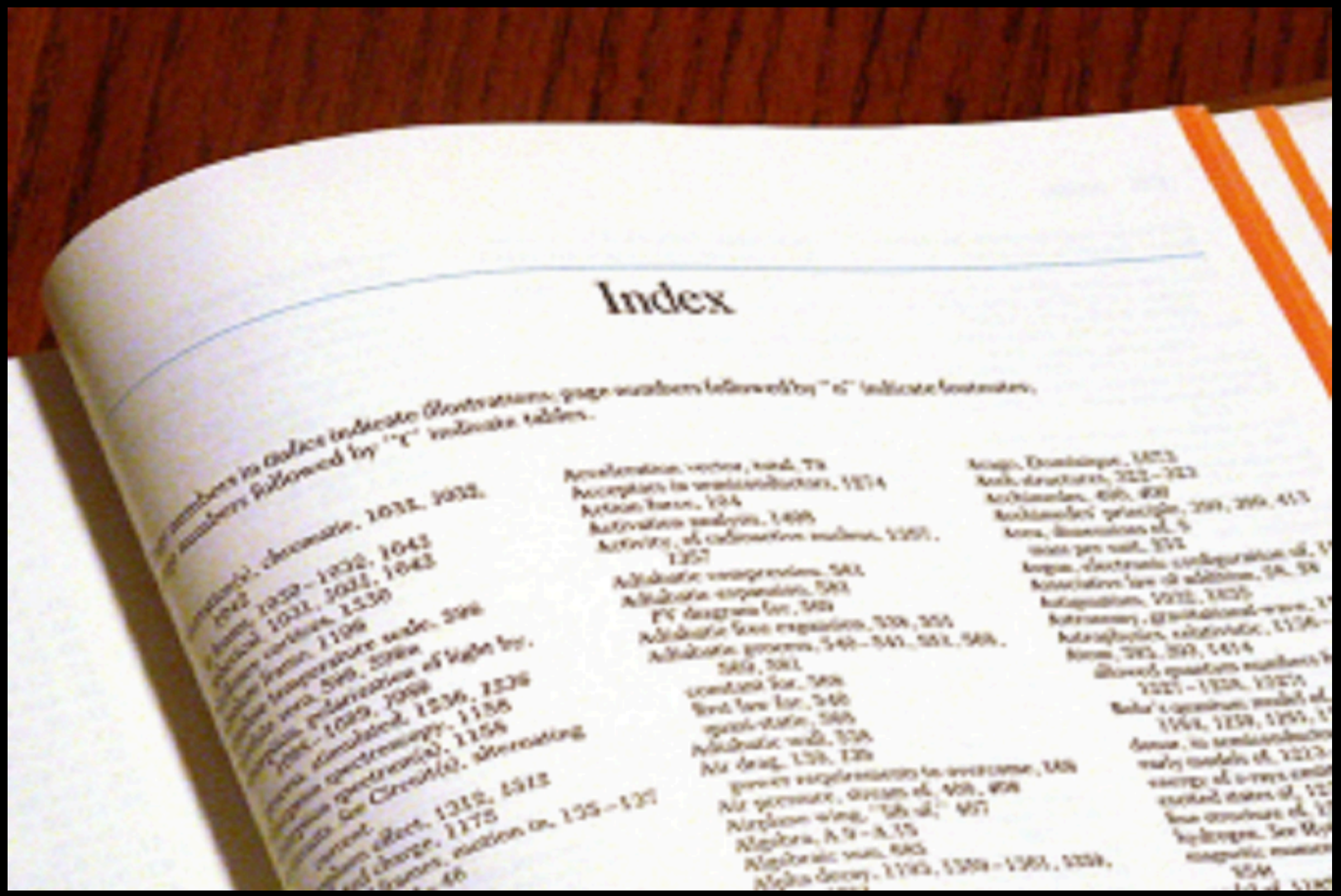
通过索引加速查询

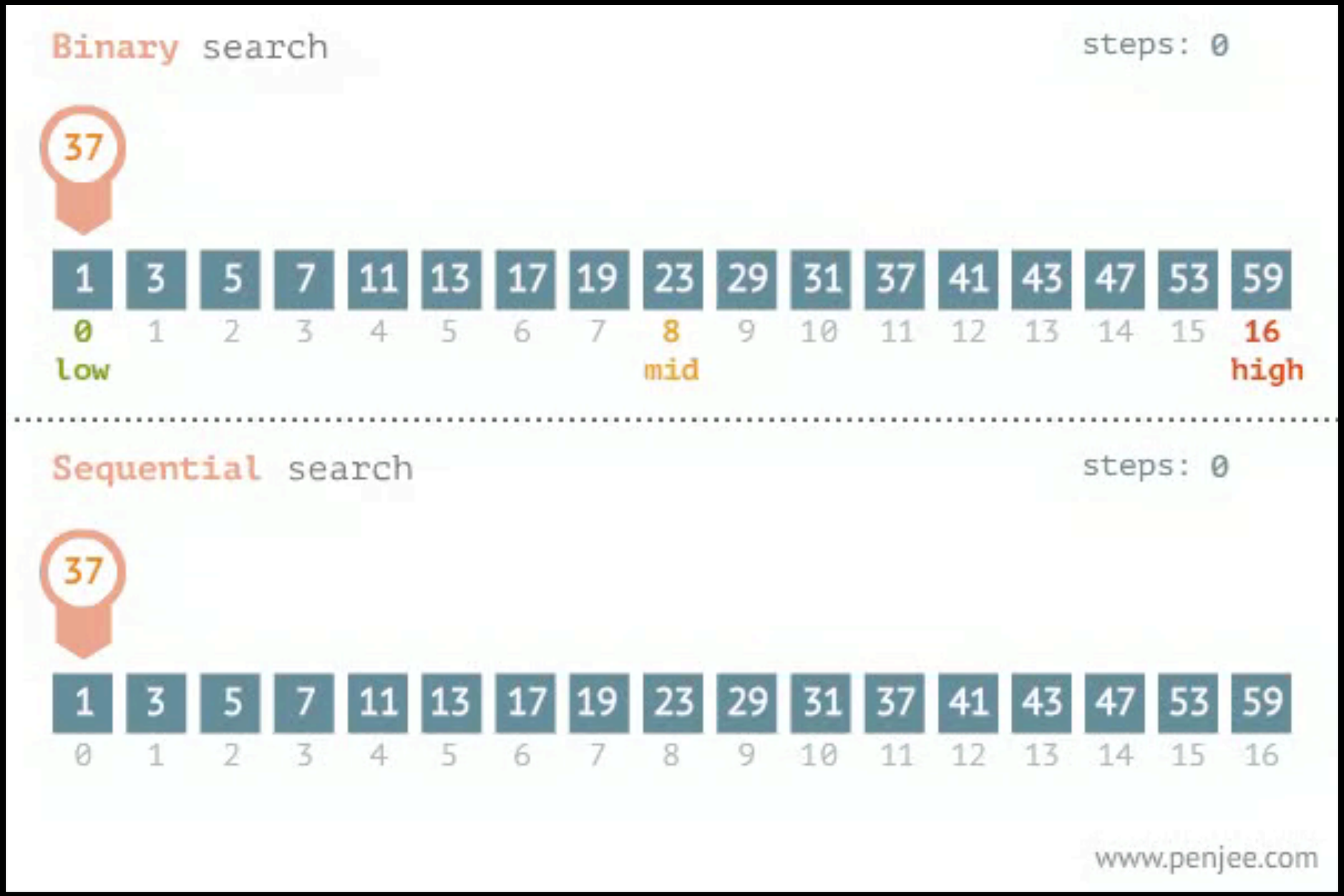
关于我

- 王子亭
- Node.js 开发者
- LeanCloud
- https://jysperm.me
- GitHub: jysperm

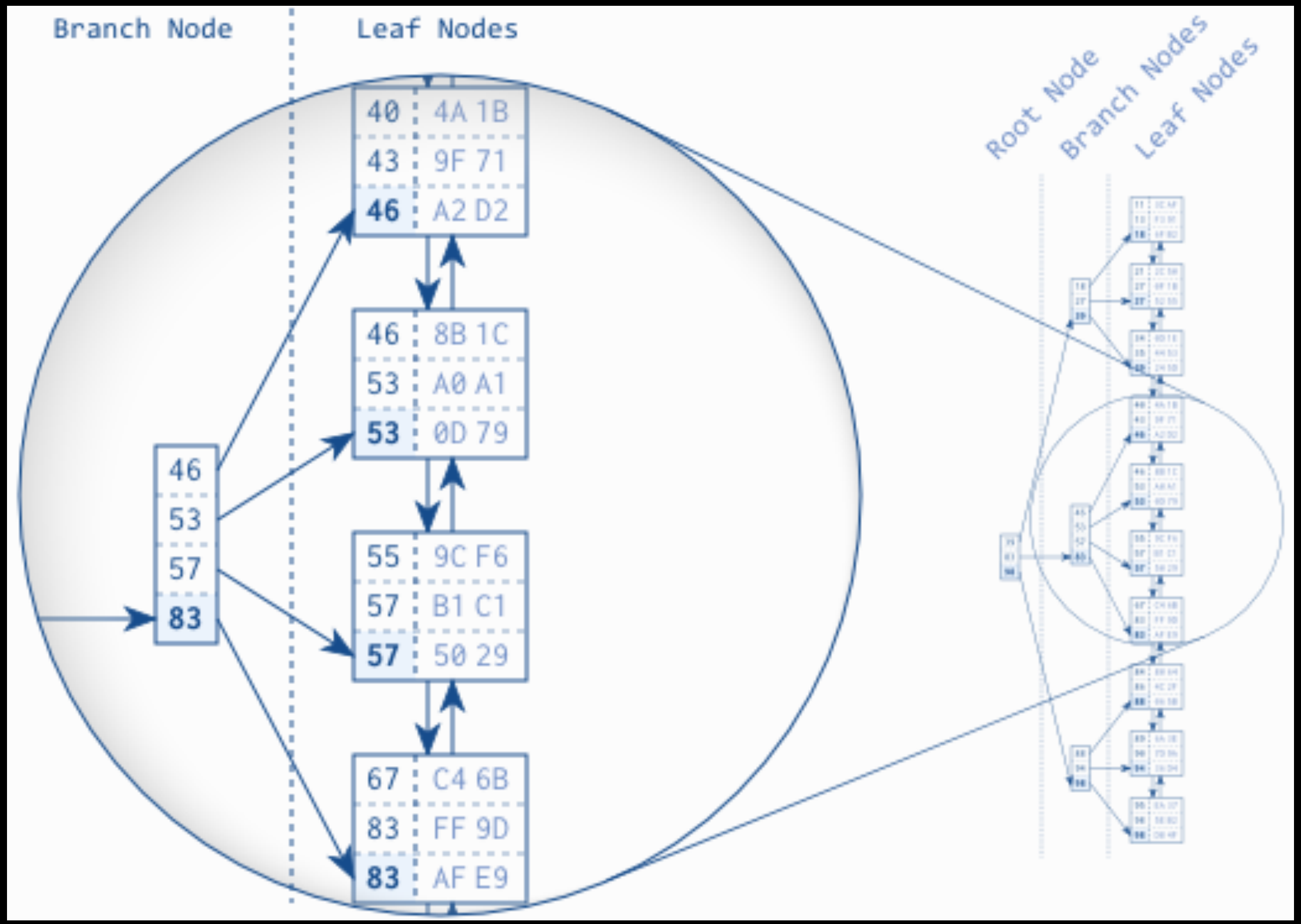
Index

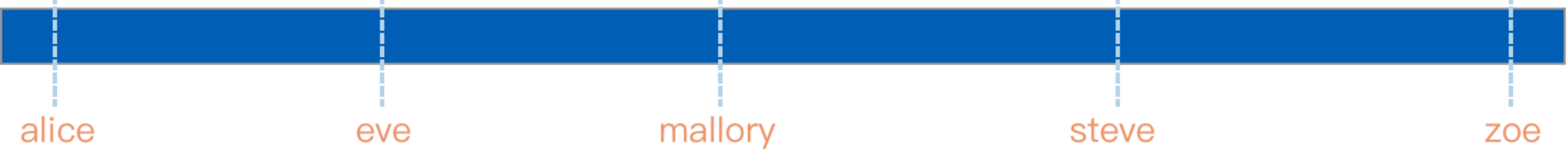


二分查找

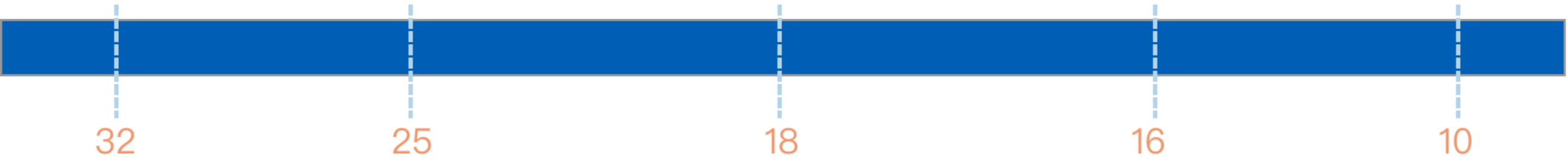

Binary search
steps: 0
37
1 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
low
mid
high
Sequential search
steps: 0
37
1 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
www.penjee.com

B-Tree


Branch Node
Leaf Nodes
40 4A 1B
43 9F 71
46 A2 D2
46 8B 1C
53 A0 A1
53 0D 79
46
53
57
83
55 9C F6
57 B1 C1
57 50 29
67 C4 6B
83 FF 9D
83 AF E9
Root Node
Branch Nodes
Leaf Nodes



{name: 1}



{age: -1}

```
select * from _User where name = 'steve'
select * from _User where name like 'ma%'

select * from _User where age > 18
select * from _User where age > 18 and age < 22
select * from _User order by age
select * from _User order by age desc
```

单列索引

- √ 全匹配查询
- √ 包含查询（数组）
- √ 前缀模糊查询
- √ 大于、小于等范围查询
- √ 正、反排序
- X 后缀模糊查询

「区分度」

```
{name: 'alice', age: 20, gender: 'female'}
{name: 'bob', age: 20, gender: 'male'}
{name: 'eve', age: 18, gender: 'female'}
{name: 'mallory', age: 29, gender: 'female'}
{name: 'steve', age: 28, gender: 'male'}
{name: 'justin', age: 28, gender: 'male'}
```

```
select * from _User where name = ? and gender = ?
```

复合索引

```
select * from _User where city = 'beijing' and age = 18
select * from _User where city = 'beijing' order by age
```

{city: 1, age: 1}

适用于

```
{city: 1, age: 1}
  beijing-17
  beijing-20
  beijing-21
  beijing-25
  chongqing-18
  chongqing-20
  shanghai-16
  shanghai-25
  qingdao-22
```

- √ 在多个字段上进行匹配查询

```
select ... where city = 'beijing' and age = 18
```

- √ 匹配的同时进行范围查询或排序

```
select ... where city = 'beijing' and age < 16
select ... where city = 'beijing' order by age
```

- √ 可以不使用右侧的字段

```
select ... where city = 'beijing'
select ... where order by city
```

- √ 排序或反向排序

```
select ... order by city, age
select ... order by city desc, age desc
```

不适用于

```
{city: 1, age: 1}
  beijing-17
  beijing-20
  beijing-21
  beijing-25
  chongqing-18
  chongqing-20
  shanghai-16
  shanghai-25
  qingdao-22
```

- X 直接使用右侧的字段

```
select ... where age > 18
```

- X 范围查询和排序不在最右侧
- X 多个范围查询

```
select ... where city like 'b%' and age = 18
select ... where city like 'b%' and age > 18
select ... where age = 18 order by city
```

- X 排序方向与索引不一致

```
select ... order by city asc, age desc
```

复合索引

- √ 在多个字段上进行匹配查询
- √ 匹配的同时进行范围查询或排序
- √ 将最左字段当作单列索引使用
- √ 排序或反向排序

- X 直接使用右侧的字段
- X 范围查询和排序不在最右侧
- X 多个范围查询
- X 排序方向和索引不一致

复合索引

—— 顺序十分重要

- 如果有范围查询和排序字段的话，放在最右侧
- 将区分度越高的字段放在越左侧

LeanCloud

xxxxxxxxxxxxxxxxxxx._Installation 26289290

Query	Count	Slow	Avg Cost	Advice
updatedAt>,valid,deviceType,channels	325325	18969	103	**{ 'channels':1, 'valid':1, 'deviceType':1, 'updatedAt':1 }**
updatedAt>,valid,deviceType,buildVersion,channels	2	0	3	**{ 'buildVersion':1, 'channels':1, 'valid':1, 'updatedAt':1 }**

Exist	Advisor
{ 'deviceUDID':1 } Drop { 'createdAt':1, 'valid':1 } { 'installationId':1 } { 'deviceType':1 } { '_id':1 } { 'createdAt':1 } { 'updatedAt':1 } { 'deviceToken':1 } { 'channels':1 }	{ 'buildVersion':1, 'channels':1, 'valid':1, 'updatedAt':1 } Create { 'channels':1, 'valid':1, 'deviceType':1, 'updatedAt':1 } Create

属性

- 主键索引（objectId）
- 唯一（Unique）索引
- 稀疏（Sparse）索引

常见慢查询

- 不等于和不包含查询

```
select * from _User where name ≠ 'alice'
select * from _User where city not in ('beijing', 'shanghai')
```

- 通配符在前面的模糊查询

```
select * from _User where city like '%jing'
```

- 无索引的 count、查询和排序（复合索引顺序不匹配）
- 多个范围查询
- skip 跳过较多的行数

```
select * from _User limit 10000, 10 -- wrong
select * from _User limit 10 where createdAt < '2017-02-13T10:52:07.490Z' -- correct
```

性能优化

1.优化索引（整理查询条件，用最少的索引来覆盖）

2.添加缓存（用 Redis 缓存热点数据）

3.优化查询（添加更多限制条件，使用高区分度字段）

4.优化数据结构（适当冗余）

Q & A

- https://en.wikipedia.org/wiki/B-tree
- https://docs.mongodb.com/manual/indexes
- <高性能 MySQL>

LeanCloud

https://leancloud.cn/jobs/