RDS for MySQL 表和索引优化实战

田杰

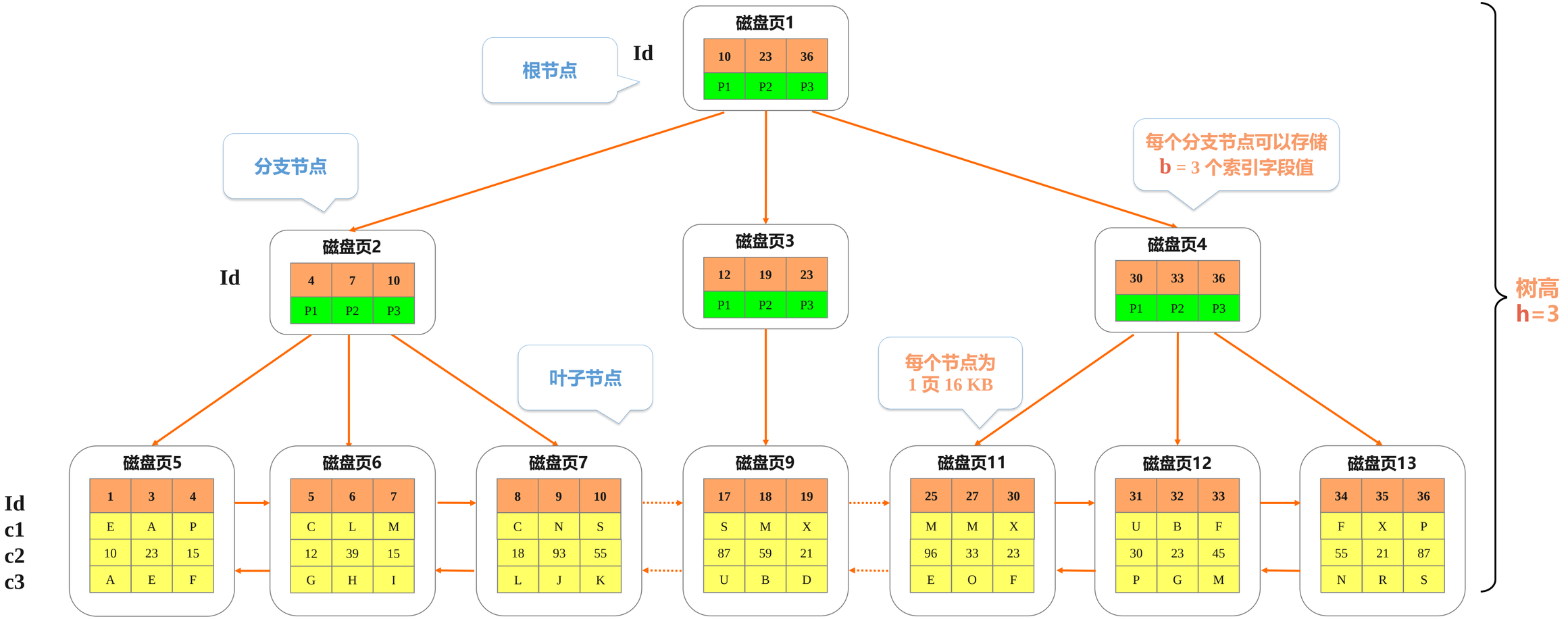
高级运维专家

RDS for MySQL 表和索引优化实战 表和索引

阿里云

主键索引

Table: (id int primary key, c1 char(1), c2 int, c3 char(1), key idx_c1 (c1));

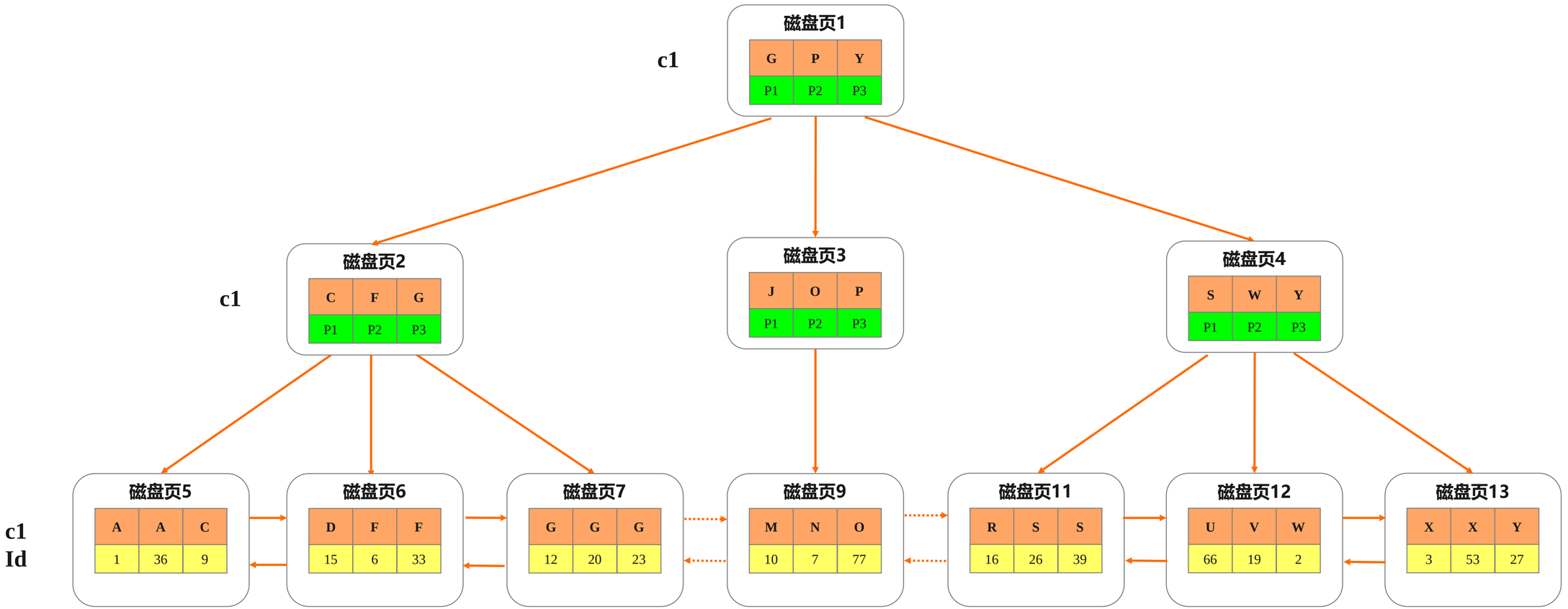


RDS for MySQL 表和索引优化实战 表和索引

阿里云

二级索引

Table: (id int primary key, c1 char(1), c2 int, c3 char(1), key idx_c1 (c1));



RDS for MySQL 表和索引优化实战 表和索引

阿里云

数学分析

#	项目	说明	举例	
1	行数 n	表内的记录行数	一亿行记录 100,000,000	
2	分支因子 b	每个 16 KB 分支节点可以存储的 索引字段 个数	b = 2	b = 100
3	树高 h	• 索引 B 树 高度：从叶子节点到根节点的节点个数 • 计算公式：$h = \lceil \log_b n \rceil = \lceil \log n / \log b \rceil$	$h = \lceil \log^{100,000,000} / \log^{2} \rceil = 27$	$h = \lceil \log^{100,000,000} / \log^{100} \rceil = 4$
4	定位 1 行记录的开销	树高 $h = \lceil \log_b n \rceil = \lceil \log n / \log b \rceil$	27	4
5	做 k 个元素的范围查询的开销	树高 $h + k = \lceil \log_b n \rceil = \lceil \log n / \log b \rceil + k$	27 + k	4 + k
6	存储空间	磁盘存储空间 = 16 KB * n / b	16 KB * 100,000,000 / 2 = 781,250 MB	16 KB * 100,000,000 / 100 = 15,625 MB

阿里云

索引的作用

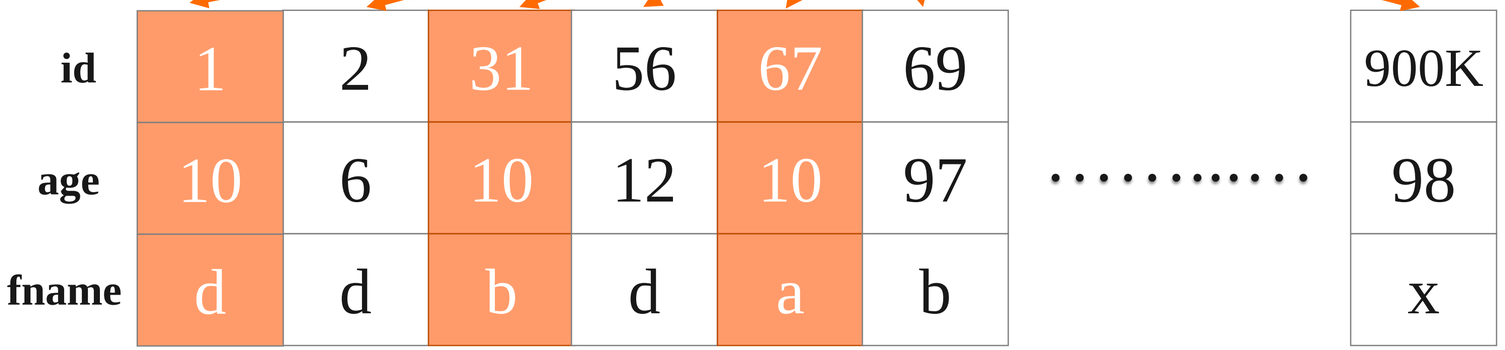
Users (id int primary key, age int, fname char(1));

Select fname from users where age = 10

RC

Primary Key

TR x 1 + TS x (n – 1)



TR = 10 ms

TS = 0.01 ms

RT = TR * 1 + TS * (n - 1)
= 10 ms * 1 + 0.01 ms * (900K -1)
= 10 ms + 9000 ms
= 9.01 Sec

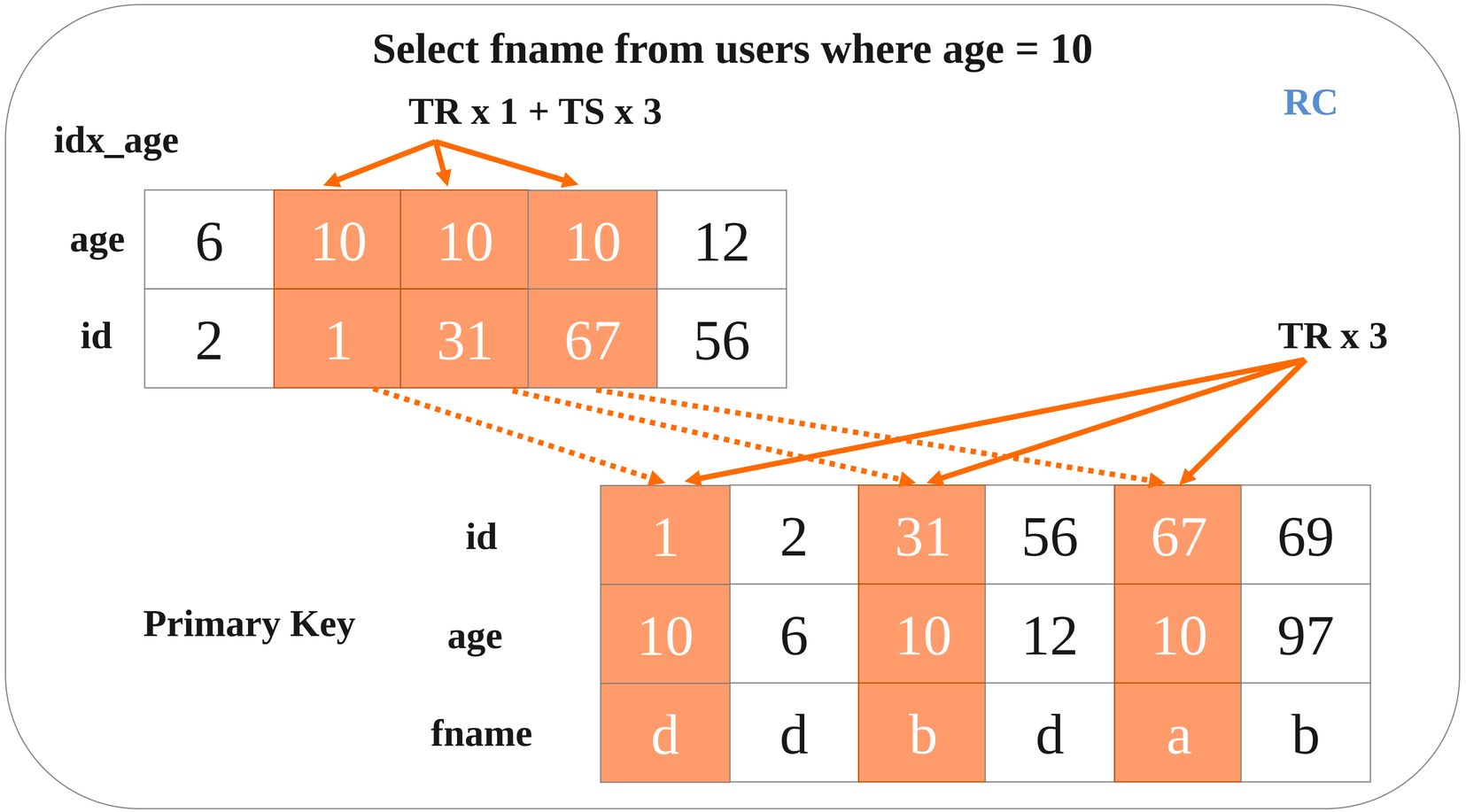
RT = TR * 1 + TS * (n - 1)
= 10 ms * 1 + 0.01 ms * (9000 -1)
= 10 ms + 90 ms
= 100 ms

RT = TR * 1 + TS * (n - 1)
= 10 ms * 1 + 0.01 ms * (9 -1)
= 10 ms + 0.08 ms
= 10.08 ms

阿里云

索引的作用

Users (id int primary key, age int, fname char(1), key idx_age (age));



TR = 10 ms

TS = 0.01 ms

RT = TR * 4 + TS * 3
= 10 ms * 4 + 0.01 ms * 3
= 40 ms + 0.03 ms
= 40.03 ms

Full Table Scan : 9.01 Sec

Index Seek : 40.03 ms

9.01 * 1000 ms / 40.03 ms = 225

阿里云

索引的作用

Users (id int primary key, age int, fname char(1), key idx_age_fname (age,fname));

Select fname from users where age = 10

RC

idx_age_fname

TR x 1 + TS x 3

age	6	10	10	10	12
fname	d	a	b	d	d
id	2	67	31	1	56

TR = 10 ms

TS = 0.01 ms

RT = TR * 1 + TS * 3
= 10 ms * 1 + 0.01 ms * 3
= 10 ms + 0.03 ms
= 10.03 ms

Full Table Scan : 9.01 Sec

Index Seek : 10.03 ms

9.01 * 1000 ms / 10.03 ms = 900

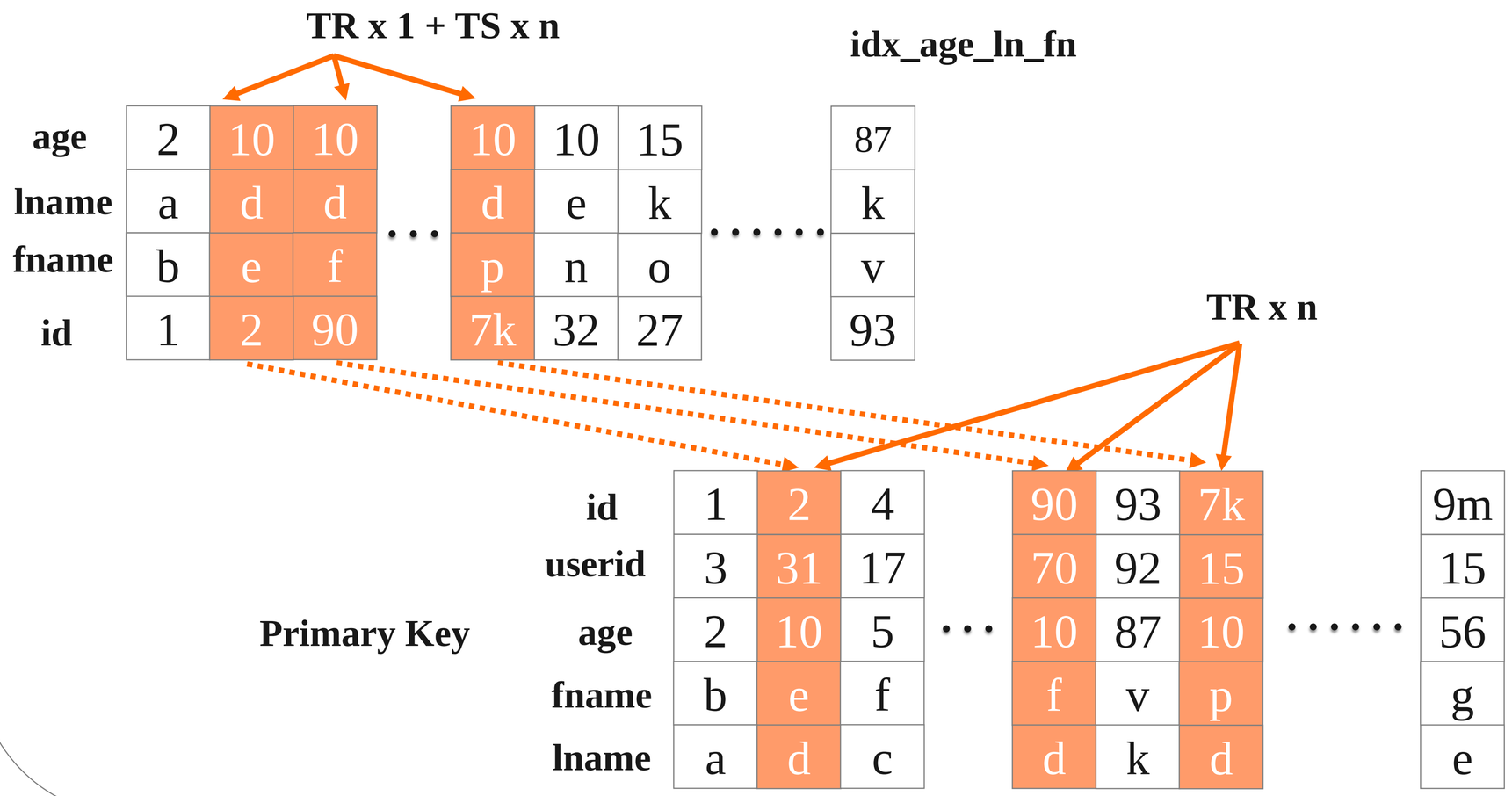
RDS for MySQL 表和索引优化实战 表和索引

阿里云

索引的作用

Users (id int primary key, userid int, age int, fname char(1), lname char(1), key idx_age_ln_fn (age,lname,fname));

Select userid, lname, fname from users where age = 10 and lname = 'd' order by fname



RC

TR = 10 ms

TS = 0.01 ms

N = 1000

TO – Time on Order (Sort)

Idx_age_ln_fn:

RT = TR * (n+1) + TS * n
= 10 ms * 1k + 0.01 ms * 1k
= 10 sec + 10 ms
= 10.01 Sec

Full Table Scan:

RT = TR * 1 + TS * 9m + TO
= 10 ms + 90 Sec + TO
= 90.01 Sec + TO

RDS for MySQL 表和索引优化实战 表和索引

阿里云

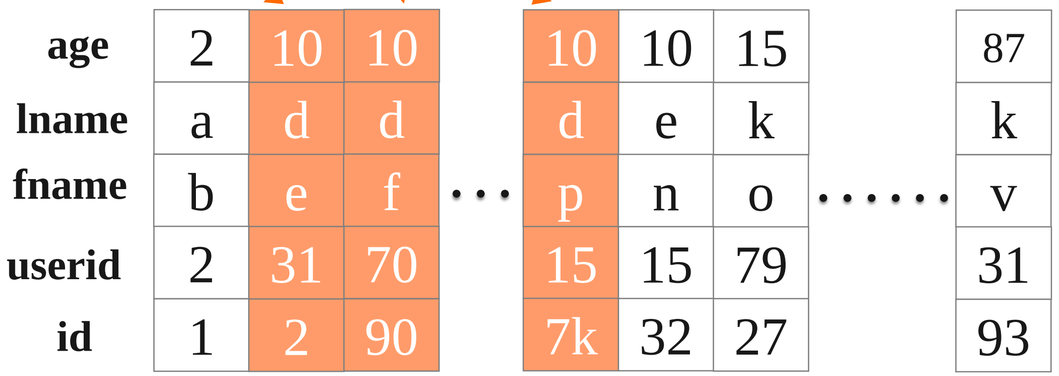
索引的作用

Users (id int primary key, userid int, age int, fname char(1), lname char(1), key idx_age_ln_fn_uid (age,lname,fname,userid));

Select userid, lname, fname from users where age = 10 and lname = 'd' order by fname

Idx_age_ln_fn_uid

TR x 1 + TS x n



RC

TR = 10 ms

TS = 0.01 ms

N = 1000

TO – Time of Order by (Sort)

Idx_age_ln_fn_uid:

RT = TR * 1 + TS * n
= 10 ms * 1 + 0.01 ms * 1k
= 10 ms + 10 ms
= 20 ms

Full Table Scan:

RT = TR * 1 + TS * 9m + TO
= 10 ms + 90 Sec + TO
= 90.01 Sec + TO

RDS for MySQL 表和索引优化实战 表和索引

阿里云

规范

#	分类	内容	举例	说明
1	基本	不要在数据库做运算	order by rand(), md5()	节省 CPU 资源，避免 打满
2		控制单表数据量	1000 万行，5 GB 以内	优化访问速度 和 DDL 时间
3		控制单表数量、字段数量	20万、50 个以内	避免宽表导致行迁移
4		平衡范式，适当冗余	避免 3 个以上 表 Join	避免过度表 Join
5		拒绝 3 B	大的 事务、SQL、批量	避免延迟、空间 等问题
6		一次查询，多次复用	多个相同查询并发执行	通过 Redis 缓存结果复用
7	字段	合适的数据类型	tinyint ~ bigint unsigned	节省存储空间，避免行迁移
8		避免 Null	int(11) not null default 0	字段设置默认值，避免异常
9		避免 & 拆分 Text / Blob	content text	大字段拆分到单独表
10	索引	合理索引，组合索引优先	key (a,b,c) / key (a)	1. 依据 查询 决定索引 2. 控制单表索引数量 3. 左前缀匹配原则 4. 最频繁查询建立覆盖索引 5. Join 字段上建立索引
11		避免索引计算	unix_timestamp(c1) > xxxxxxxx	1. 避免索引失效 2. 避免大量调用函数
12		无符号整型自增类型主键	int unsigned auto_increment	1. 避免无主键 2. 避免 uuid 做主键
13		避免外键	foreign key	应用实现业务逻辑

规范

#	分类	内容	举例	说明
14	SQL	保持语句简单	多个子查询嵌套	1. 拆分 或 应用实现 2. 一个 SQL 使用一个 CPU
15		保持语句符合 DB “思维”	子查询 ➔ 表 Join	便于优化器选择最优执行计划
16		尽量减少返回数据量	select *	1. 只选择需要的字段 2. 避免框架自动生成类似语句
17		提高分页查询效率	limit 300000,50	避免分页查询带来性能影响
18		字段类型要一致	int = ‘3’;	避免隐式转换
19	行为	隔离线上、线下环境	开发、测试、预发布、生产	1. 各个环境隔离 2. 测试后才可以上线
20		禁止未审核 SQL 上线	未测试引入性能问题	专人审核评估
21		统一字符集、字符序	utf8 = latin1	1. 避免性能问题 2. 避免异常
22		统一命名规范	表名 order	1. 要望文生义 2. 避免使用关键字
23		大数据量操作有报备	凌晨拉取全量数据 凌晨清理数据 凌晨加载大量数据	1. 避免性能问题 2. 考虑数据安全
24		隔离 应用 运维操作	开发直接操作生产库	1. 避免误操作 2. 避免性能问题
25		术业有专攻	报表直接使用生产库	1. OLAP 尽量使用独立环境 2. 模糊查询考虑 Open Search
26		尽量使用 InnoDB 引擎	Engine=Memory	综合看 InnoDB 引擎比较均衡