B-Tree

Core idea by Rudolf Bayer:

- Store not just one, but a list of *B* entries per node
  - Binary search tree is special case with *B* = 2
- Entries in the **same node** are stored in **the same page**
  - No further disk lookups are required to check those
- Higher value of *B*
  - means tree height gets reduced
  - means fewer pages need to be loaded per search
  - means faster lookups
  - typical value in practice is *B* = 8




GREAT SUCCESS!
quickmeme.com

The Ubiquitous B-Tree

1979

DOUGLAS COMER

Computer Science Department, Purdue University, West Lafayette, Indiana 47907

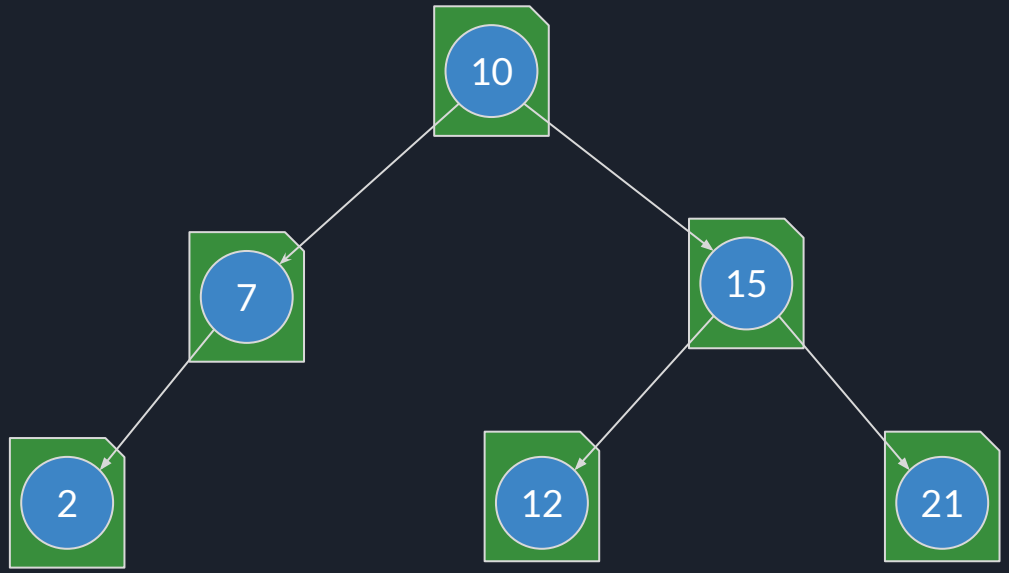
B-trees have become, de facto, a standard for file organization. File indexes of users, dedicated database systems, and general-purpose access methods have all been proposed and implemented using B-trees. This paper reviews B-trees and shows why they have been so successful. It discusses the major variations of the B-tree, especially the B^+-tree, contrasting the relative merits and costs of each implementation. It illustrates a general purpose access method which uses a B-tree.

Keywords and Phrases: B-tree, B^*-tree, B^+-tree, file organization, index

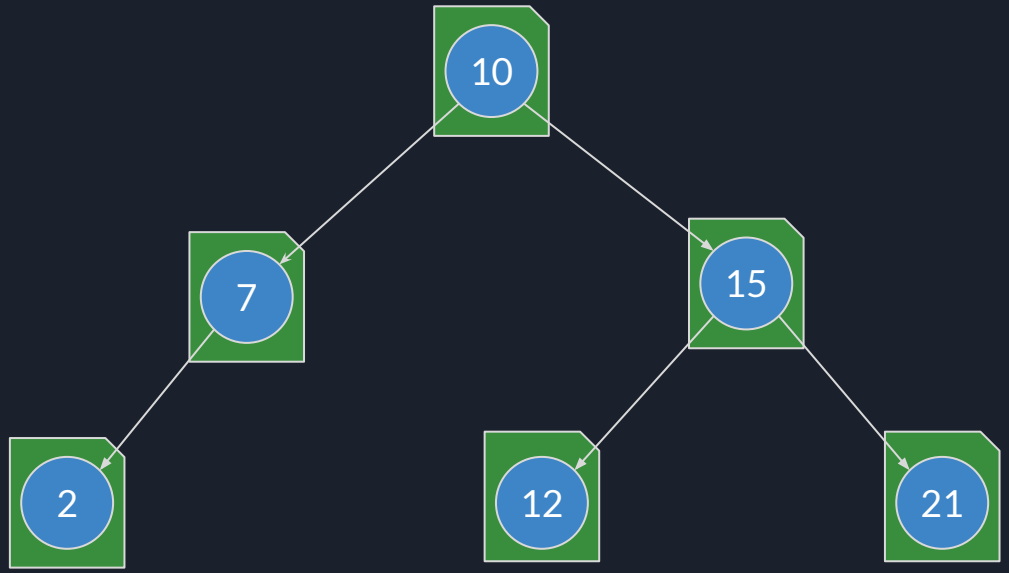
CR Categories: 3.73 3.74 4.33 4 34

HOWEVER.

Binary Search Tree



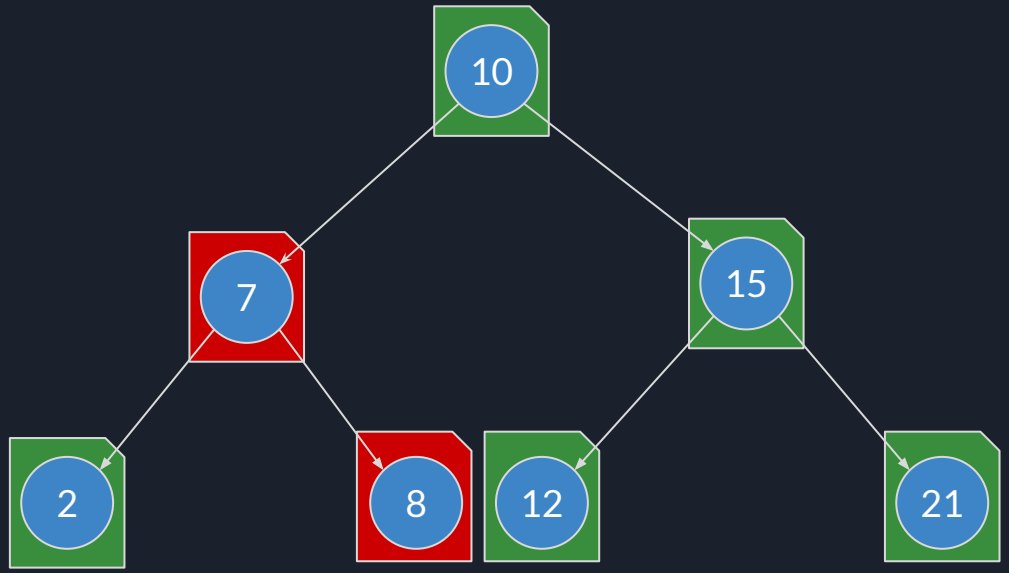
Binary Search Tree



Command:

Insert 8.

Binary Search Tree

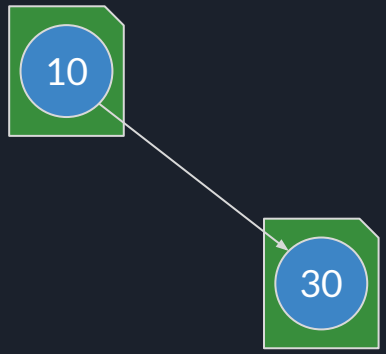


Command:

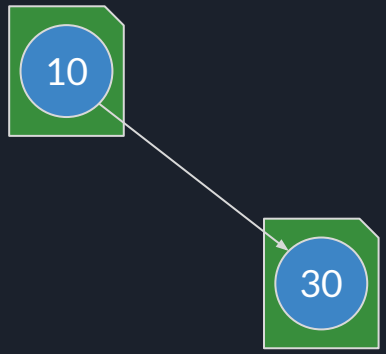
Insert 8.

Dirty Pages:
[7, 8]

Binary Search Tree

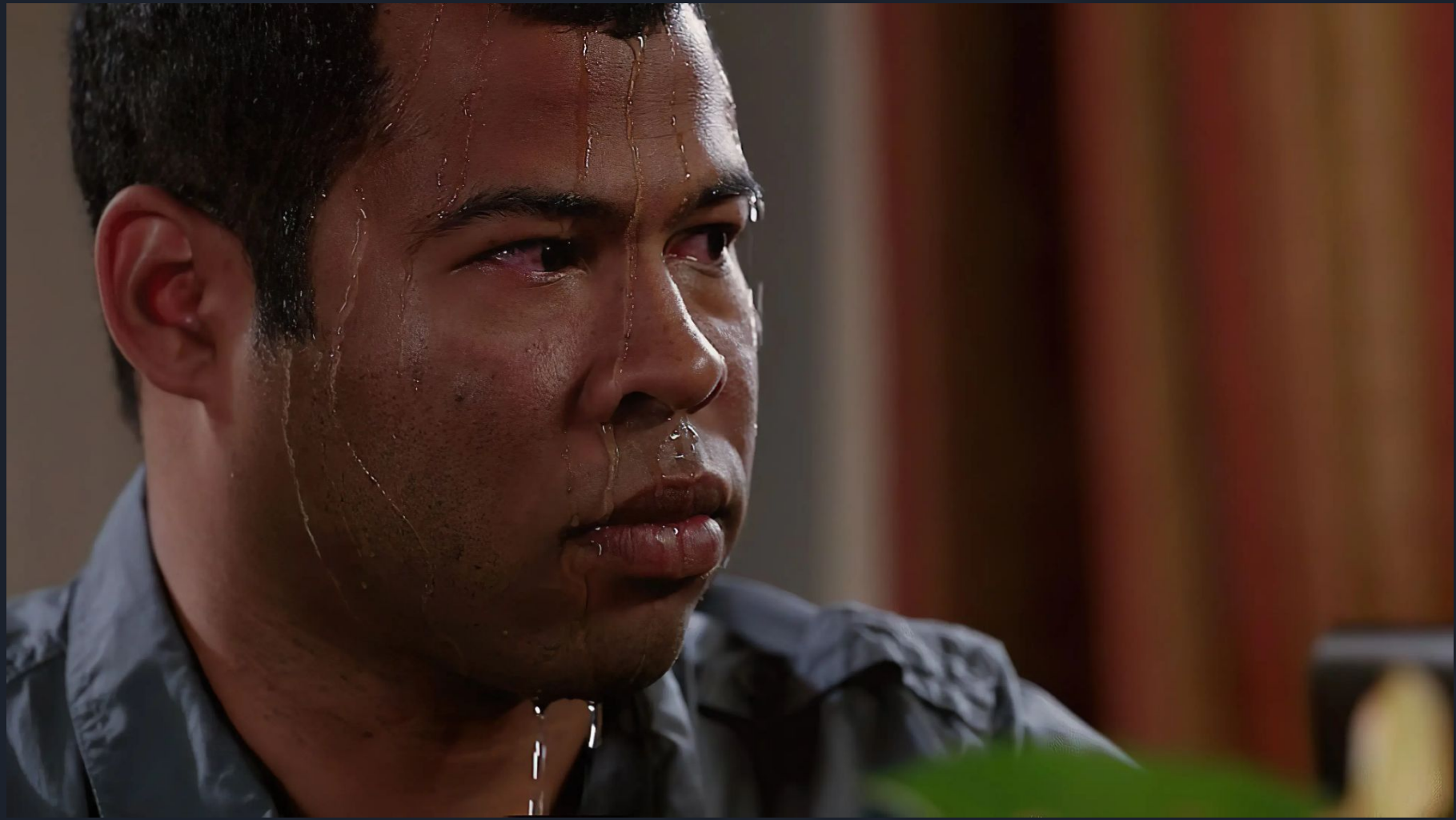


Binary Search Tree

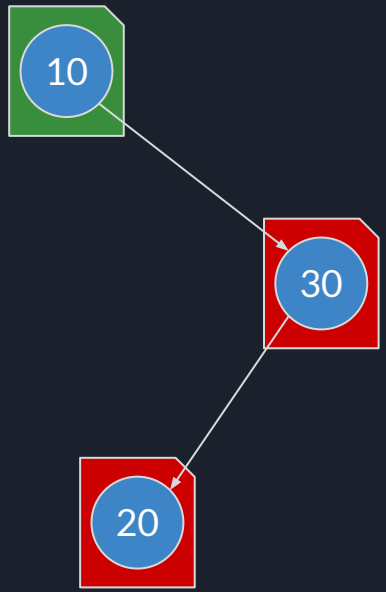


Command:

Insert 20.



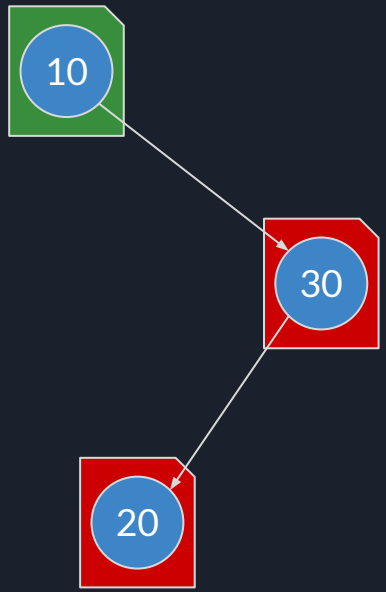
Binary Search Tree



Command:

Insert 20.

Binary Search Tree



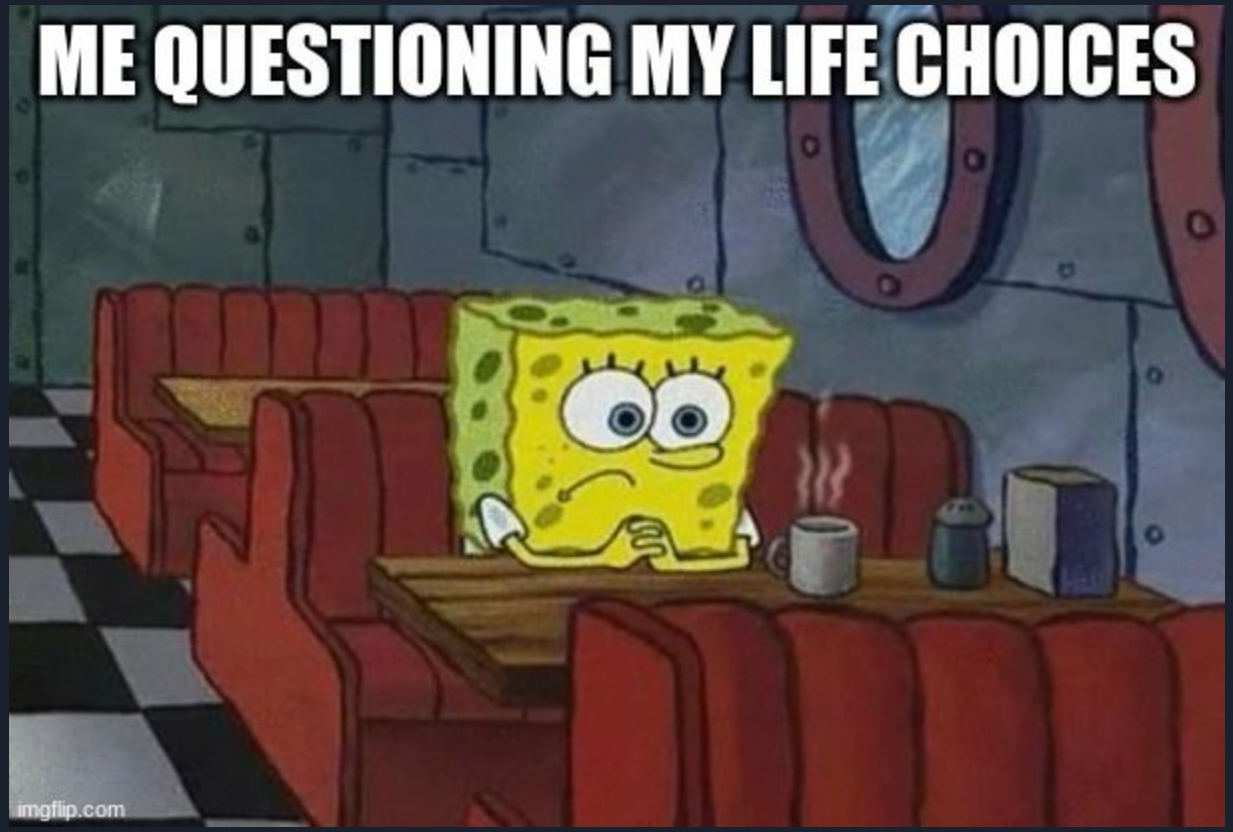
Command:

Insert 20.

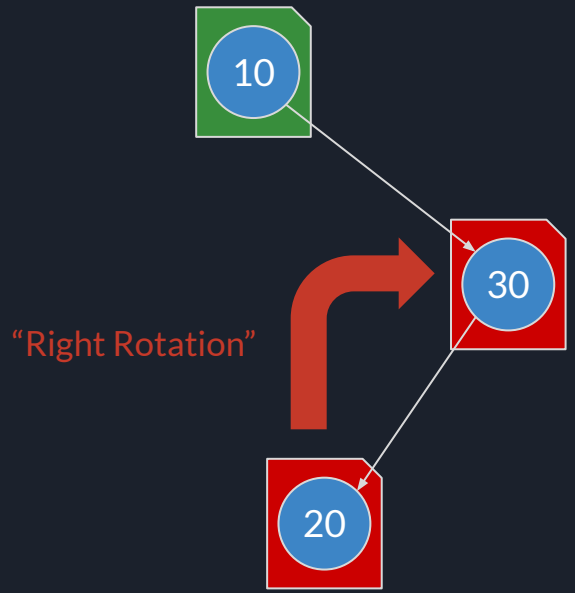
Tree is unbalanced!

If we keep going like this, it will degrade into a linked list!

We have to re-balance!


ME QUESTIONING MY LIFE CHOICES
imgflip.com

Binary Search Tree



Command:

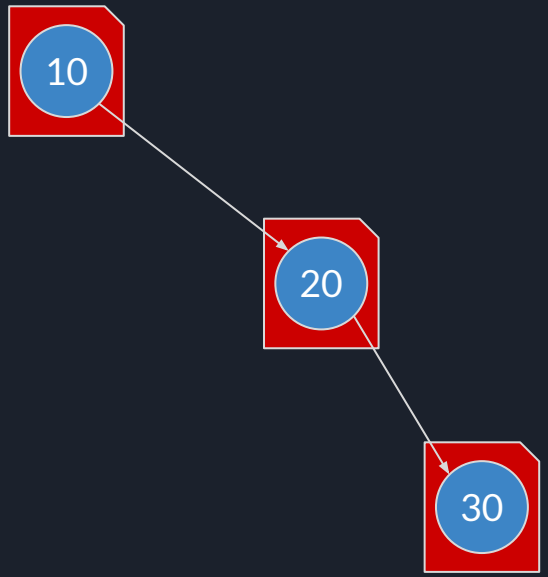
Insert 20.

Tree is unbalanced!

If we keep going like this, it will degrade into a linked list!

We have to re-balance!

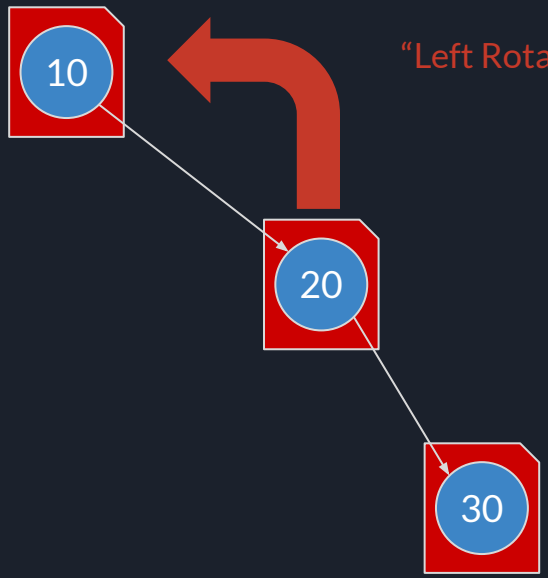
Binary Search Tree



Command:

Insert 20.

Binary Search Tree

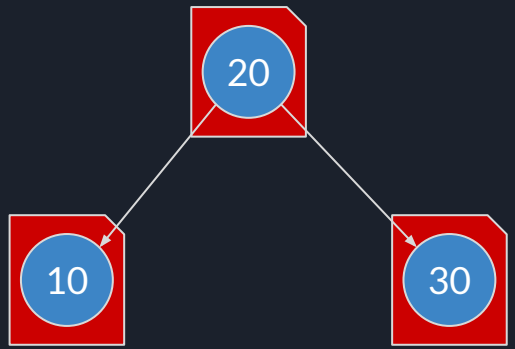


“Left Rotation”

Command:

Insert 20.

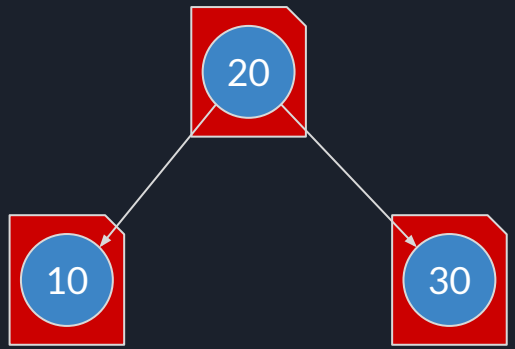
Binary Search Tree



Command:

Insert 20.

Binary Search Tree



Command:

Insert 20.

We managed to **rebalance** the tree…

… but **ALL pages are dirty**!

A **single** insert turned into **many** disk writes.

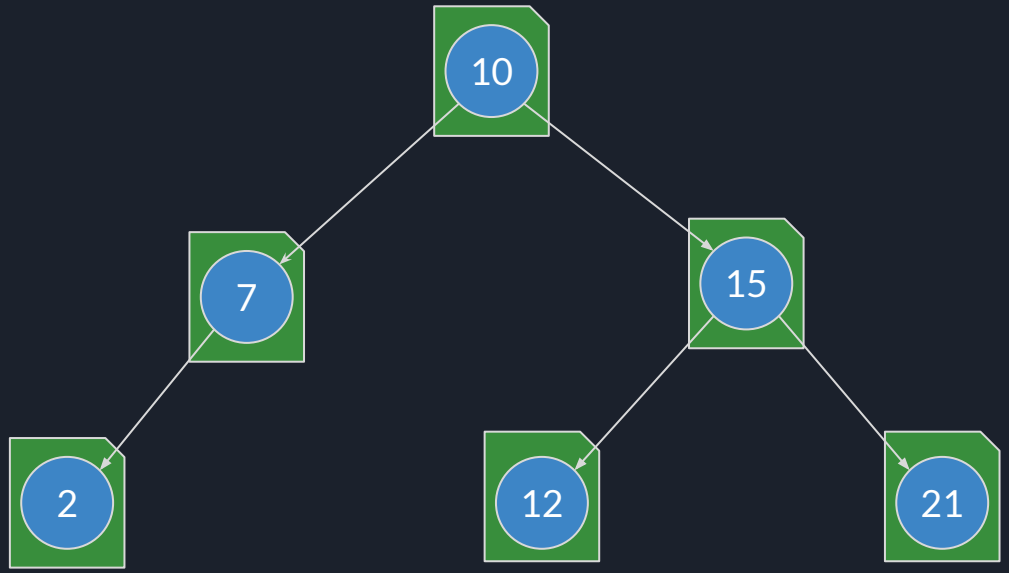
This is called **High Write Amplification**

Problem #1:
B-Tree inserts have unpredictable performance due to *high write amplification* in *random* pages.

Problem #2:
Operations on B-Trees are not atomic by default since all pages are *mutable*.

Problem #3:
Since a mutation can occur in any page at any point in time, B-Trees cannot be *efficiently compressed*.

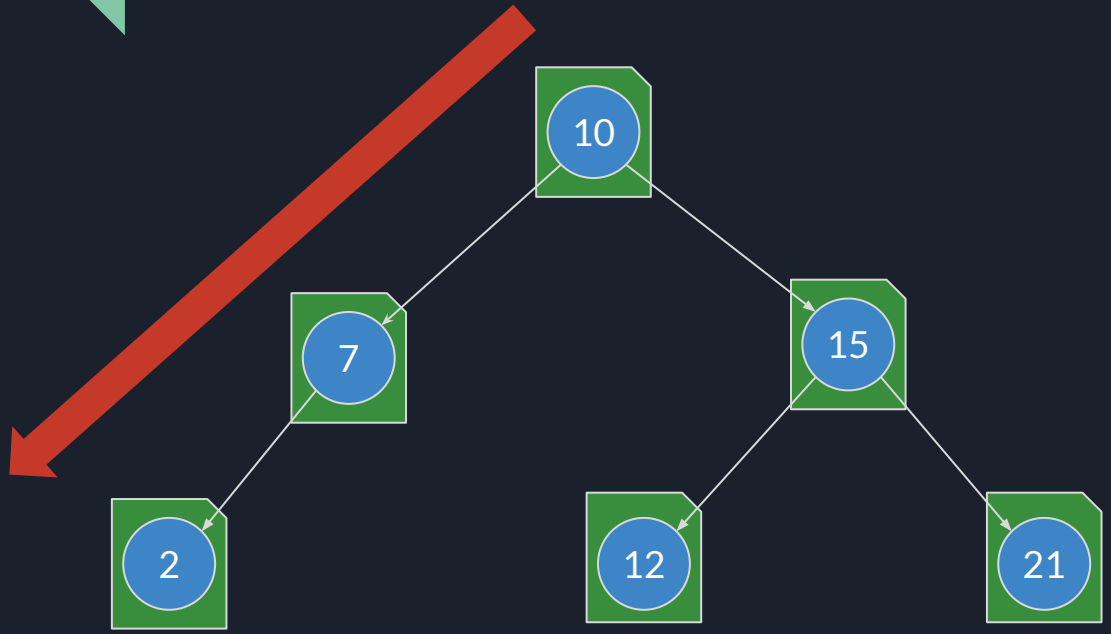
Binary Search Tree



Command:

Get all entries in the tree.

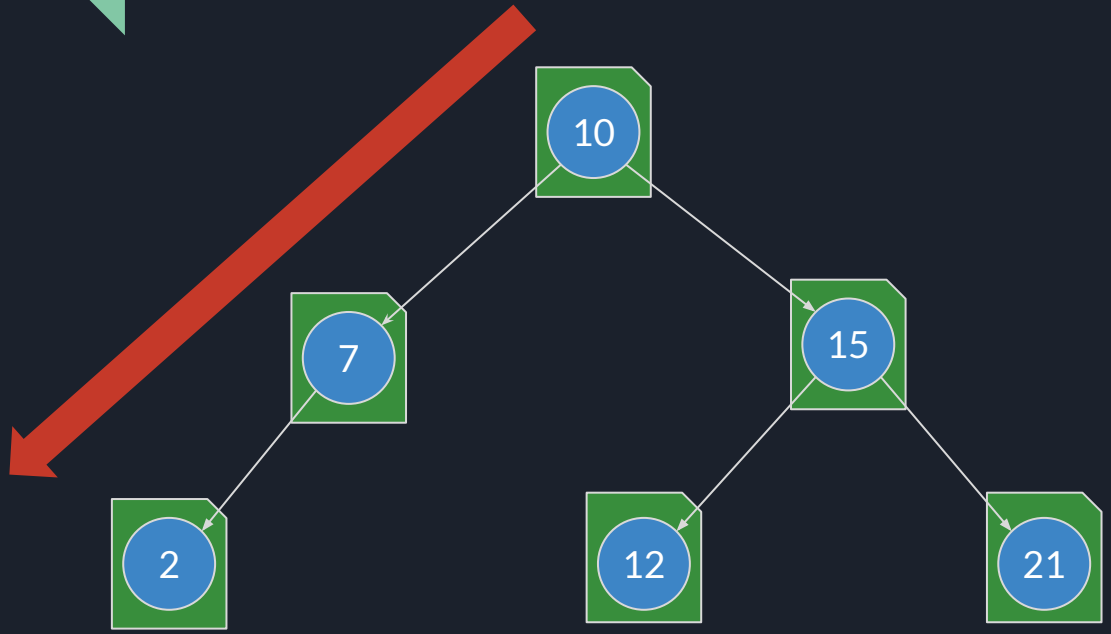
Binary Search Tree



Command:

Get all entries in the tree.

Binary Search Tree



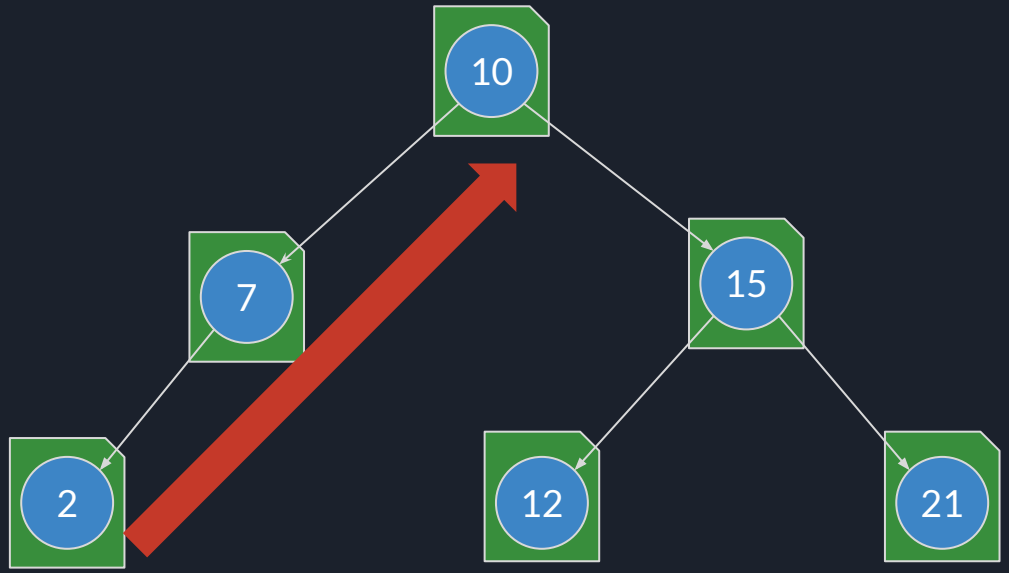
Command:

Get all entries in the tree.

Result:

2

Binary Search Tree



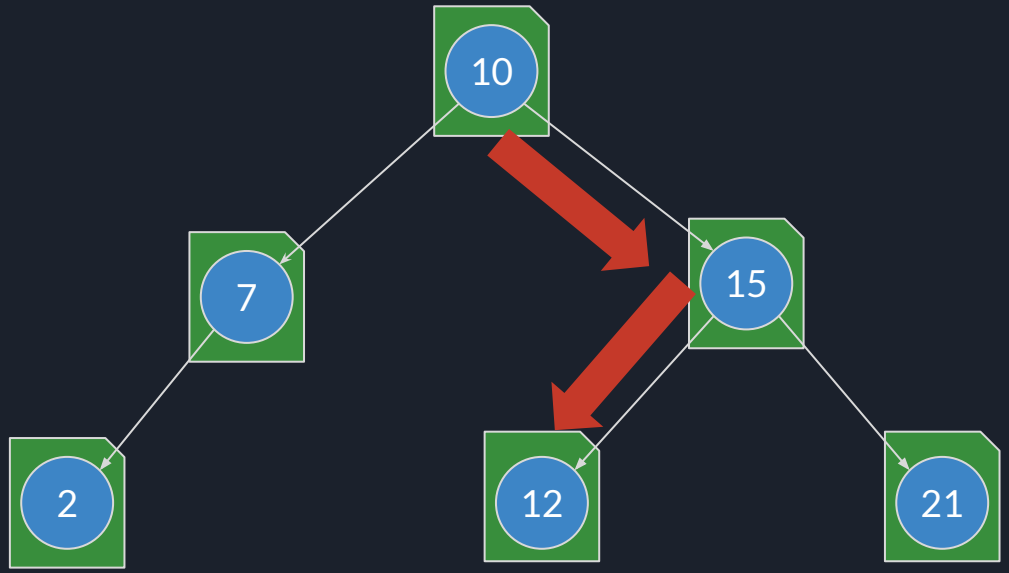
Command:

Get all entries in the tree.

Result:

2, 7, 10

Binary Search Tree



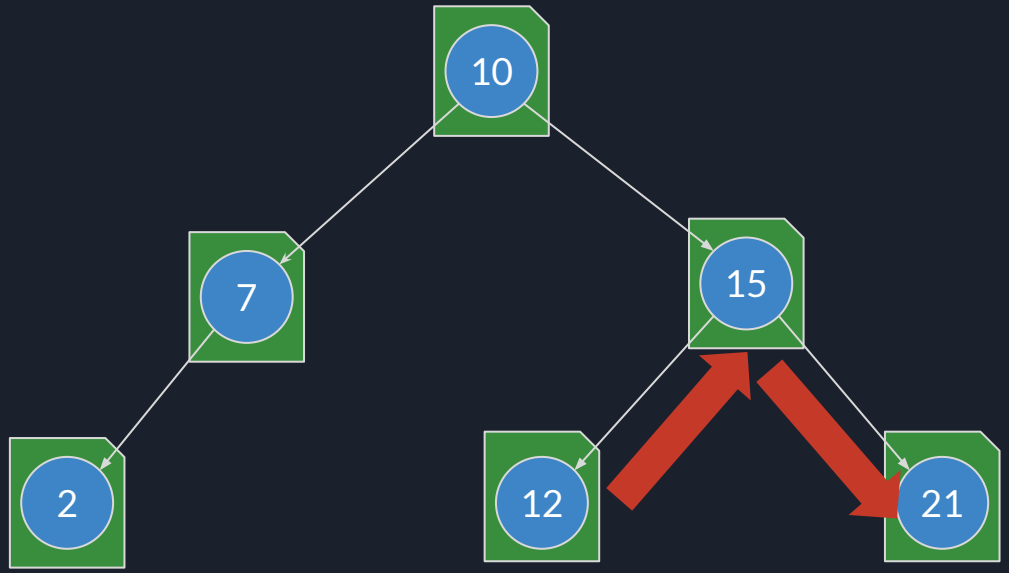
Command:

Get all entries in the tree.

Result:

2, 7, 10, 12

Binary Search Tree



Command:

Get all entries in the tree.

Result:

2, 7, 10, 12, 15, 21

Binary Search Tree on disk

Command:

Get all entries in the tree.

Result:

2, 7, 10, 12, 15, 21

Binary Search Tree on disk

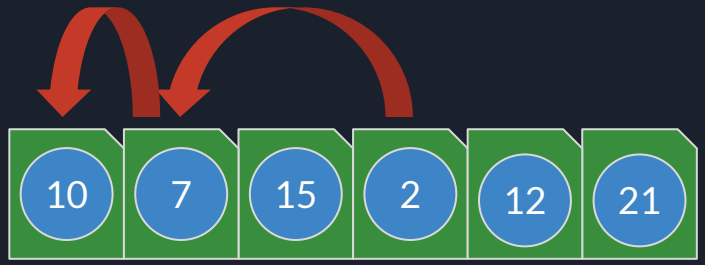
Command:

Get all entries in the tree.

Result:

2, 7, 10, 12, 15, 21

Binary Search Tree on disk



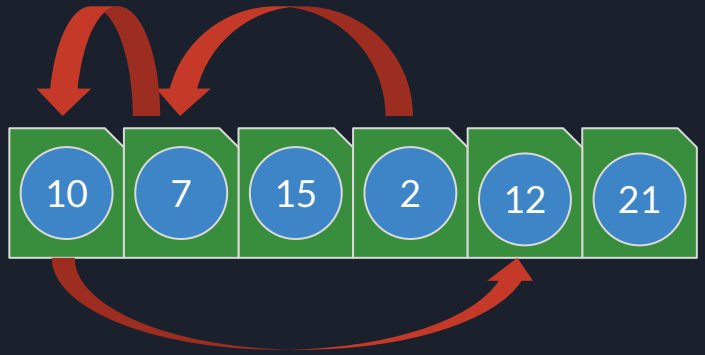
Command:

Get all entries in the tree.

Result:

2, 7, 10, 12, 15, 21

Binary Search Tree on disk



Command:

Get all entries in the tree.

Result:

2, 7, 10, 12, 15, 21

Binary Search Tree on disk

Command:

Get all entries in the tree.

Result:

2, 7, 10, 12, 15, 21

Binary Search Tree on disk

Command:

Get all entries in the tree.

Result:

2, 7, 10, 12, 15, 21


Hard drive go
BRRRRRRRRR
RRRRRRRRRR
RRRRRRR

Problem #4:
Reading a B-Tree sequentially *does not translate* to sequential disk access.

But what about SSDs?

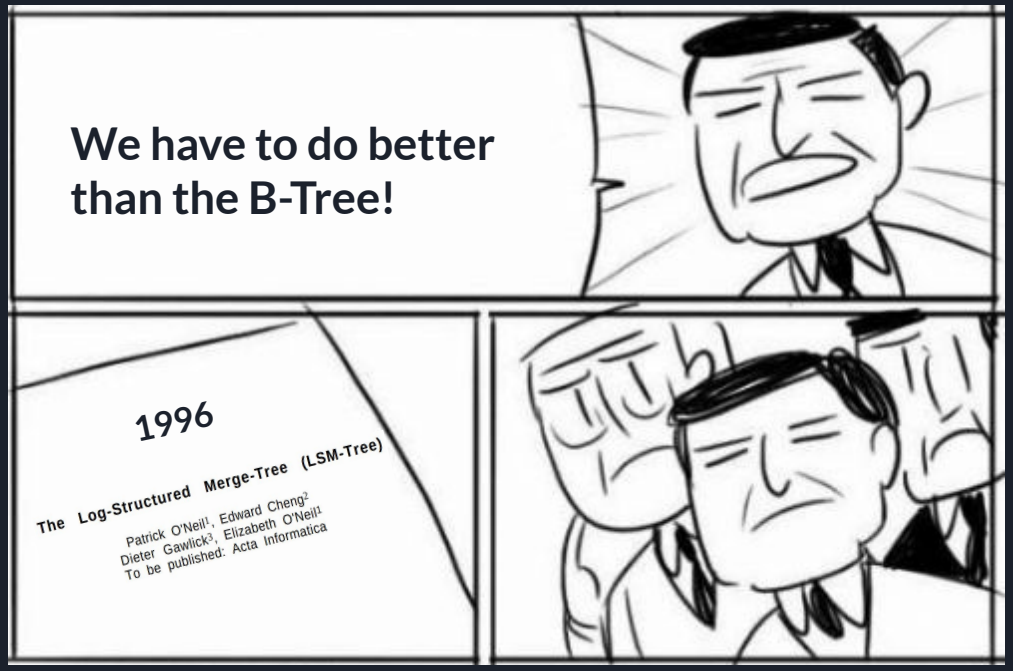
SSDs work differently...

- SSDs store data in segments
  - 1 segment is typically between 128KB and 512KB depending on the hardware
- When you access an address on disk, the SSD **always** loads the entire segment
  - When you load 1 page (4KB) the next 127 in sequence **come for free**...
  - ... **if (and only if)** you **keep reading sequentially**
- Overwriting the same segment over and over again is bad for the hardware
  - Modern SSDs can rewrite the same segment around 10.000 times before it “dies”
  - Wear leveling on hardware tries to mitigate this...
  - ... but still, continuously modifying the same page over and over is bad news for SSDs!

Problem #5:
B-Trees *do not align well* with the properties and features of *modern hardware*.

1996

We have to do better than the B-Tree!


We have to do better than the B-Tree!
1996
The Log-Structured Merge-Tree (LSM-Tree)
Patrick O'Neil[1], Edward Cheng[2]
Dieter Gawlick[3], Elizabeth O'Neil[1]
To be published: Acta Informatica


We have to do better than the B-Tree!
1996
The Log-Structured Merge-Tree (LSM-Tree)
Patrick O'Neil[1], Edward Cheng[2]
Dieter Gawlick[3], Elizabeth O'Neil[1]
To be published: Acta Informatica
2008
Bigtable: A Distributed Storage System for Structured Data
Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber
{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com
Google, Inc.

LevelDB (Google, Open Source)
2011

LevelDB: A Fast Persistent Key-Value Store

Wednesday, July 27, 2011

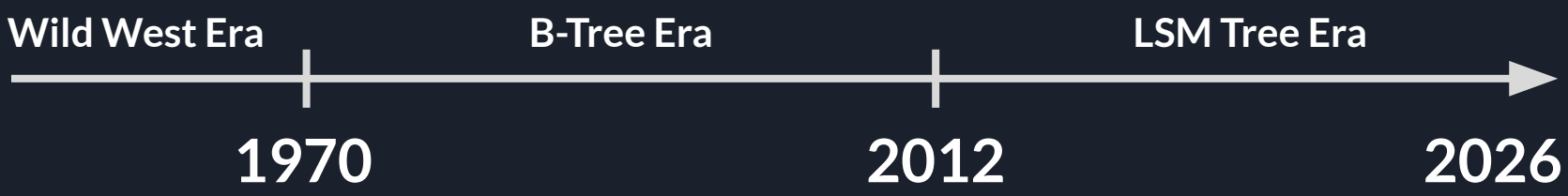
LevelDB is a fast key-value storage engine written at Google that provides an ordered mapping from string keys to string values. We are pleased to announce that we are open sourcing LevelDB under a BSD-style license.

RocksDB (Meta, Open Source)

2012

… and many more

Database Eras



So what *IS* an LSM Tree?

LSM Tree

LSM Tree

New Data
(Likely to change again soon)

Old Data
(Unlikely to change again soon)

LSM Tree

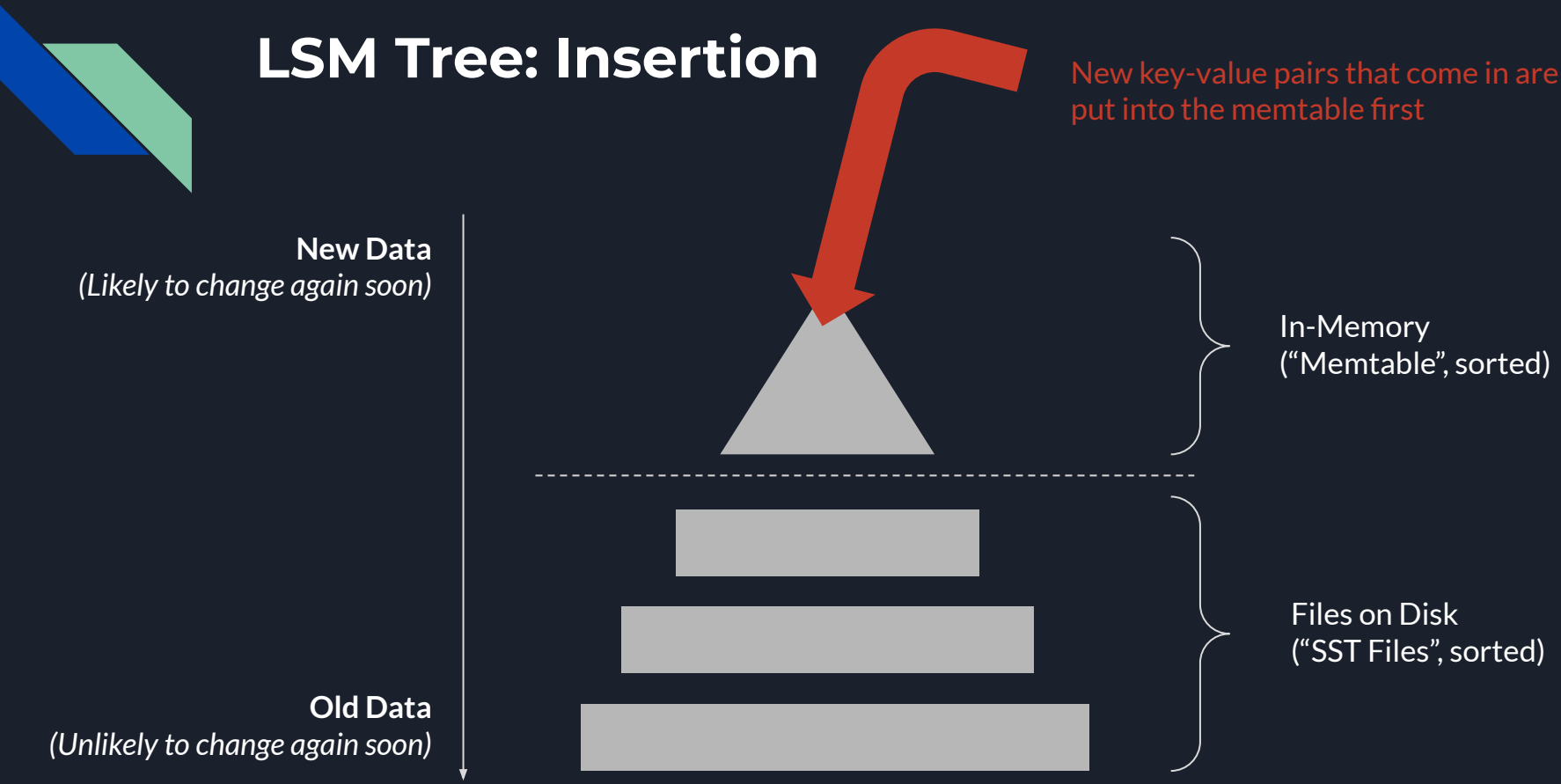
New Data
(Likely to change again soon)

In-Memory
("Memtable", sorted)

Files on Disk
("SST Files", sorted)

Old Data
(Unlikely to change again soon)

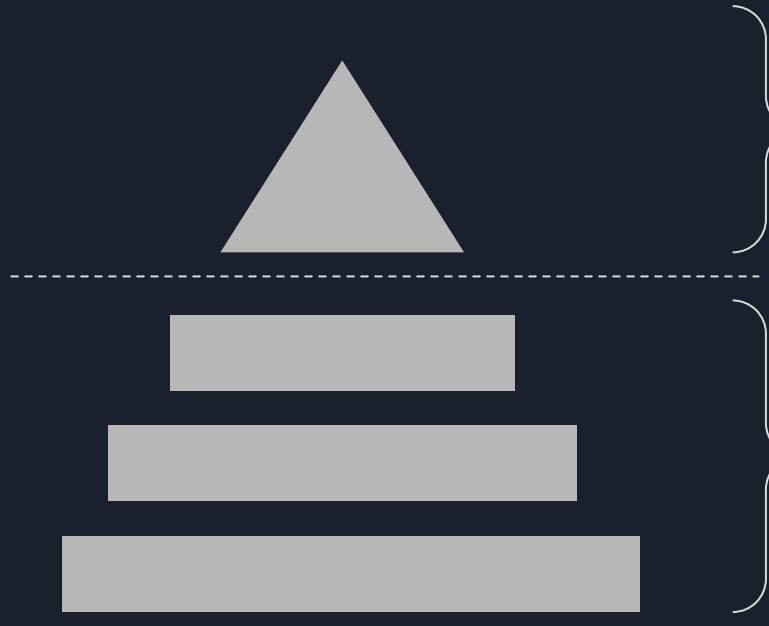
LSM Tree: Insertion



LSM Tree: Insertion

When the memtable is full…

New Data
(Likely to change again soon)



In-Memory
("Memtable", sorted)

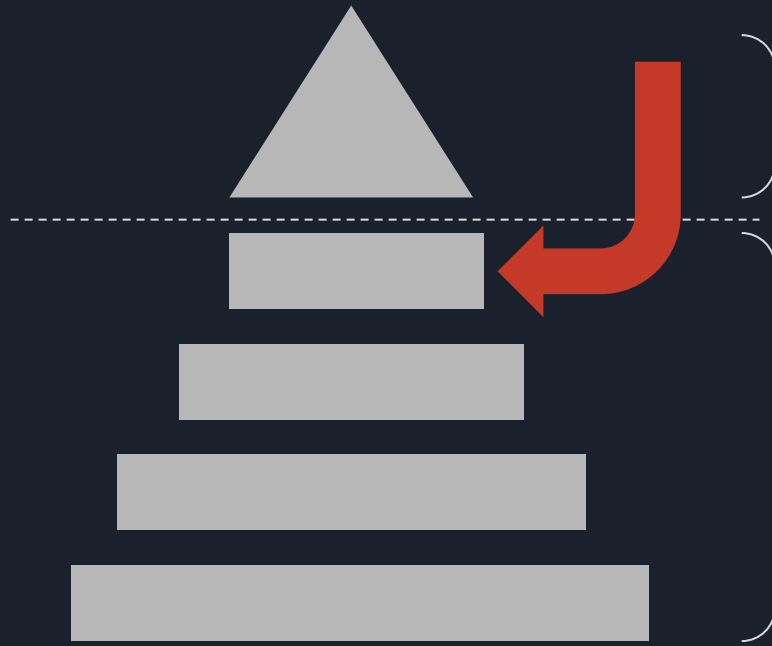
Files on Disk
("SST Files", sorted)

Old Data
(Unlikely to change again soon)

LSM Tree: Insertion

New Data
(Likely to change again soon)

Old Data
(Unlikely to change again soon)



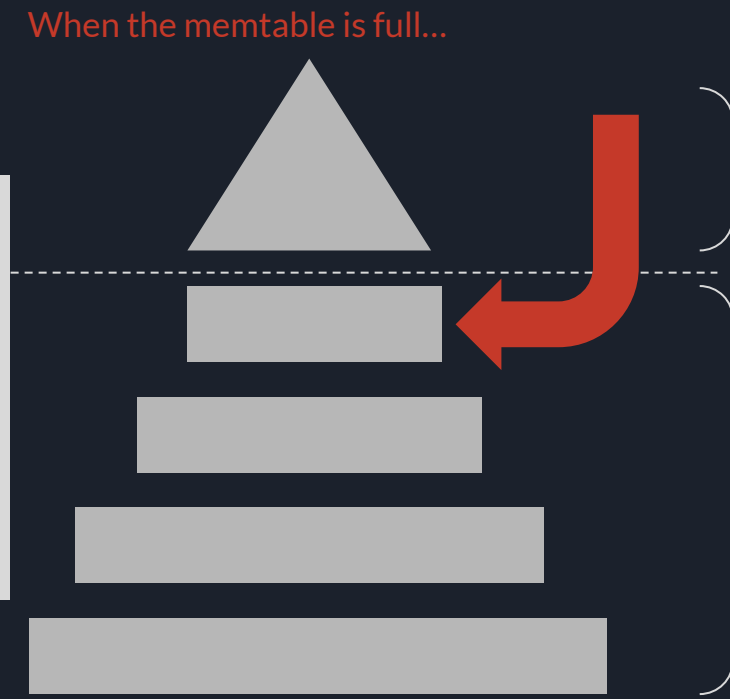
LSM Tree: Insertion

New Data
(Likely to change again soon)

This is awesome because:

- Data is written to disk **in larger chunks** and **sequentially**
- Append only: data is only written once and then **never modified again**
- Entries in each file are **sorted** and can be **efficiently compressed**

Old Data
(Unlikely to change again soon)



LSM Tree: Lookup

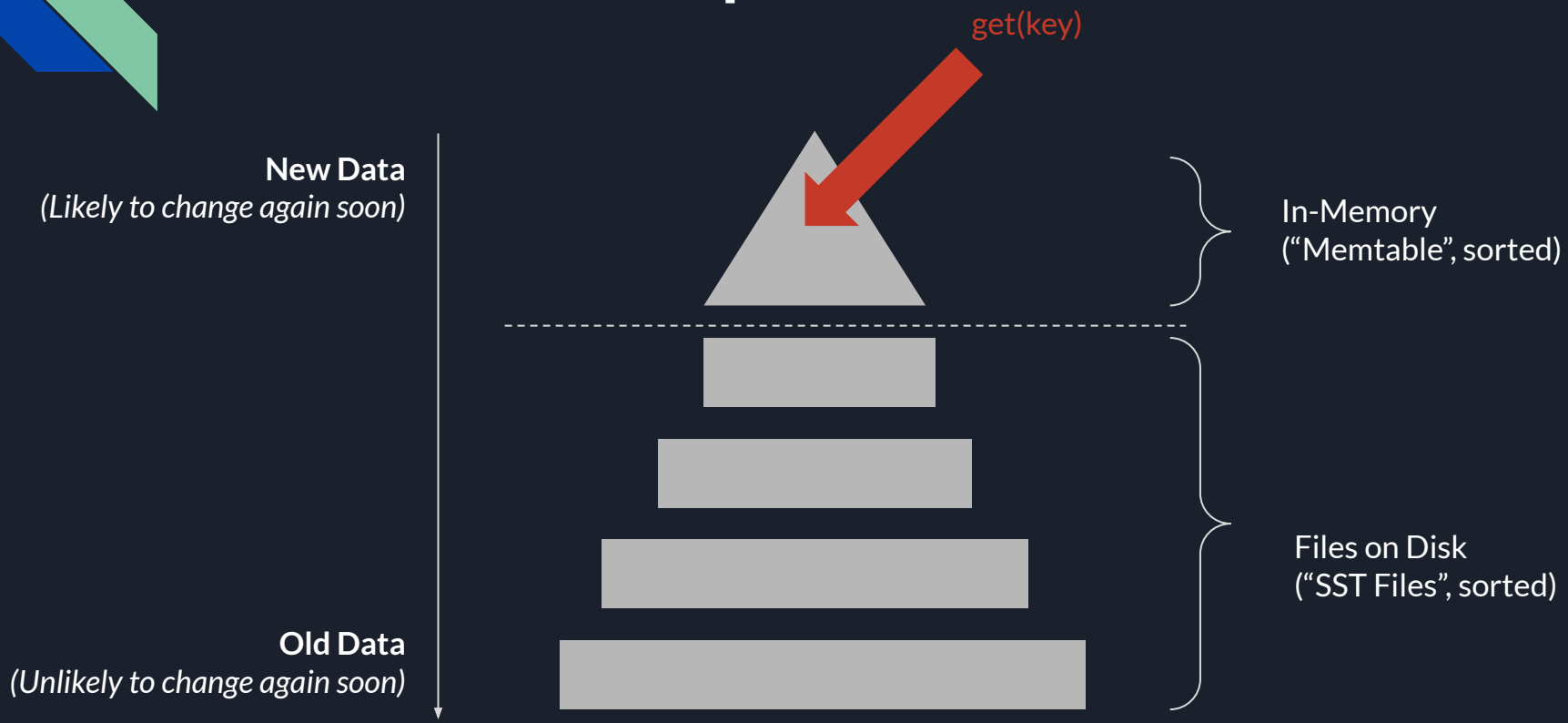
New Data
(Likely to change again soon)

In-Memory
("Memtable", sorted)

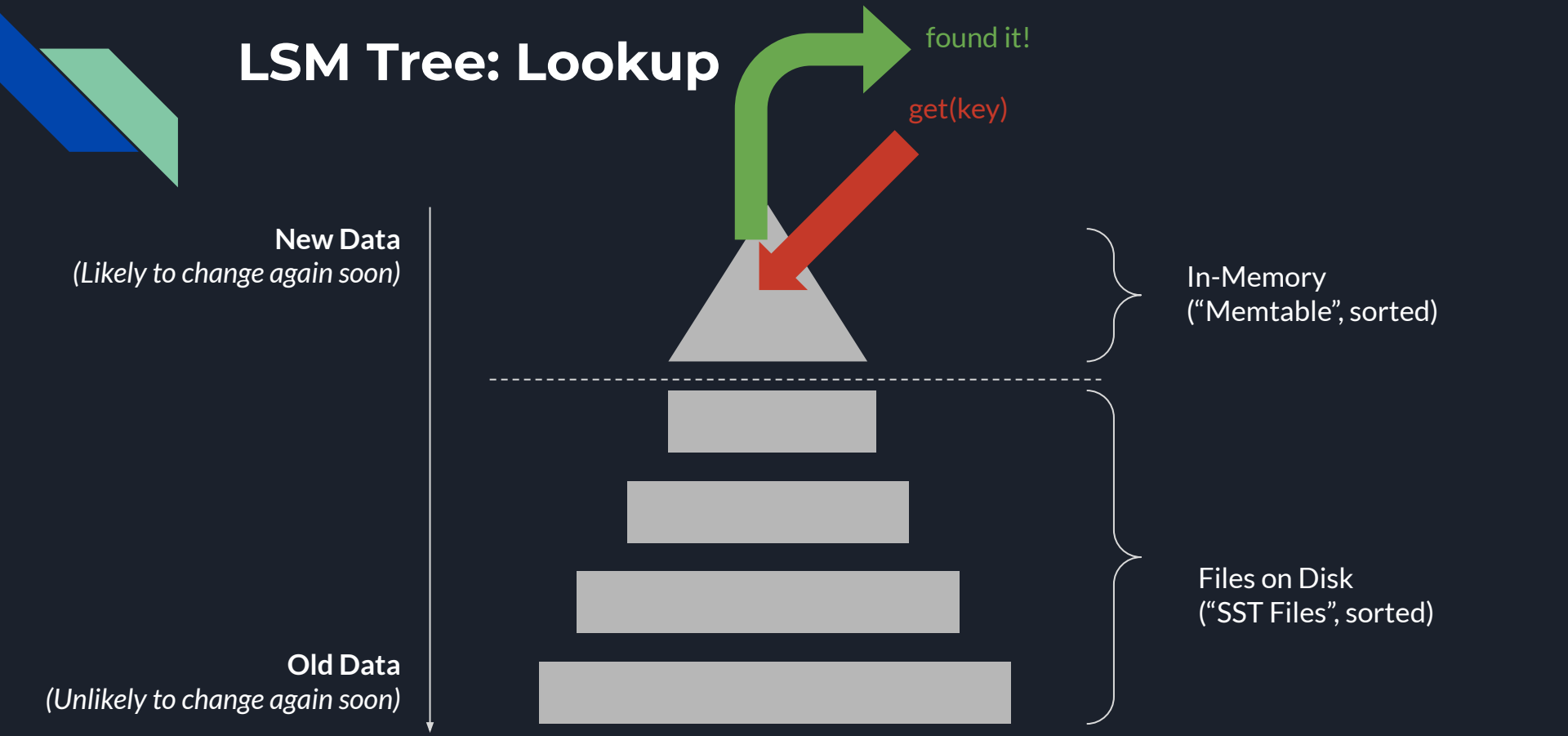
Files on Disk
("SST Files", sorted)

Old Data
(Unlikely to change again soon)

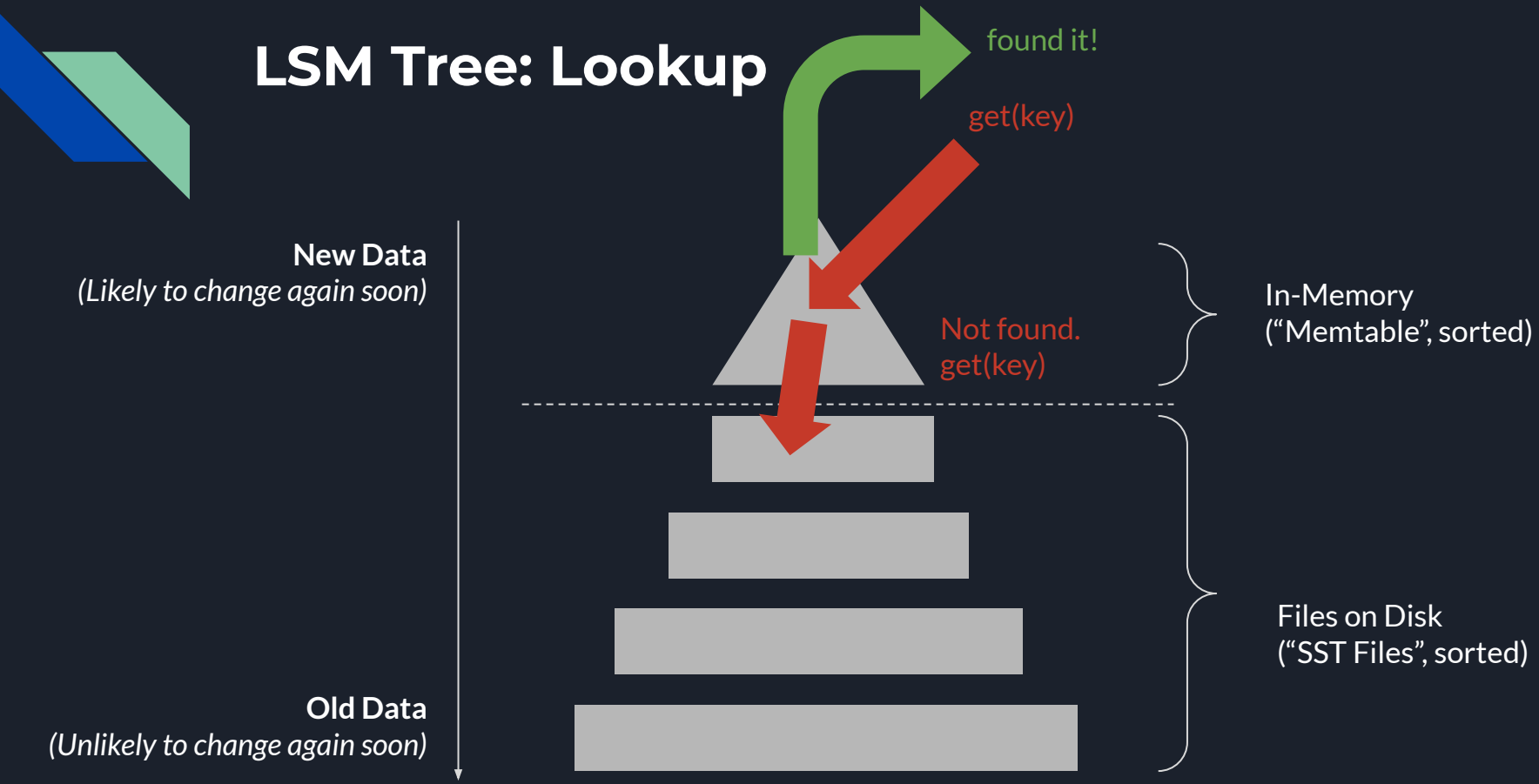
LSM Tree: Lookup



LSM Tree: Lookup



LSM Tree: Lookup



LSM Tree: Lookup

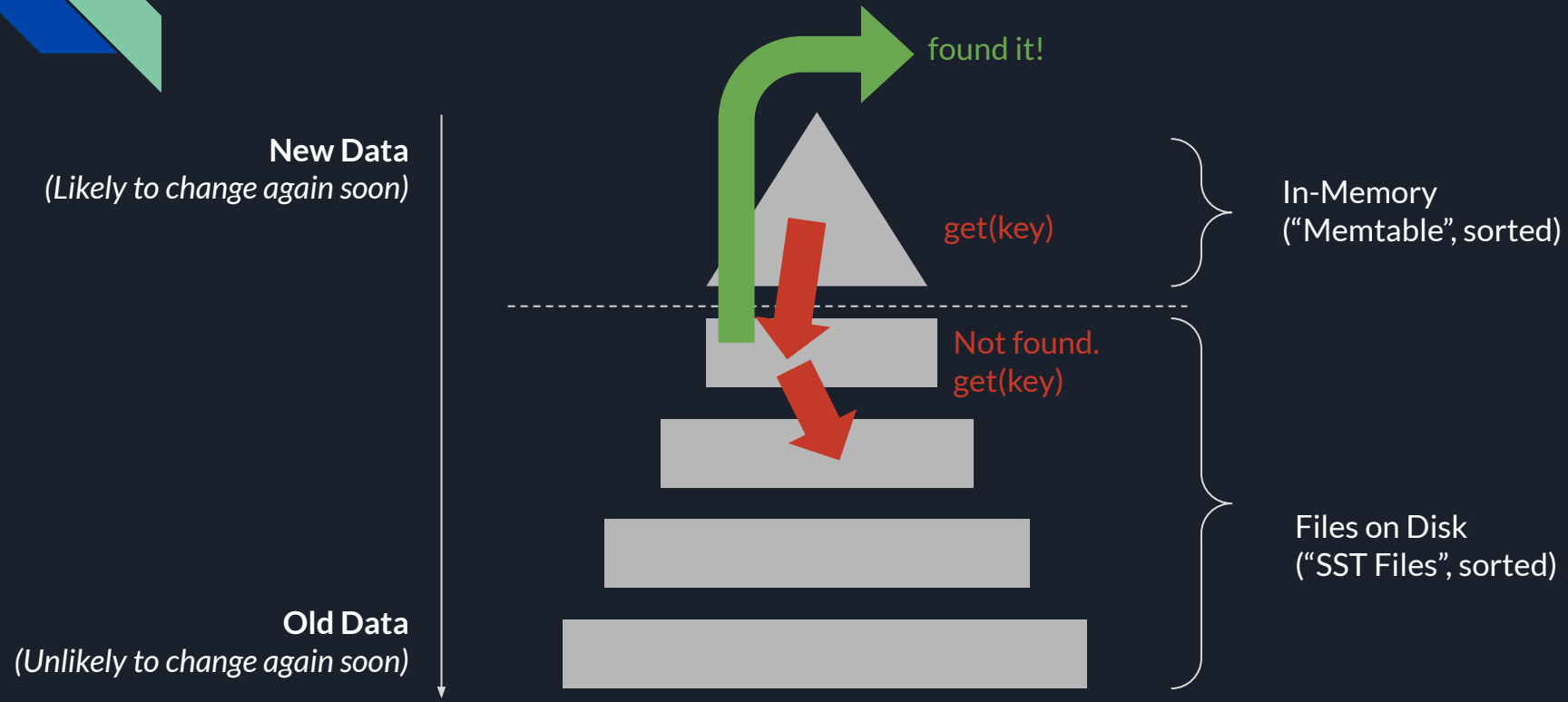
New Data
(Likely to change again soon)

In-Memory
("Memtable", sorted)

Files on Disk
("SST Files", sorted)

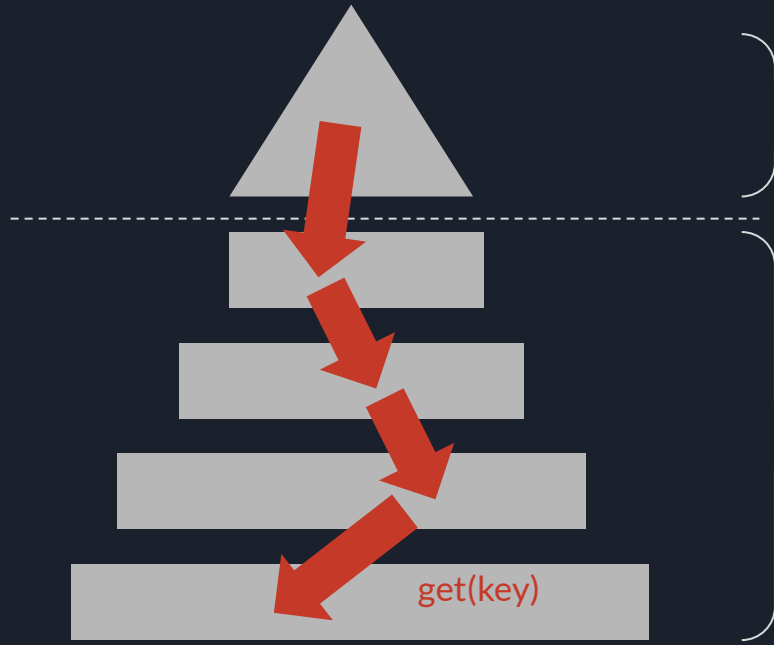
Old Data
(Unlikely to change again soon)

LSM Tree: Lookup



LSM Tree: Lookup

New Data
(Likely to change again soon)

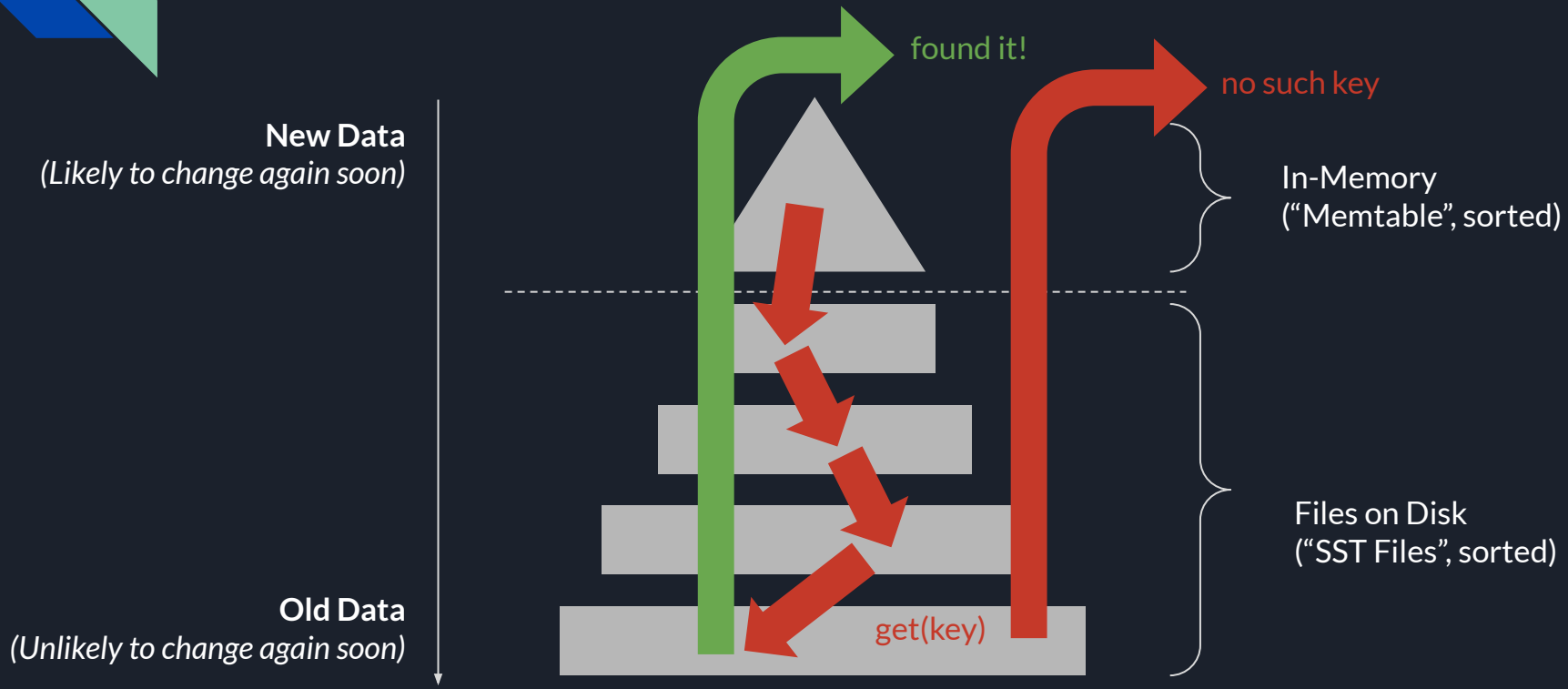


Old Data
(Unlikely to change again soon)

In-Memory
("Memtable", sorted)

Files on Disk
("SST Files", sorted)

LSM Tree: Lookup



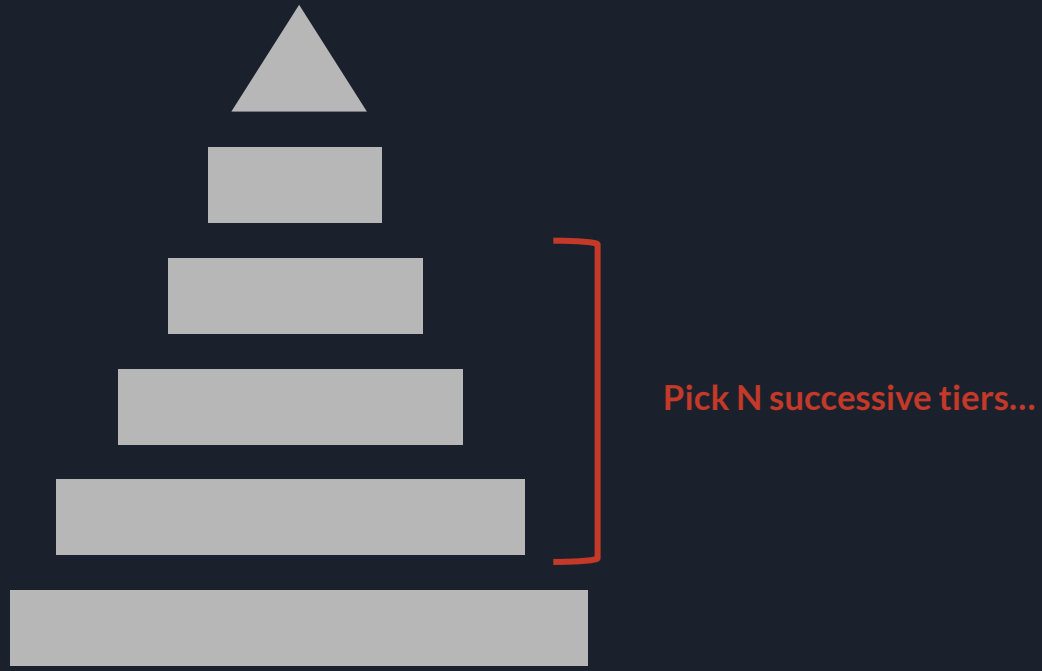
THAT’S QUITE BIG

High tree
-> Long search paths
-> Slow lookups
-> High disk footprint

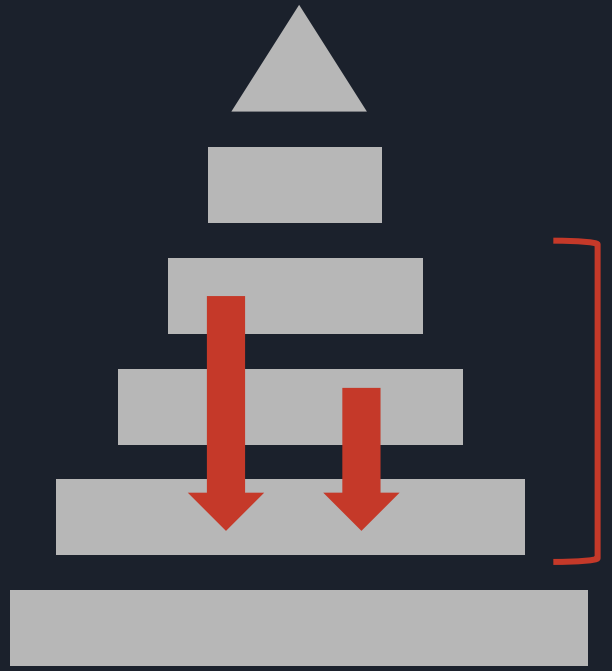
THAT'S QUITE BIG

Compaction

Compaction



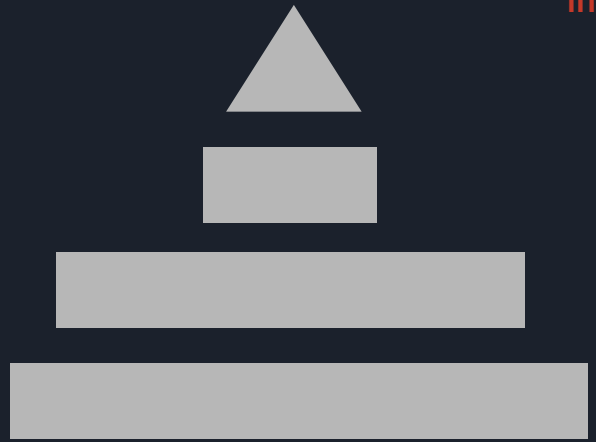
Compaction



Pick N successive tiers…

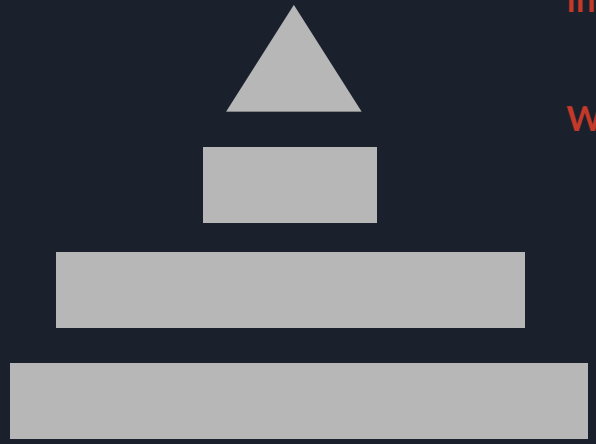
… and merge them together!

Compaction



This process can be performed in parallel in the background without interrupting writers or readers!

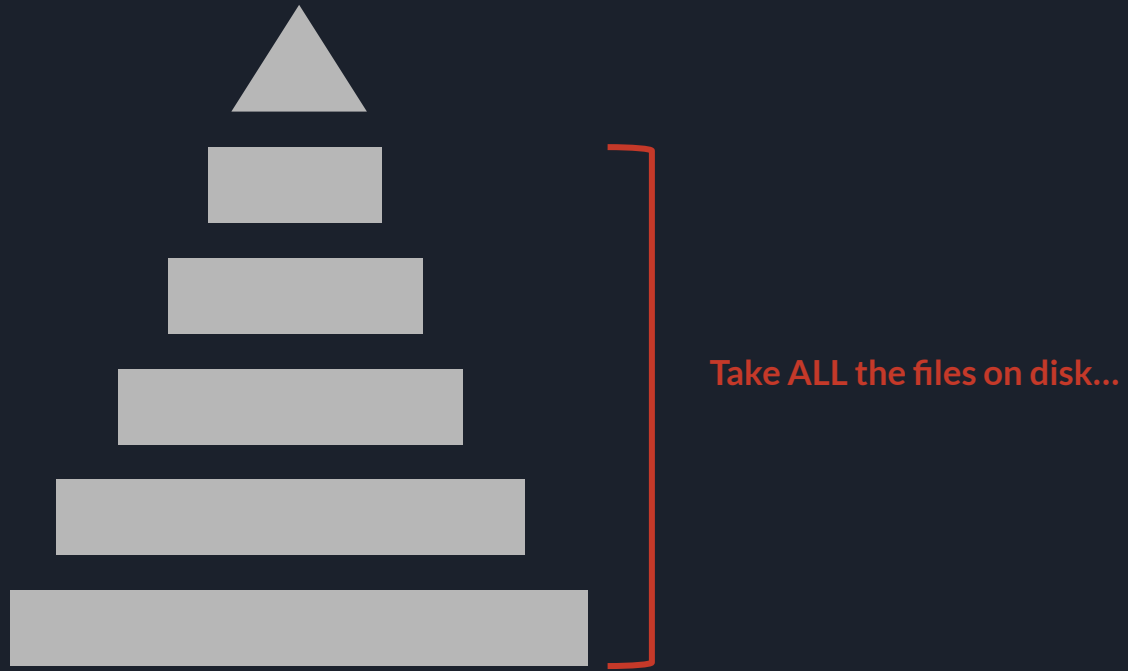
Compaction



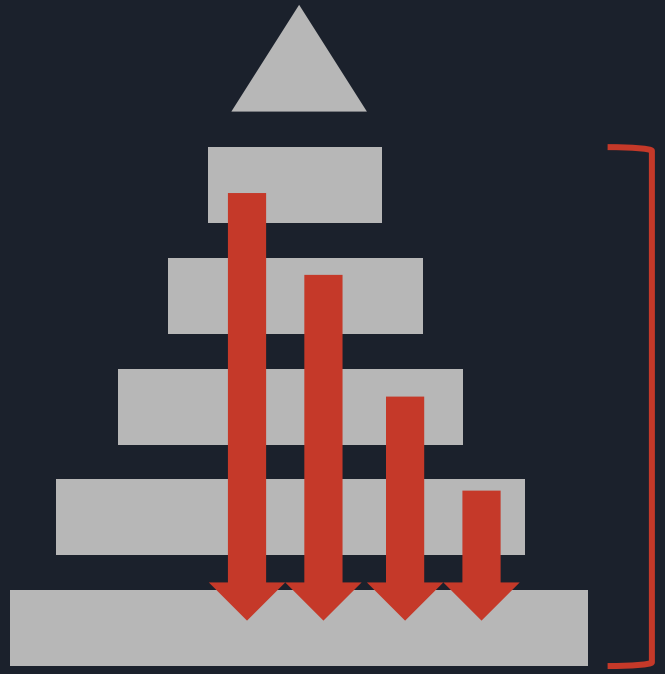
This process can be performed in parallel in the background without interrupting writers or readers!

We call this a *Minor* Compaction.

Major Compaction



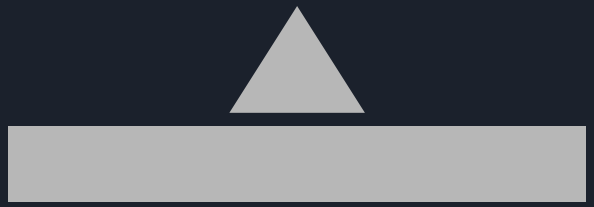
Major Compaction



Take ALL the files on disk…

… and merge them down!

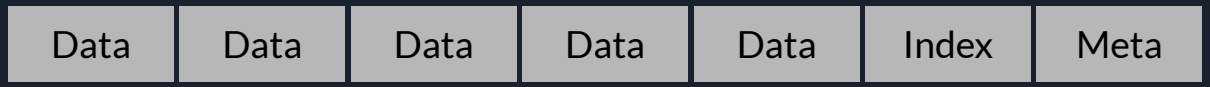
Major Compaction



String SorTed Files (SST)

- Fully sorted, self-indexed file
  - Rapid lookup
- Keys ordered sequentially on disk
  - Rapid iteration
  - Perfectly utilizes hardware burst loading
- Compressed
  - Very efficient disk utilization

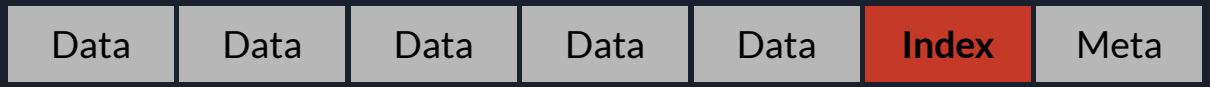
Anatomy of an SST File



Anatomy of an SST File

- Last block of an SST file
- Acts as a “header” and contains Meta-Information
  - largest key, smallest key, total count, …
  - compression algorithm of data blocks
  - file format version information
  - checksums
- Index block start & end offset
- Usually very small, < 8KB

Anatomy of an SST File



- Can be one or more blocks
- For **each** data block:
  - First key
  - Last key
  - Start offset
  - End offset

Anatomy of an SST File

- Each data block is compressed (snappy, lz4, gzip, zstd, …)
- Data blocks are big compared to traditional pages (~8MB)
- Contain key-value pairs sorted by key
- May contain a “Block Index” for faster search
    - Selection of keys (e.g. every 1000th key) and their offsets

There’s so much more but…

There’s so much more but...

It’s time to stop

Summary

- **Every** database is a (sophisticated) Key-Value Store
  - Tables, Graphs, Documents, …

Summary

- **Every** database is a (sophisticated) Key-Value Store
  - Tables, Graphs, Documents, …
- When choosing a database, the access language / format should not be your primary concern
  - What are your Access Patterns and does the database support them efficiently?
  - How much data do you expect and does the database scale to that level?
  - How important are transactions in your application and does your database support them?

Summary

- **Every** database is a (sophisticated) Key-Value Store
  - Tables, Graphs, Documents, …
- When choosing a database, the access language / format should not be your primary concern
  - What are your Access Patterns and does the database support them efficiently?
  - How much data do you expect and does the database scale to that level?
  - How important are transactions in your application and does your database support them?
- LSM Trees as the next step of evolution in Storage Engines

Where to Learn More?

Where to Learn More?

https://github.com/MartinHaeusler/lsm4k

My own LSM Tree Implementation *(100% hand-written!)*

Where to Learn More?

https://github.com/MartinHaeusler/lsm4k

My own LSM Tree Implementation *(100% hand-written!)*

… ask the Engineering Kiosk team to invite me again for Part 2!

- Why File Systems are fast, unreliable and sometimes evil
- Write Ahead Logs
- Recovery
- Transactions
- Multi-Version Concurrency Control
- … and more!

THANK YOU!

https://github.com/MartinHaeusler/lsm4k
My own LSM Tree Implementation *(100% hand-written!)*

martin.haeusler89@gmail.com

Backup Slides

Suffix Encoding: Combining key & value

- To extract key length:
  `data.slice(totalLength - sizeof(int), totalLength)`
- To extract key: `data.`
  `data.slice(0, keyLength)`
- To extract value:
  `data.slice(keyLength, totalLength - sizeof(int))`

This is awesome because:

- Minimal overhead (32 bit)
- No restriction on keys or values
  - No “forbidden” bytes
  - No “magic” bytes
  - No “separator” bytes
- Key stays in front

WiscKey

Lanyue Lu, 2016

WiscKey: Separating Keys from Values in SSD-conscious Storage

Lanyue Lu, Thanumalayan Sankaranarayana Pillai, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, *University of Wisconsin—Madison*

https://www.usenix.org/conference/fast16/technical-sessions/presentation/lu