

【VIP直播课】

深入剖析MySQL索引原理

青山





青山-咕泡盆鱼宴

曾就职于国内顶级FinTech公司

多年Java从业经验，常年从事金融领域的项目研发，拥有传统金融和大型互联网金融项目的架构设计经验。经历了消费金融公司从单体架构到大型分布式、微服务系统的演变，支撑千万级客户和日均过亿放款量。

- 1、理解索引的本质
- 2、通过推演掌握索引底层的数据结构
- 3、掌握在不同存储引擎中索引的落地方式
- 4、掌握索引的创建和使用原则



适合了解MySQL架构，知道存储引擎的概念，了解索引基本类型（普通索引、唯一索引、主键索引），了解在MySQL中创建索引基本语法的同学



- 表的索引越全越好，因为不管什么情况都能用到索引，对吗？
- 为什么不要在性别字段上建索引？
- 为什么不建议使用身份证作为主键？
- 模糊匹配 like abc%，like %2673%，like %888都用不到索引，对吗？
- 不要使用select *，写明具体查询字段，为什么？



1

索引到底是什么？



数据库索引，是数据库管理系统（DBMS）中一个排序的**数据结构**，以协助快速查询、更新数据库表中数据。



索引类型：Normal、Unique、Fulltext



索引，应该选择一种什么数据结构？



2

MySQL索引数据模型推演

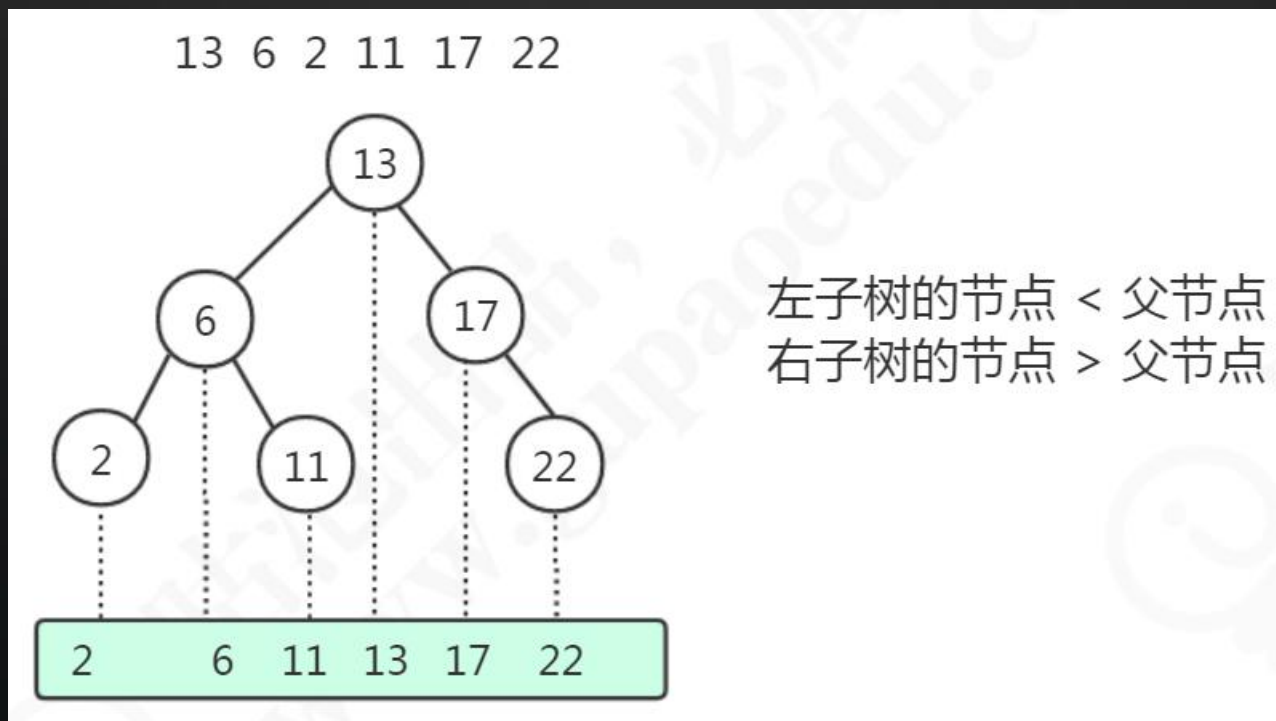


猜猜我双十一买了多少钱？

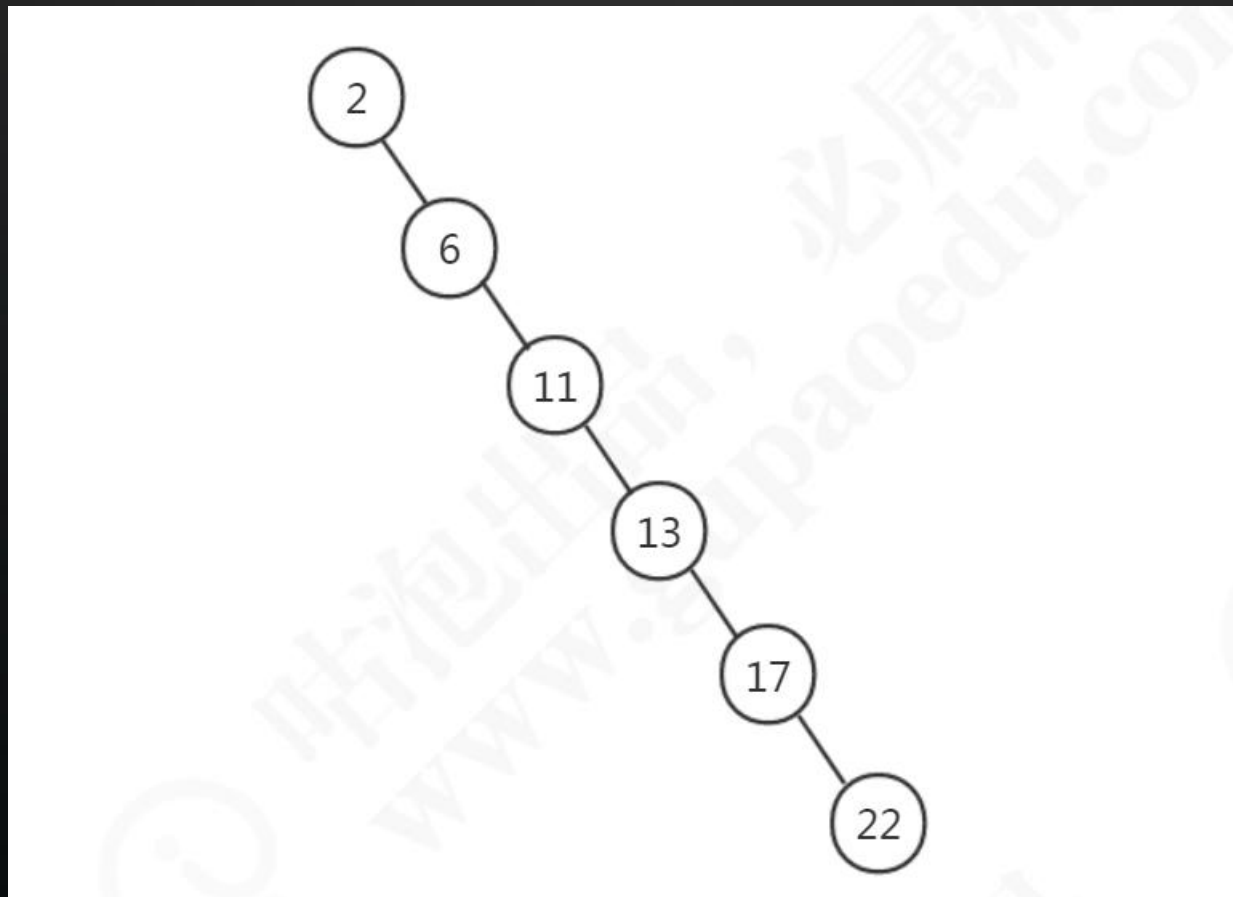


有序数组：
单链表：



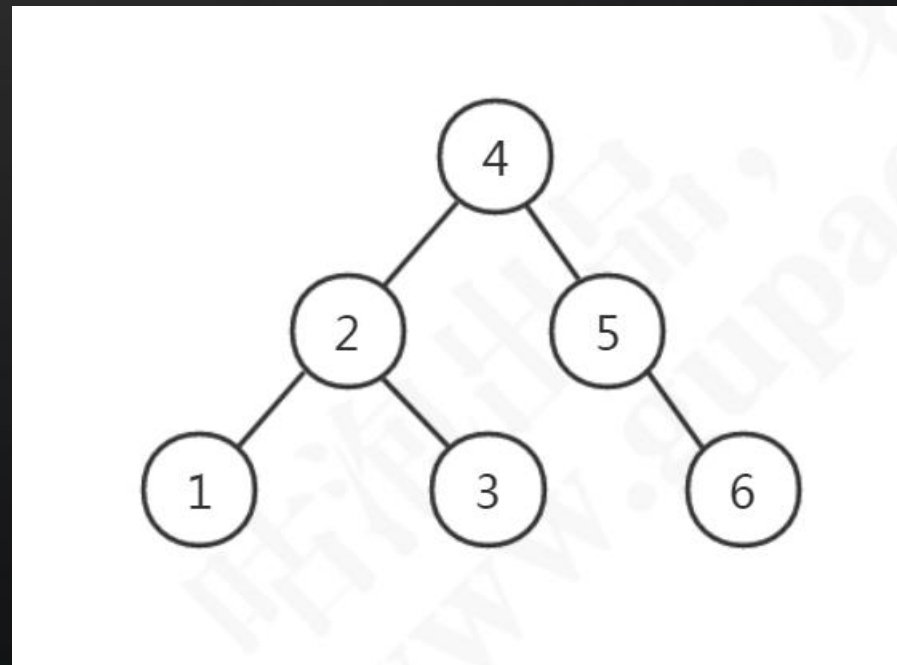


顺序插入：2、6、11、13、17、22



平衡二叉树：左右子树深度差**绝对值**不能超过1

顺序插入：1、2、3、4、5、6



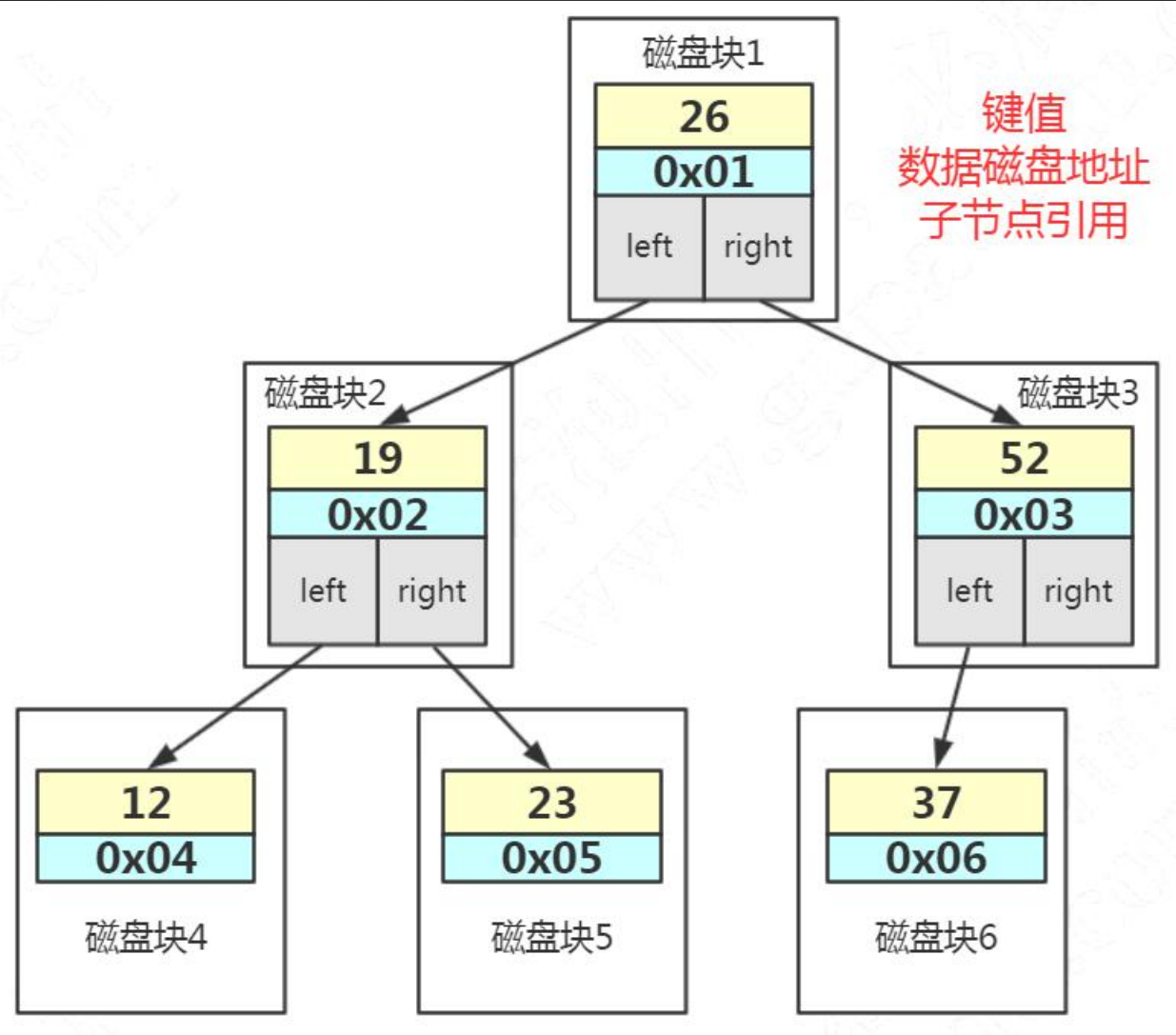
插入：1、2
再插入：3

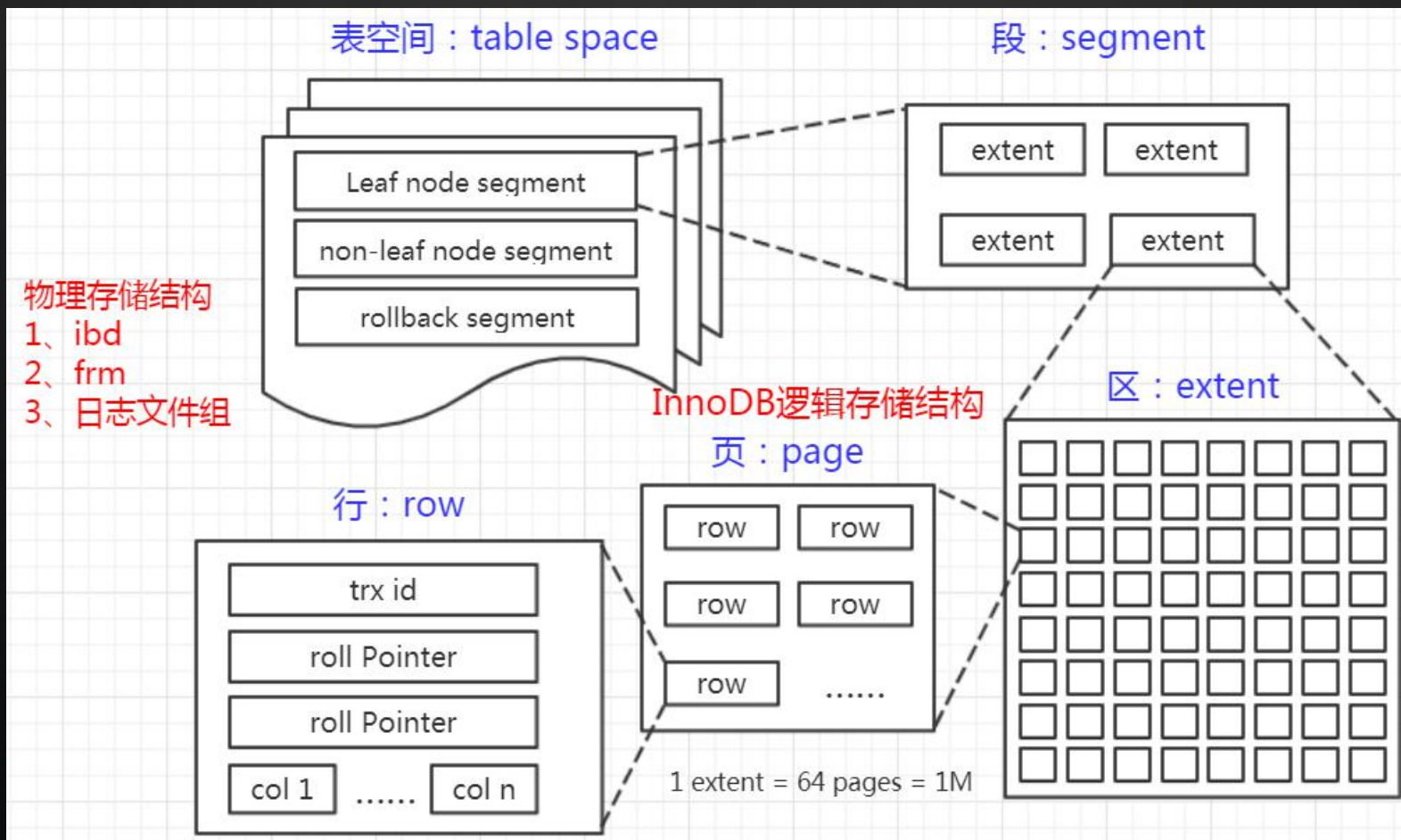


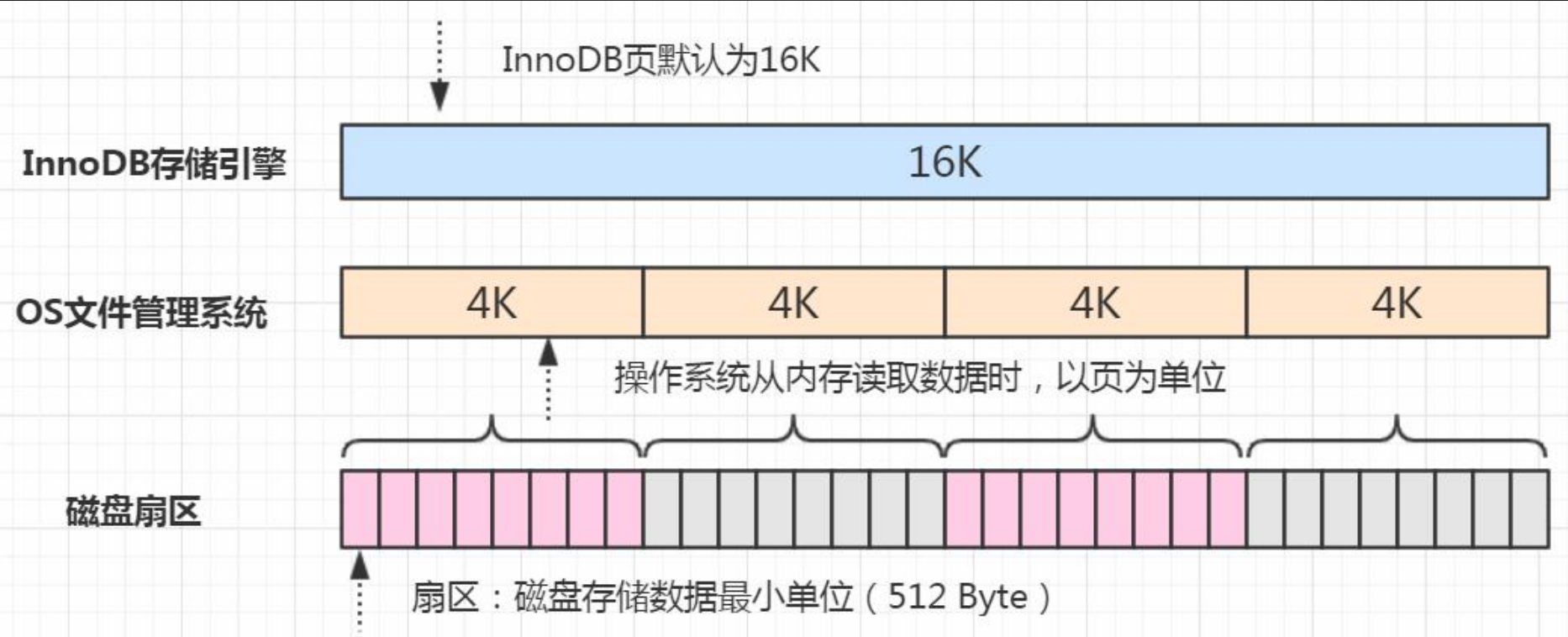
插入：7、6

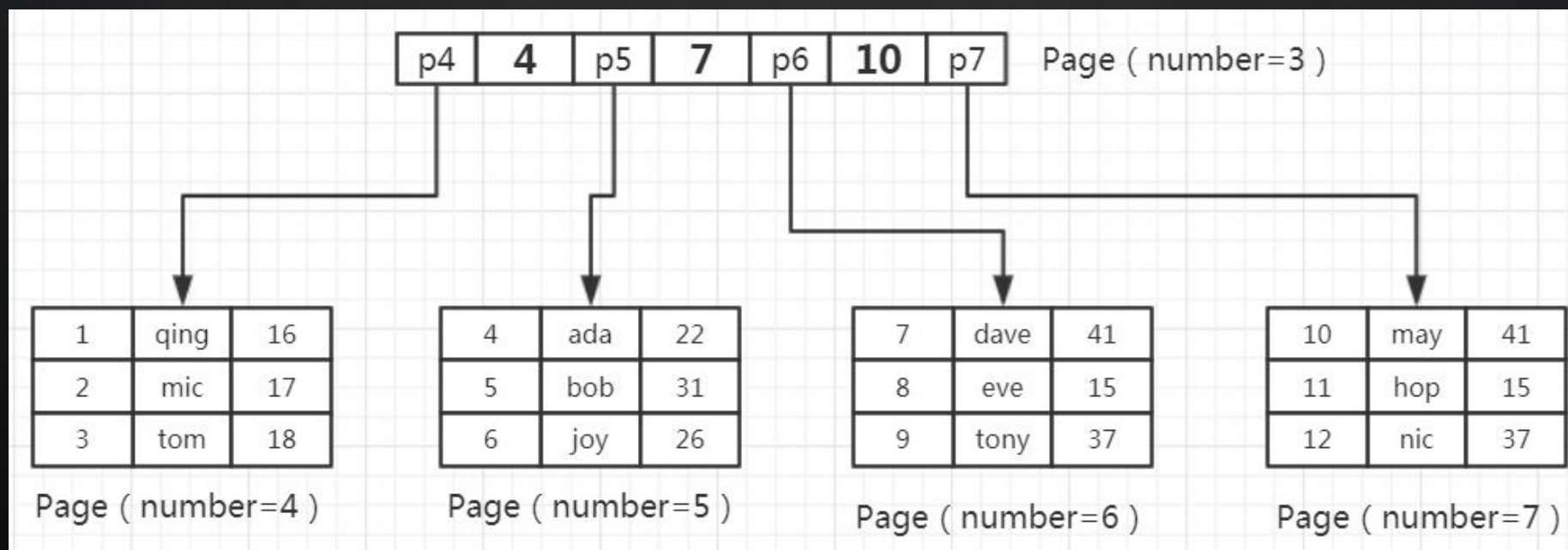
再插入：5

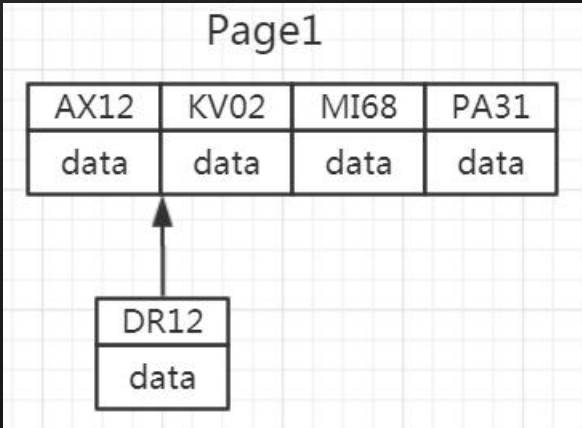
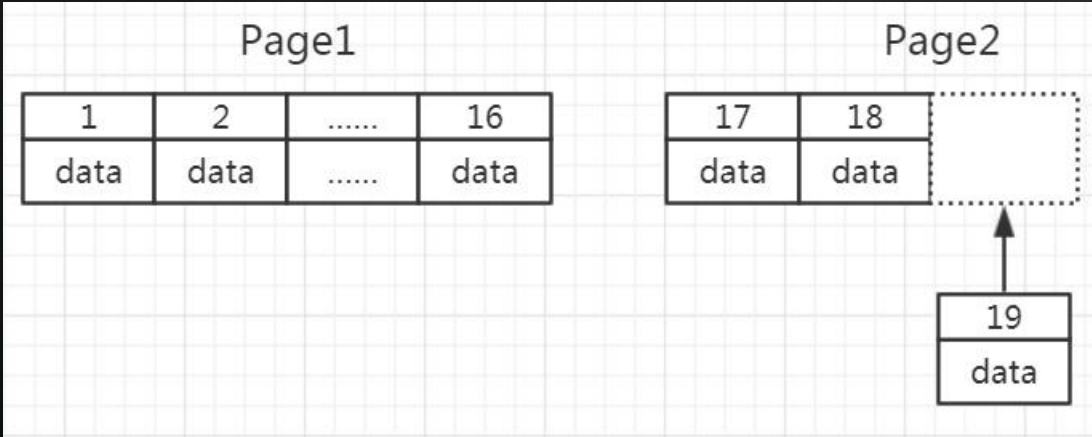












Antelope['æntɪləʊp] (羚羊) 是InnoDB内置的文件格式，有两种行格式：

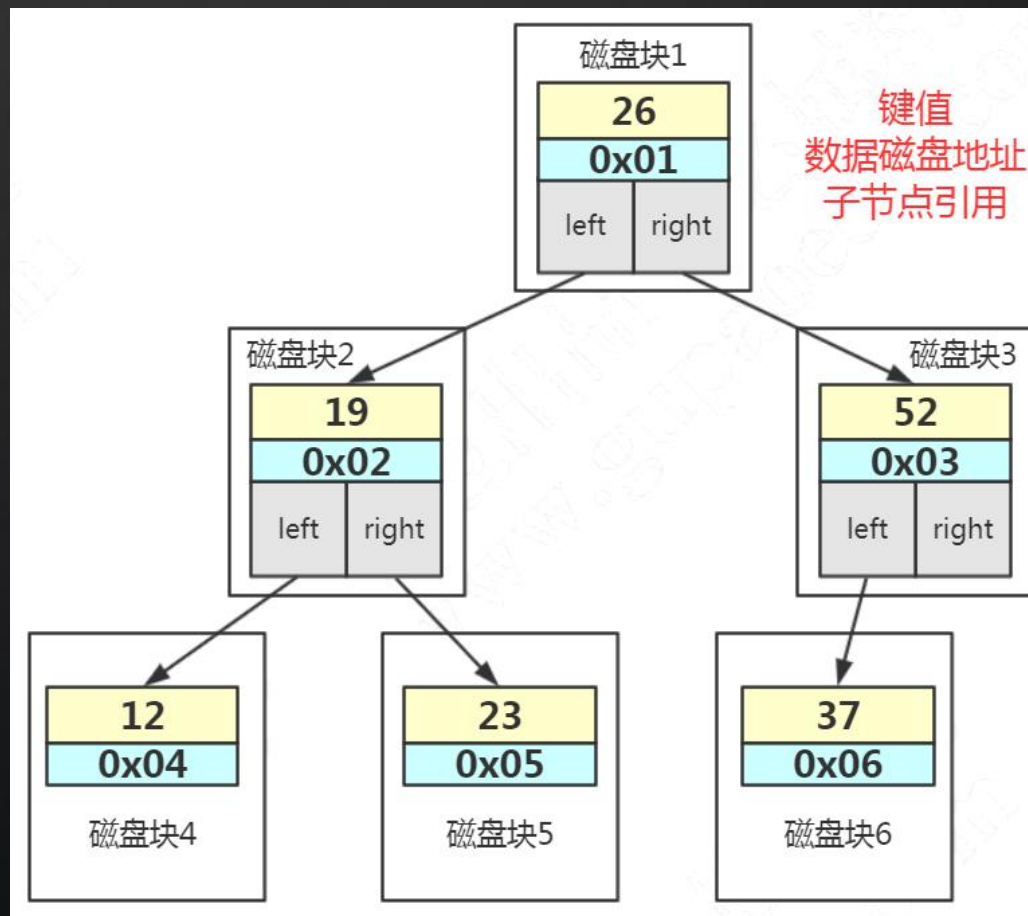
- REDUNDANT[rɪ'dʌndənt] Row Format
- COMPACT Row Format (5.6默认)

Barracuda[,bærə'kju:də] (梭子鱼) 是InnoDB Plugin支持的文件格式，新增了两种行格式：

- DYNAMIC Row Format (5.7默认)
- COMPRESSED Row Format



- 操作系统与磁盘交互
- 树的深度

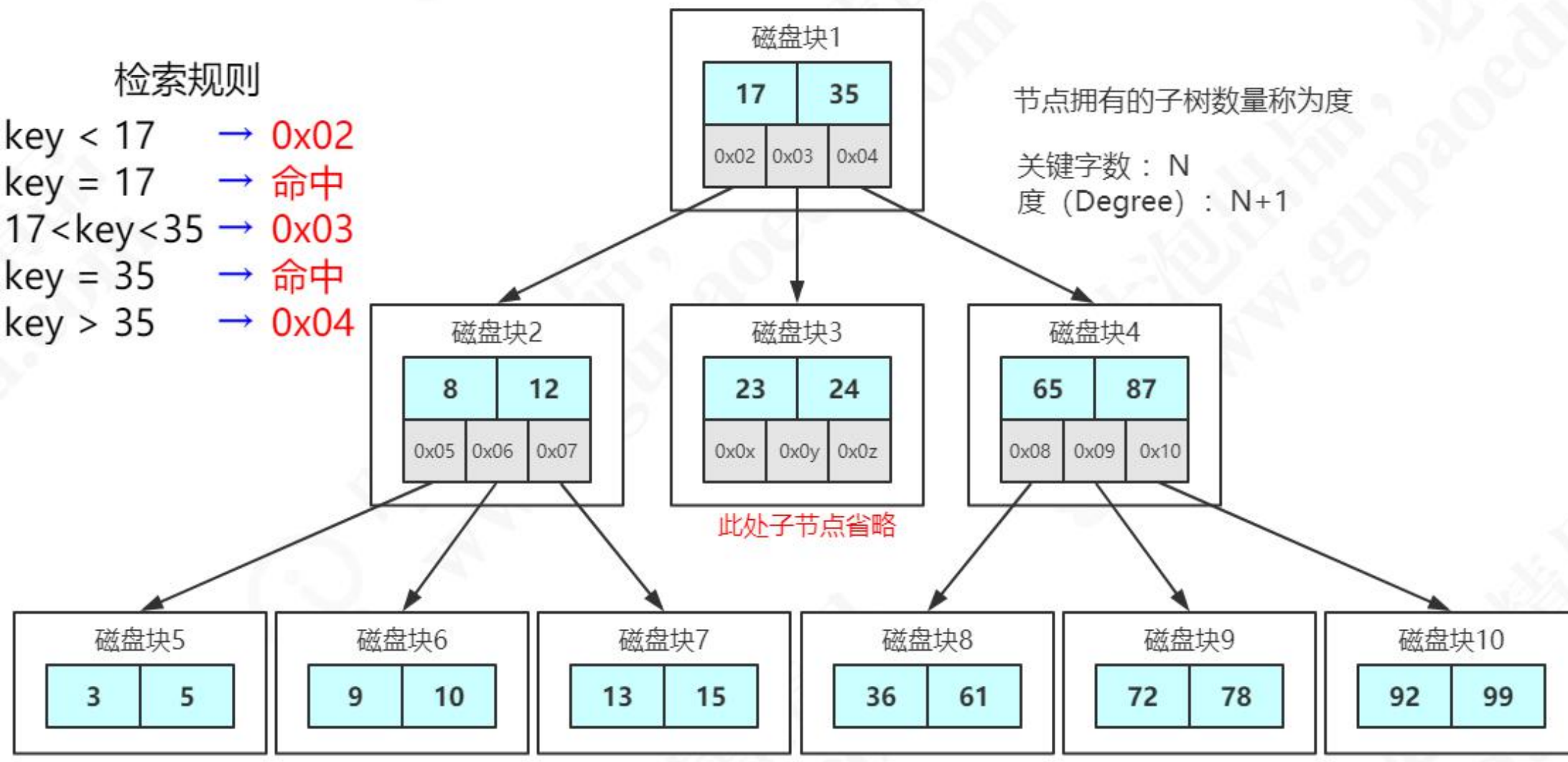


检索规则

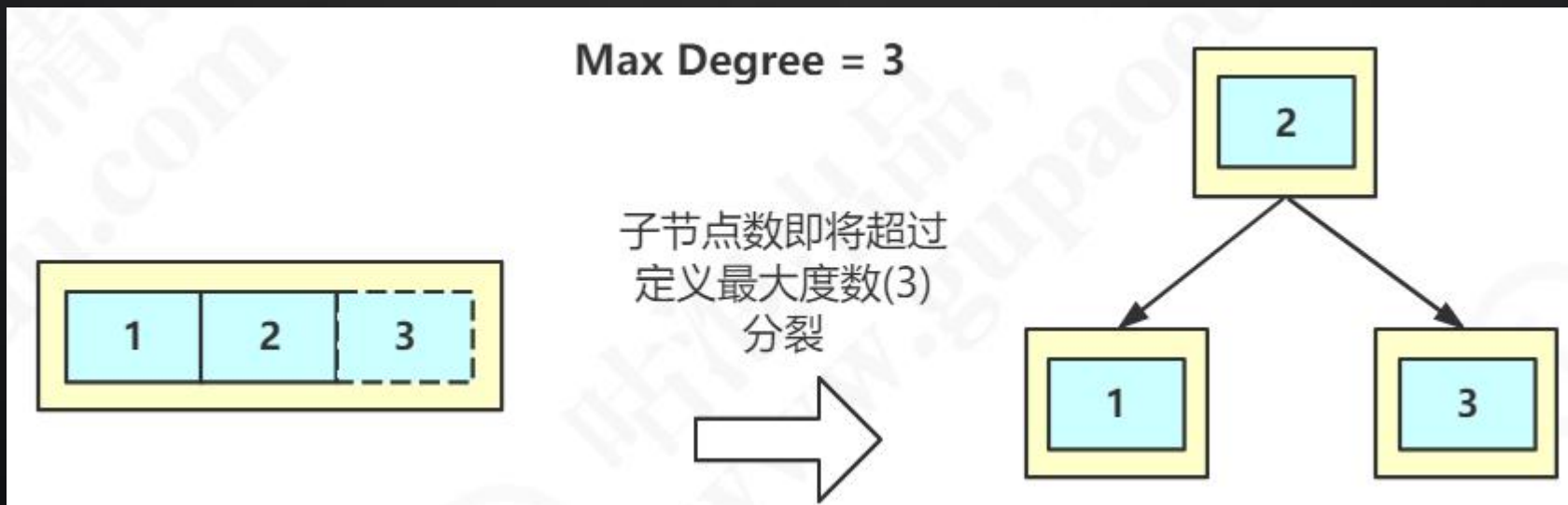
key < 17 → 0x02
key = 17 → 命中
17 < key < 35 → 0x03
key = 35 → 命中
key > 35 → 0x04

节点拥有的子树数量称为度

关键字数：N
度 (Degree)：N+1



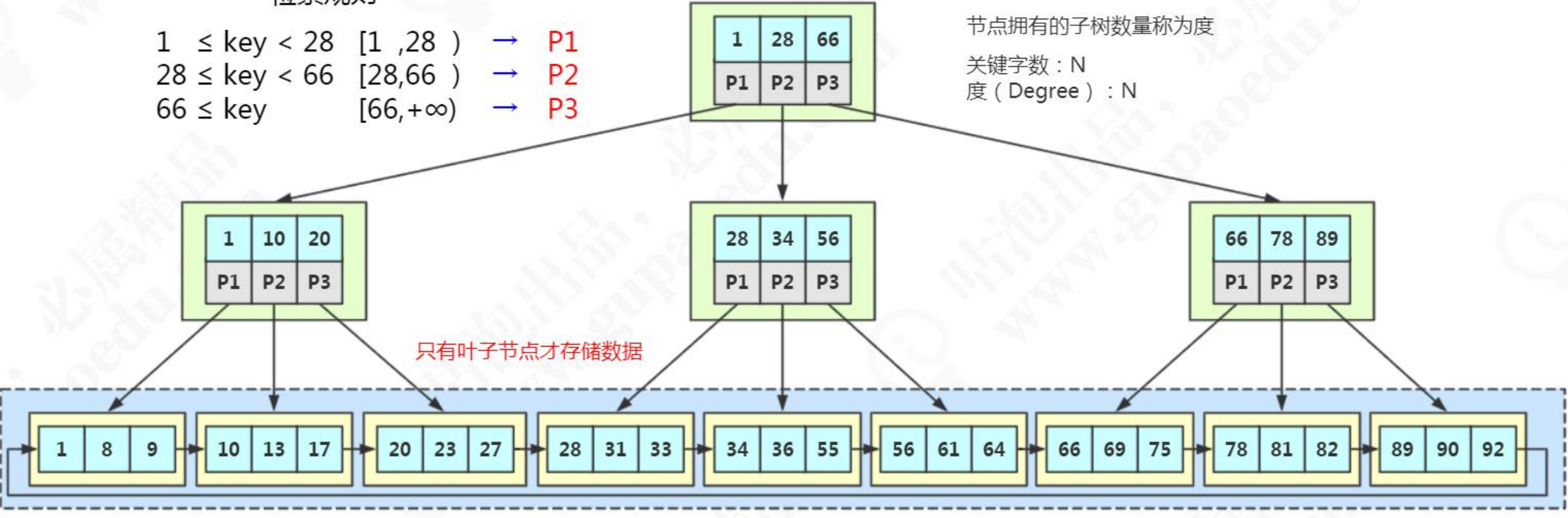
● 分裂、合并

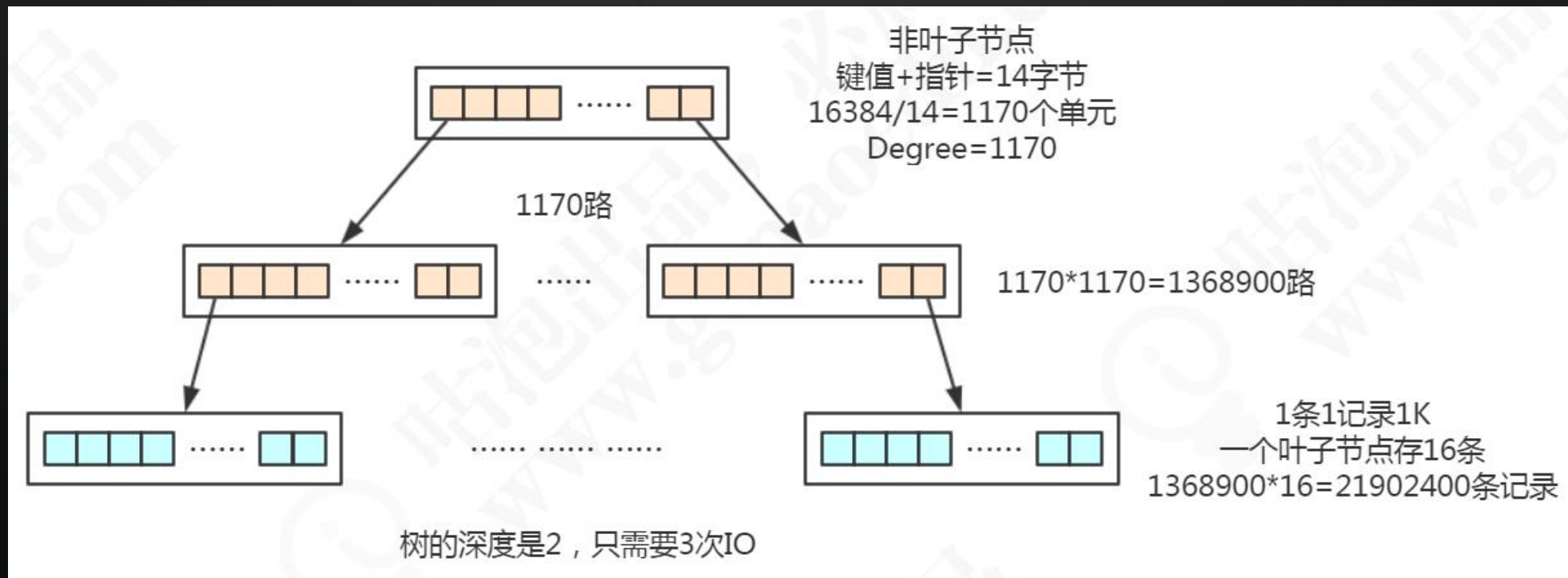


检索规则

$1 \leq \text{key} < 28$ $[1, 28)$ \rightarrow P1
 $28 \leq \text{key} < 66$ $[28, 66)$ \rightarrow P2
 $66 \leq \text{key}$ $[66, +\infty)$ \rightarrow P3

节点拥有的子树数量称为度
关键字数：N
度 (Degree)：N





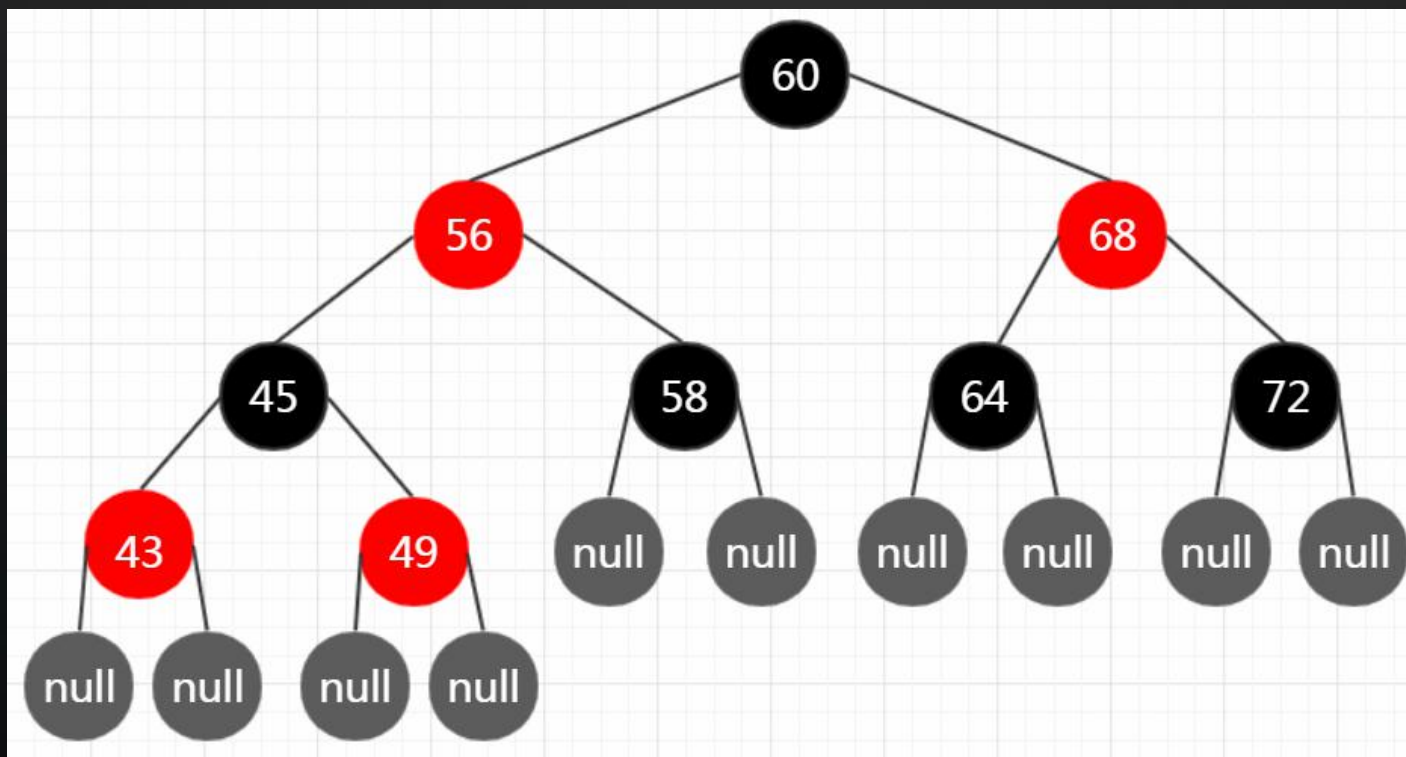
- B Tree能解决的问题，B+Tree都能解决
- 扫库、扫表能力更强
- 磁盘读写能力更强
- 排序能力更强
- 效率更加稳定



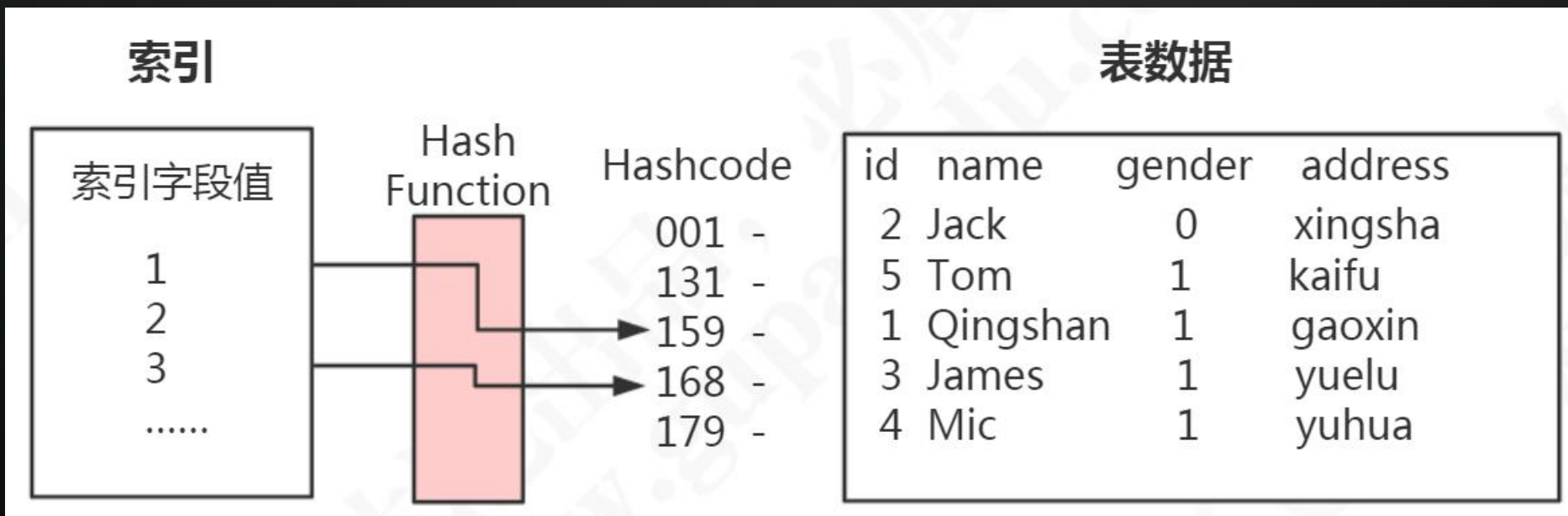
- 1、节点分为红色或者黑色。
- 2、根节点必须是黑色的。
- 3、叶子节点都是黑色的NULL节点。
- 4、红色节点的两个子节点都是黑色（不允许两个相邻的红色节点）。
- 5、从任意节点出发，到其每个叶子节点的路径中包含相同数量的黑色节点。



插入：60、56、68、45、64、58、72、43、49



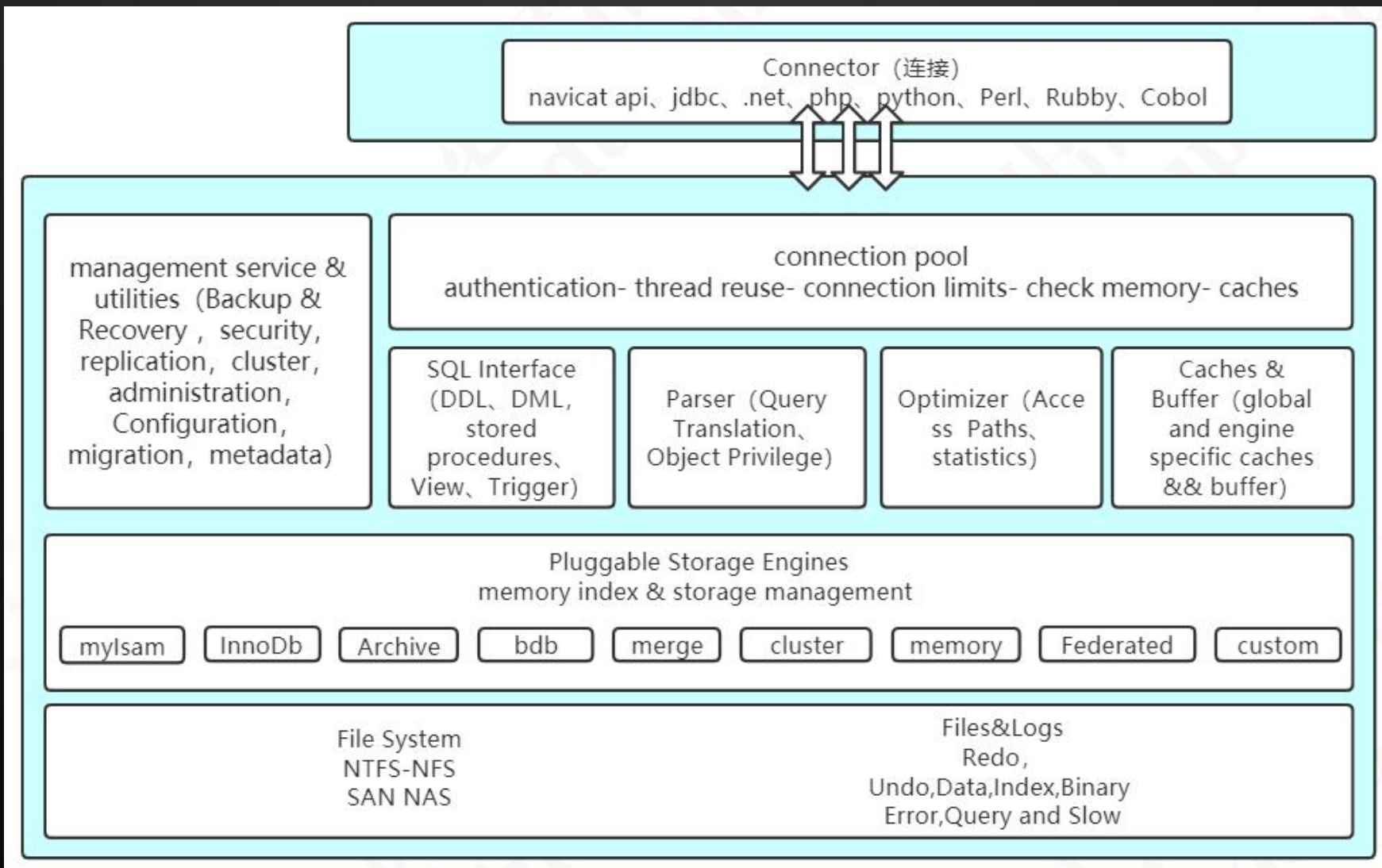
索引方式：Hash、B Tree



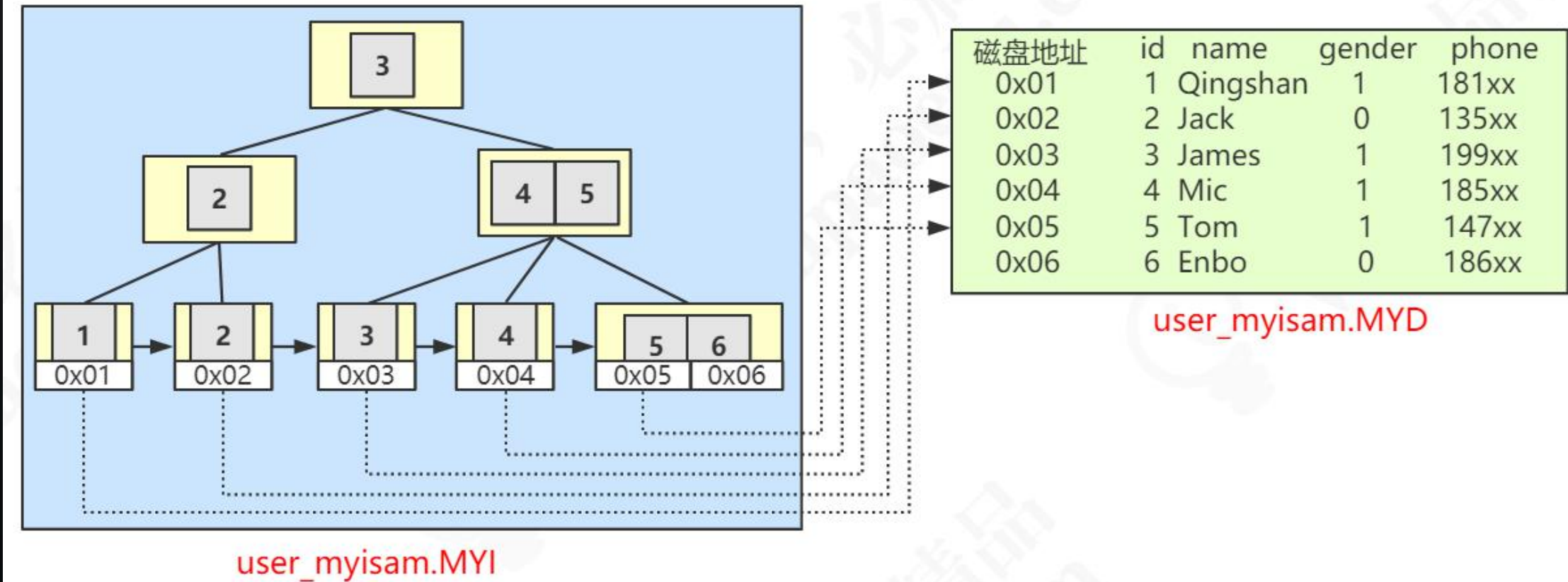
3

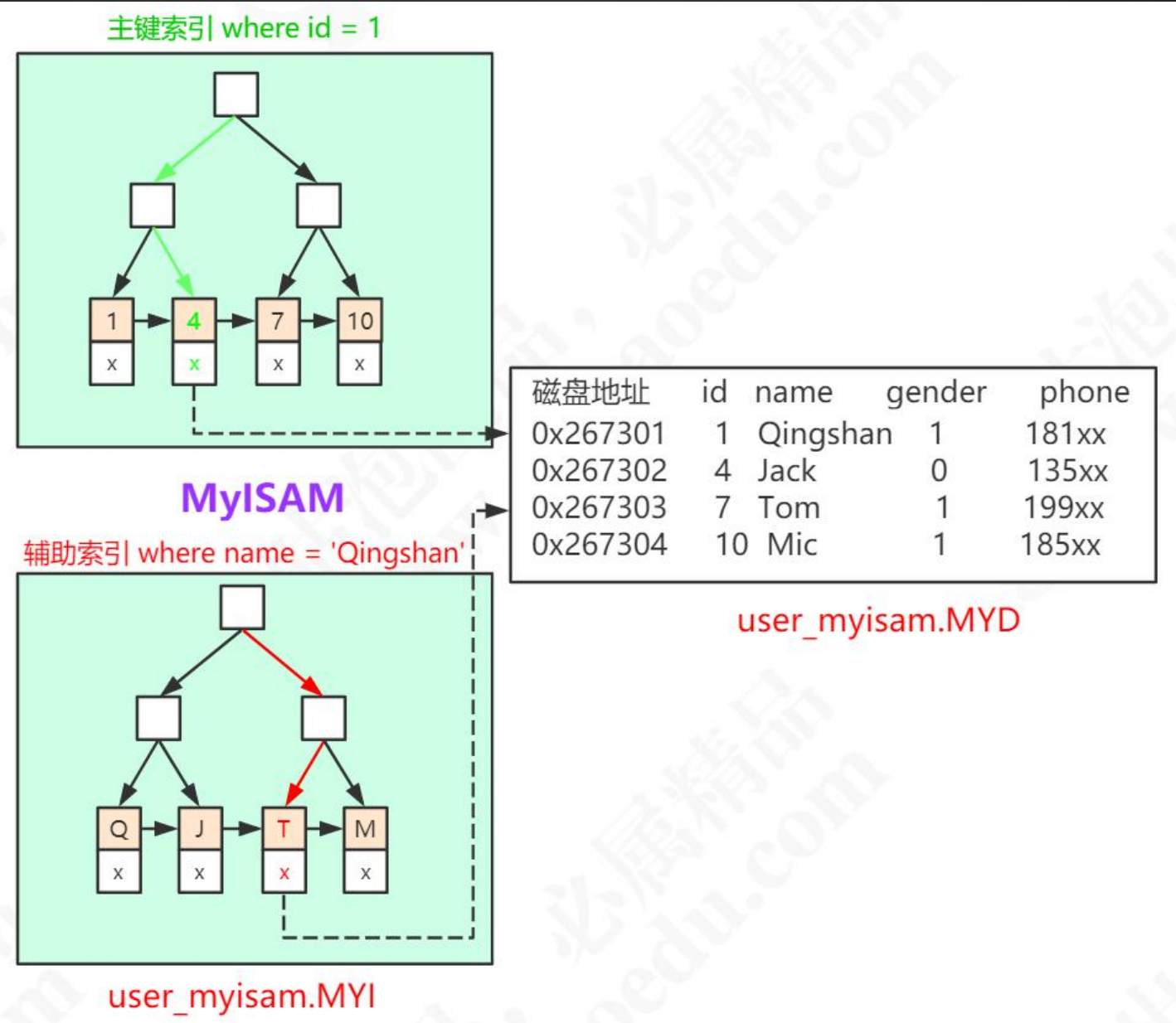
存储引擎中索引如何落地？

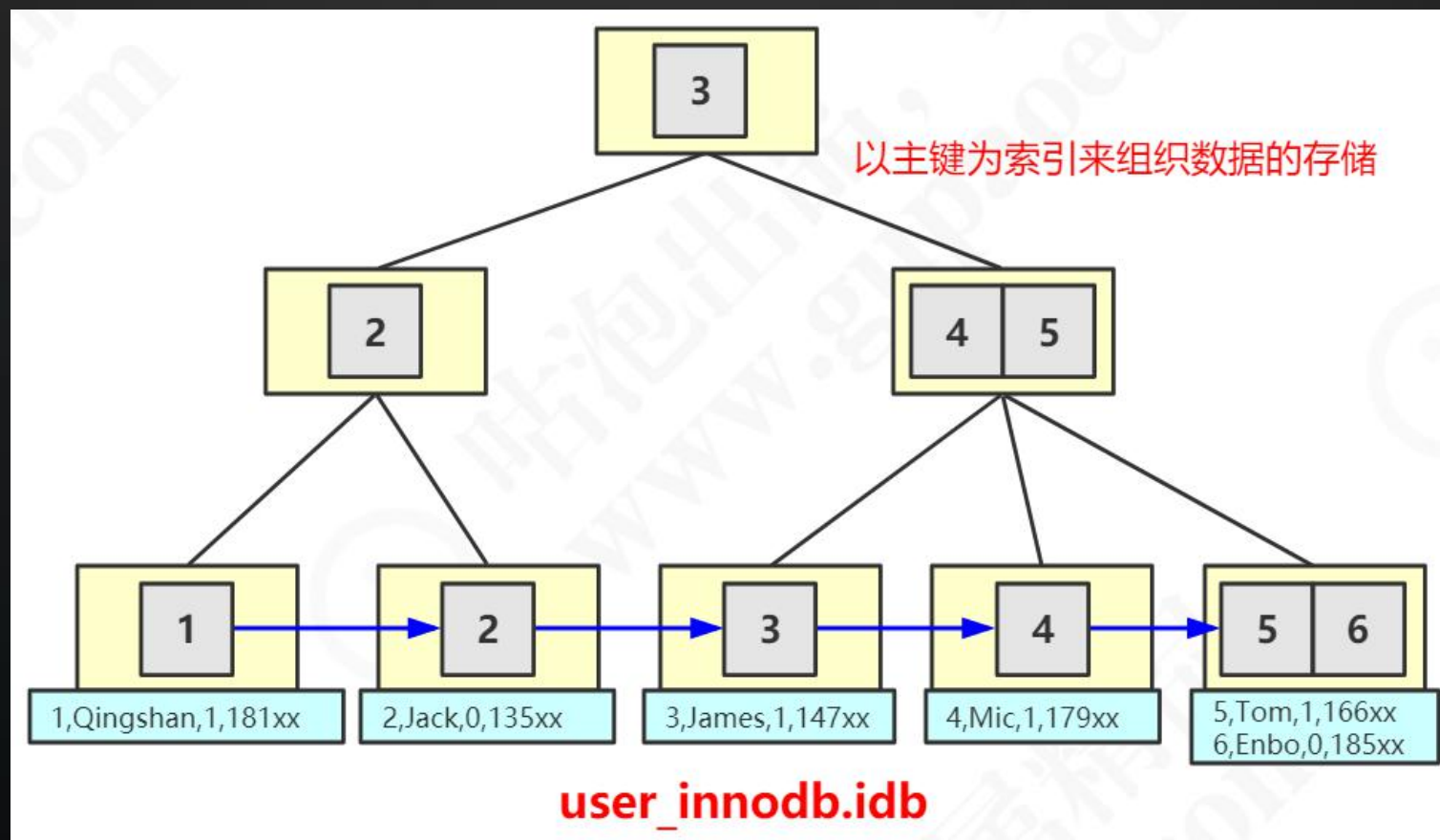


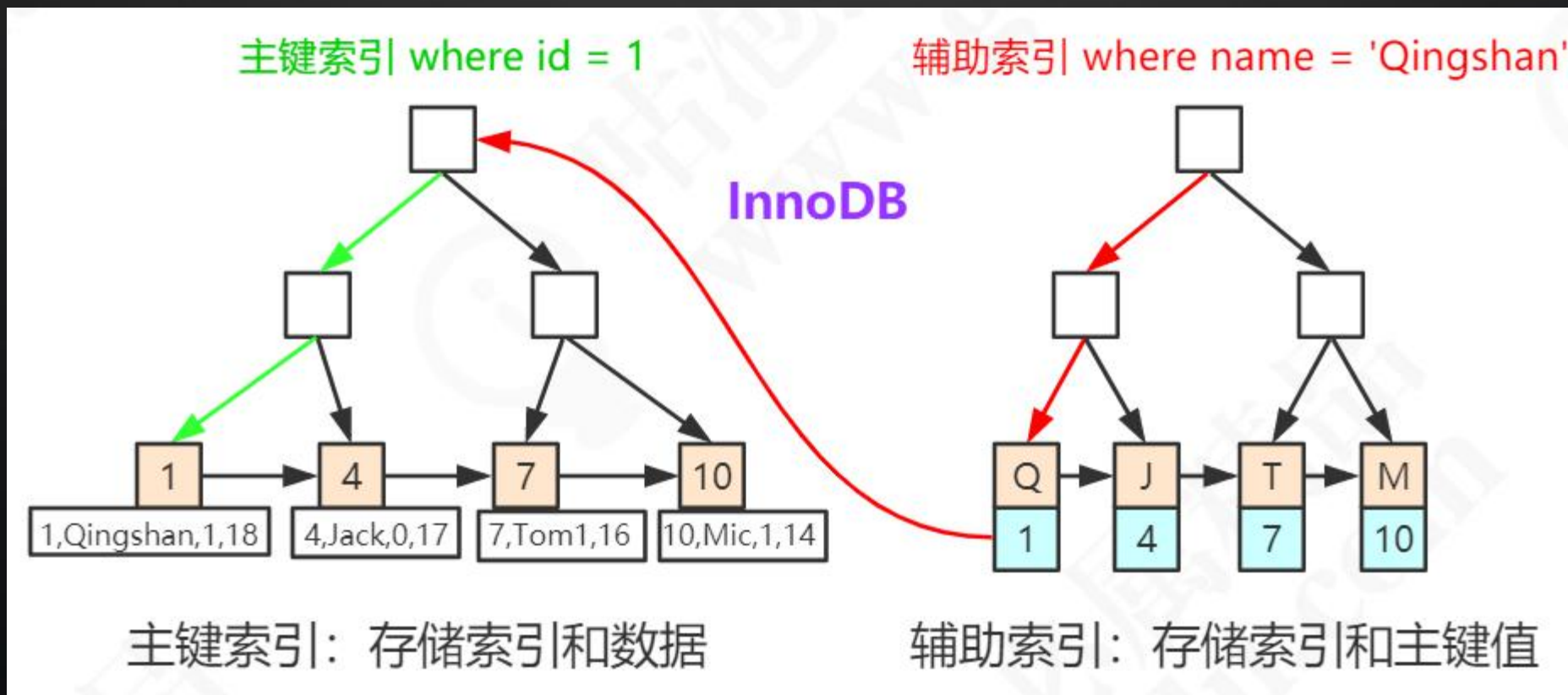


MyISAM









● 没有主键索引怎么办？

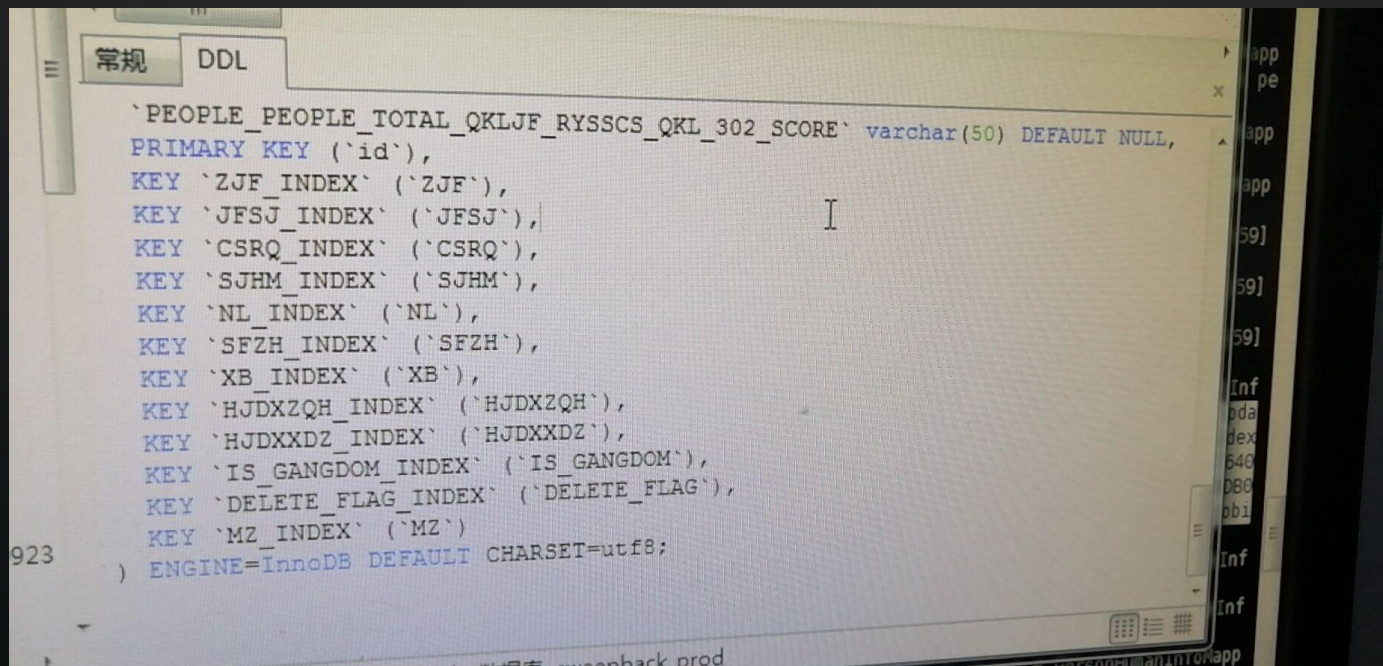


4

索引的使用原则



在所有的字段上创建索引|.....



```

    常规  DDL
    `PEOPLE_PEOPLE_TOTAL_QKLJF_RYSSCS_QKL_302_SCORE` varchar(50) DEFAULT NULL,
    PRIMARY KEY (`id`),
    KEY `ZJF_INDEX` (`ZJF`),
    KEY `JFSJ_INDEX` (`JFSJ`),
    KEY `CSRQ_INDEX` (`CSRQ`),
    KEY `SJHM_INDEX` (`SJHM`),
    KEY `NL_INDEX` (`NL`),
    KEY `SFZH_INDEX` (`SFZH`),
    KEY `XB_INDEX` (`XB`),
    KEY `HJDXZQH_INDEX` (`HJDXZQH`),
    KEY `HJDXXDZ_INDEX` (`HJDXXDZ`),
    KEY `IS_GANGDOM_INDEX` (`IS_GANGDOM`),
    KEY `DELETE_FLAG_INDEX` (`DELETE_FLAG`),
    KEY `MZ_INDEX` (`MZ`)
    923 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
  
```



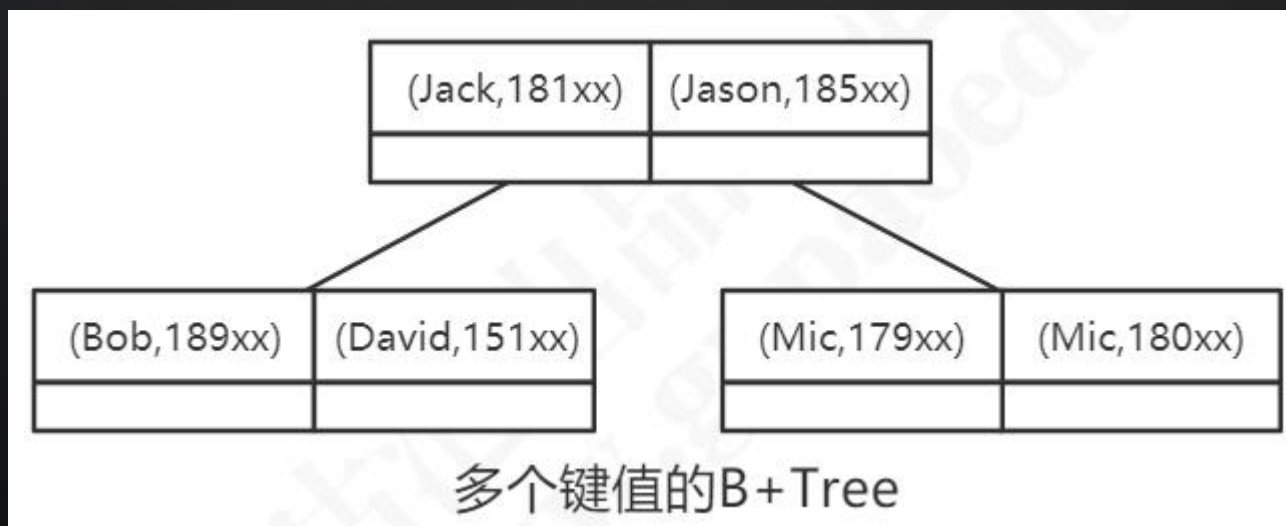
离散度公式： $\text{count}(\text{distinct}(\text{column_name})) : \text{count}(*)$

id	name	gender	phone
1	青山	0	13666666666
2	岑粒	0	13800722654
3	章机轻	1	16607463532
4	于琴旺	0	15205286470
5	皮明业	1	15901557881
6	伊颢	0	15003812562
7	宁慨	1	19900275915
8	明鹏	0	15104237001
9	蒋椴	0	13206846562
10	魏仿巡	1	15104584161

gender 和 phone，哪一列的离散度更高？



```
ALTER TABLE user_innodb  
add INDEX `comidx_name_phone` (`name`,`phone`);
```



INDEX `comidx_name_phone` (`name`,`phone`);

① SELECT * FROM user_innodb WHERE name= '青山' AND phone = '13666666666';

② SELECT * FROM user_innodb WHERE phone = '13666666666' AND name= '青山';

③ SELECT * FROM user_innodb WHERE name= '青山';

④ SELECT * FROM user_innodb WHERE phone = '13666666666';



下面这种做法是对的吗？

常用的查询SQL：

```
> SELECT * FROM user_innodb WHERE name= ? AND phone = ?;
```

```
> SELECT * FROM user_innodb WHERE name= ?;
```

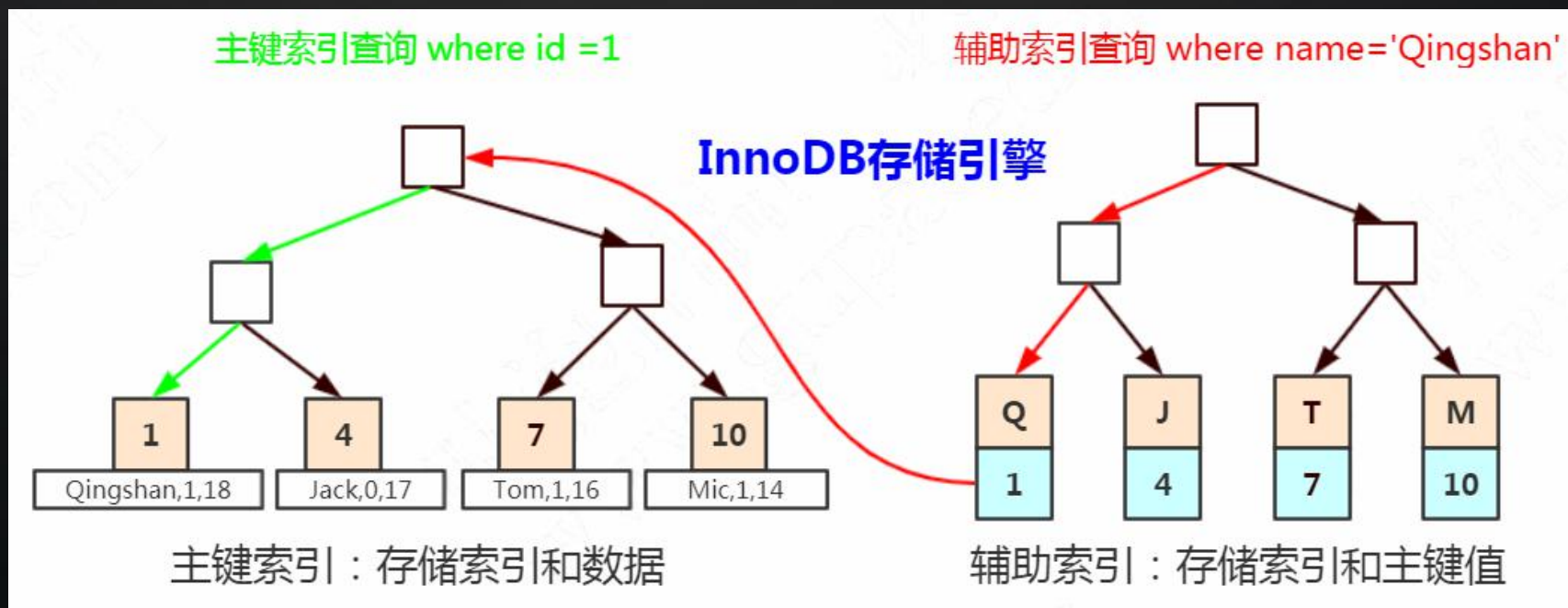
以下做法对吗？

```
① CREATE INDEX idx_name on user_innodb(name);
```

```
② CREATE INDEX idx_name_phone on user_innodb(name,phone);
```



- 什么是回表？
- 什么是覆盖索引？



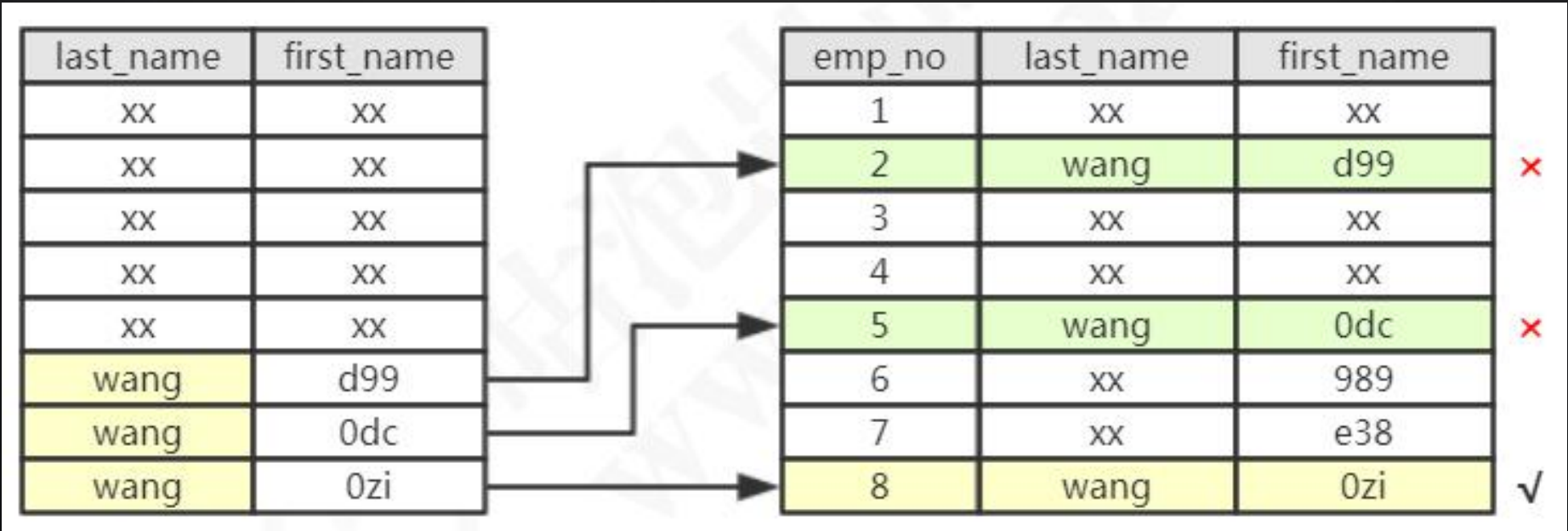
```
INDEX `comidx_name_phone` (`name`,`phone`);
```

```
EXPLAIN SELECT * FROM user_innodb WHERE name= '青山' ;
```

```
EXPLAIN SELECT name,phone FROM user_innodb WHERE name= '青山' ;
```

```
EXPLAIN SELECT name FROM user_innodb WHERE name= '青山' AND phone = '13666666666';
```





5

索引的创建与使用原则



- 1、在用于where判断order排序和join的（ on ）字段上创建索引。
- 2、索引的个数不要过多。
- 3、区分度低的字段，例如性别，不要建索引。
- 4、频繁更新的值，不要作为主键或者索引。
- 5、符合索引把散列性高（区分度高）的值放在前面。
- 6、创建复合索引，而不是修改单列索引。
- 7、过长的字段，怎么建立索引？
- 8、为什么不建议用无序的值（例如身份证、UUID）作为索引？



- 1、索引列上使用函数 (replace\SUBSTR\CONCAT\sum count avg)、表达式
- 2、字符串不加引号，出现隐式转换
- 3、like条件中前面带%
- 4、负向查询能用到索引吗？ <> != NOT in



谢谢你的鼓励和支持

青山老师

专业IT教育培训，做技术人的指路明灯，职场生涯的精神导师
咕泡学院官网：<http://www.gupaoedu.com>



美洼咕泡
码上升职加薪

