http://weibo.com/junsansi

君三思 2012-10

ORACLE & MYSQL

SQL优化与索引技巧

古人云：

授人以鱼，

不如授人以渔。

你应该知道的一些知识

磁盘所做操作不过READ & WRITE，谁在决定读写效率

磁盘I/O子系统体系结构

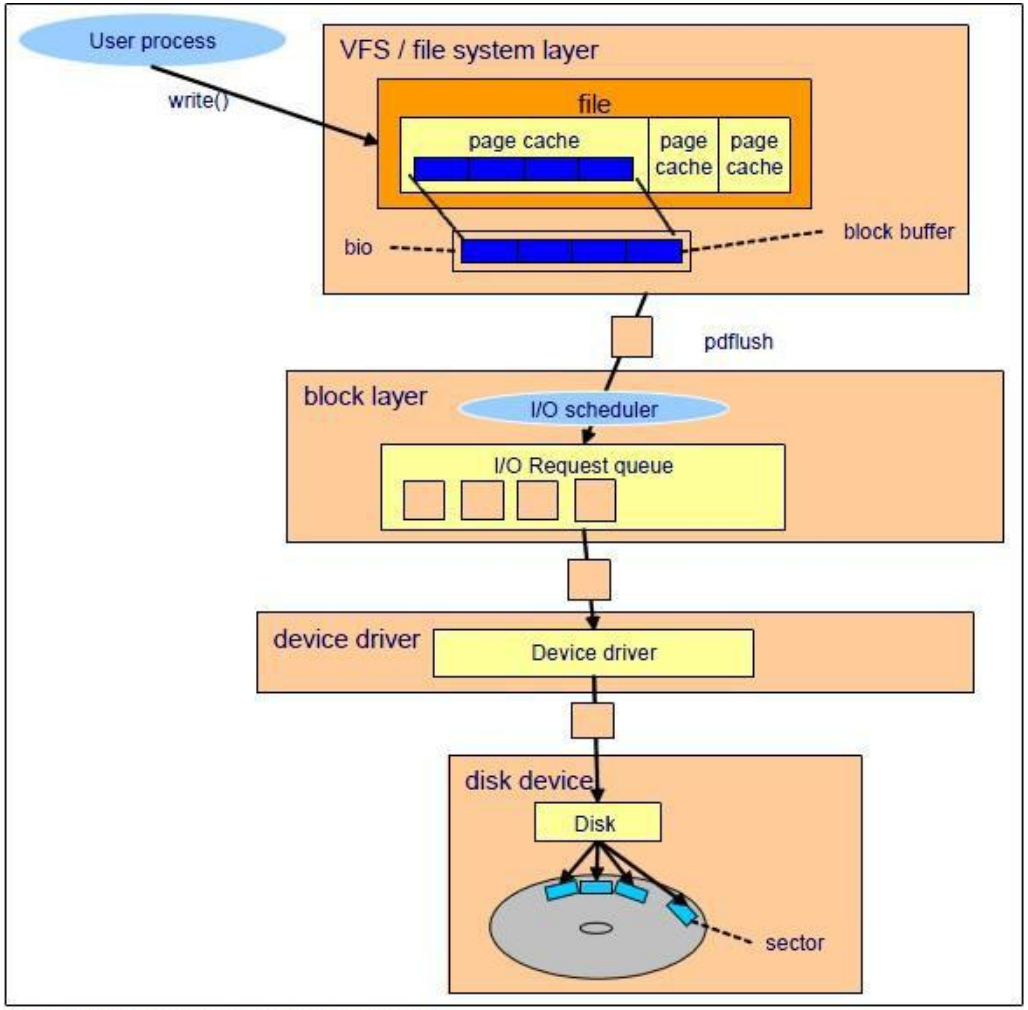


Figure 1-18 I/O subsystem architecture

磁盘I/O性能？

IOPS

全称I/O per second，即每秒进行读写（I/O）操作的次数

SATA II 7200 IOPS: 90

SAS 15K IOPS: 160

SSD IOPS:15000-100000

THROUGHPUT

吞吐量，衡量顺序访问的性能

SATA II 7200 RPM IOPS: 60M

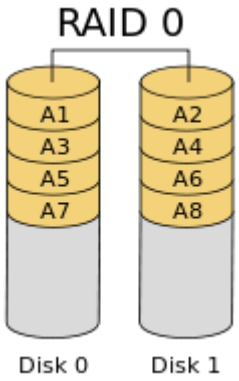
SAS 15K RPM IOPS: 150M

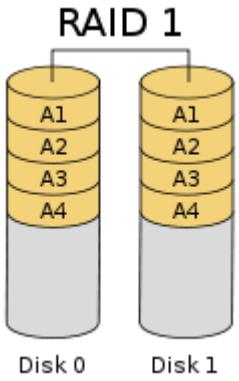
SSD IOPS: 300-500M

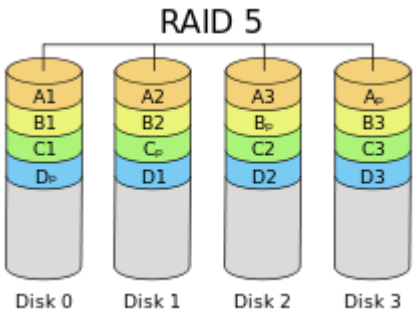
选择RAID类型？

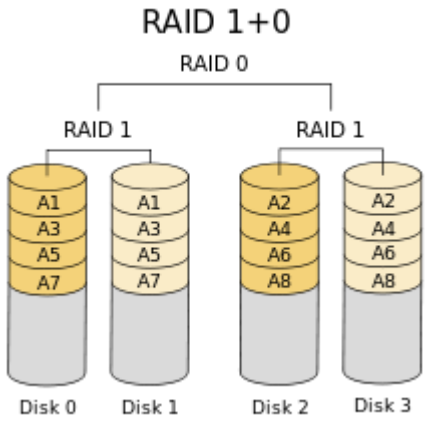
常用：

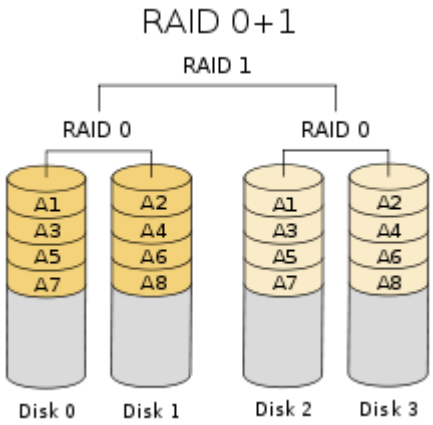
- RAID 0
- RAID 1
- RAID 5
- RAID 10
- RAID 0+1 & RAID 1+0











Tape is Dead

Disk is Tape

Flash is Disk

RAM Locality is King

--Jim Gray 2006

SQL语句优化技巧

核心：尽可能少读、尽可能少写

灵活利用现有功能

优化前：00:00:01.77s

```
select t.*,
       su.DisplayName,
       su.TypeId,
       su.HeadPicture,
       su.Email,
       sur.Role_Id
  from (select *
          from (SELECT mu.Id,
                       mu.User_Id,
                       mu.ShowName,
                       mu.Description,
                       mu.MicroblogTotal,
                       mu.FollowTotal,
                       mu.FansTotal,
                       mu.CreateTime,
                       mu.IsValid
                  FROM Microblog_User mu
                 where mu.User_Id != 0
                   and mu.IsValid = 1
                 ORDER BY DBMS_RANDOM.VALUE) t
         where rownum <= 2*9) t
 INNER JOIN Sys_User su  ON t.User_Id = su.Id  AND su.IsValid = 1
  LEFT JOIN Sys_User_Role sur  ON sur.User_Id = t.User_Id
 where rownum <= 9
```

优化后：00:00:00.02s

```
select t.*,
       su.DisplayName,
       su.TypeId,
       su.HeadPicture,
       su.Email,
       sur.Role_Id
  from (select *
          from (SELECT mu.Id,
                       mu.User_Id,
                       mu.ShowName,
                       mu.Description,
                       mu.MicroblogTotal,
                       mu.FollowTotal,
                       mu.FansTotal,
                       mu.CreateTime,
                       mu.IsValid
                  FROM Microblog_User sample(1) mu
                 where mu.User_Id != 0
                   and mu.IsValid = 1) t
         where rownum <= 2 * 9) t
 INNER JOIN Sys_User su  ON t.User_Id = su.Id  AND su.IsValid = 1
  LEFT JOIN Sys_User_Role sur  ON sur.User_Id = t.User_Id
 where rownum <= 9
```

去除不必要关联和排序

优化前：00:00:03.84s

```
SELECT * FROM (SELECT A.*, ROWNUM RN
        FROM (SELECT A.ID,
                     C.NAME              AS LINKMEMBER,
                     A.JOBTYPE,
                     A.JOBTYPE           JOBNAME,
                     A.WORKYEAR,
                     A.SYS_CORELOCATION_ID,
                     E.LEVEL1NAME        CORELOCATION_LEVEL1NAME,
                     E.LEVEL2NAME        CORELOCATION_LEVEL2NAME,
                     E.LEVEL3NAME        CORELOCATION_LEVEL3NAME,
                     E.LEVEL4NAME        CORELOCATION_LEVEL4NAME,
                     E.LEVEL5NAME        CORELOCATION_LEVEL5NAME,
                     A.CREATEDATETIME,
                     C.SEX,
                     A.JOBSALARY
                FROM nxt.JOB_PERSONAL A
                LEFT JOIN nxt.SYS_USER_EXTEND C
                  ON A.USER_ID = C.USER_ID
                LEFT JOIN nxt.SYS_USER SU
                  ON A.USER_ID = SU.ID
                LEFT JOIN nxt.SYS_CORELOCATION E
                  ON A.SYS_CORELOCATION_ID = E.ID
               WHERE A.ISVALID = 1
                 AND A.EFFECTIVEDATETIME > SYSDATE
               ORDER BY A.CREATEDATETIME DESC, A.ID DESC) A
      WHERE ROWNUM <= 20)  WHERE RN > 0
```

优化后：00:00:00.01s

```
SELECT * FROM (SELECT A.*, ROWNUM RN
        FROM (SELECT A.ID,
                     C.NAME              AS LINKMEMBER,
                     A.JOBTYPE,
                     A.JOBTYPE           JOBNAME,
                     A.WORKYEAR,
                     A.SYS_CORELOCATION_ID,
                     E.LEVEL1NAME        CORELOCATION_LEVEL1NAME,
                     E.LEVEL2NAME        CORELOCATION_LEVEL2NAME,
                     E.LEVEL3NAME        CORELOCATION_LEVEL3NAME,
                     E.LEVEL4NAME        CORELOCATION_LEVEL4NAME,
                     E.LEVEL5NAME        CORELOCATION_LEVEL5NAME,
                     A.CREATEDATETIME,
                     C.SEX,
                     A.JOBSALARY
                FROM nxt.JOB_PERSONAL A
                LEFT JOIN nxt.SYS_USER_EXTEND C
                  ON A.USER_ID = C.USER_ID
                LEFT JOIN nxt.SYS_CORELOCATION E
                  ON A.SYS_CORELOCATION_ID = E.ID
               WHERE A.ISVALID = 1
                 AND A.EFFECTIVEDATETIME > SYSDATE
               ORDER BY A.ID DESC) A
      WHERE ROWNUM <= 20) WHERE RN > 0
```

修改关联方式

优化前：00:00:05.13s

```
select *
  from (SELECT mu.Id,
               mu.User_Id,
               mu.ShowName,
               mu.Description,
               mu.MicroblogTotal,
               mu.FollowTotal,
               mu.FansTotal,
               mu.CreateTime,
               mu.IsValid,
               su.DisplayName,
               su.TypeId,
               su.HeadPicture,
               su.Email,
               sr.Description SecondTypeName,
               sur.Role_Id
          FROM Microblog_User mu
         INNER JOIN Sys_User su    ON mu.User_Id = su.Id
           AND mu.IsValid = 1   AND su.IsValid = 1
          LEFT JOIN Sys_User_Role sur  ON mu.User_Id = sur.User_Id
          LEFT JOIN Sys_Role sr    ON sur.Role_Id = sr.Id
         ORDER BY mu.MicroblogTotal DESC)
 where rownum <= 10
```

优化后：00:00:00.54s

```
select a.*,
       (select su2.DisplayName
          from nxt.Sys_User su2
         where a.User_Id = su2.Id)
  from
(SELECT row_number()
       over(order by mu.MicroblogTotal desc) rn,
             mu.Id,
             mu.User_Id,
             mu.ShowName,
             mu.Description,
             mu.MicroblogTotal,
             mu.FollowTotal,
             mu.FansTotal,
             mu.CreateTime,
             mu.IsValid
          FROM Microblog_User mu
         where mu.IsValid = 1) a
 where rn <= 10
```

ORACLE数据库中没有绝对的慢，”快”与”慢”的定义都是相对的，具体问题要具体分析，优化方案也多种多样。

SQL执行慢的关键因素！

寻找较占资源的SQL语句：

Statspack & AWR/ASH

分析”慢”的原因：

Too More & Too Long

ORACLE优化器类型：

RBO & CBO

查看执行计划：

EXPLAIN & Pl/SQL developer

ORACLE优化器类型

RBO(Rule Based Optimizer)基于规则的优化方法：

- 根据数据字典，查询有无可用的索引，如果有则使用，否则不使用
- 不同的访问方法有预定好的优先级，选择优先级高的执行方法

CBO(Cost Based Optimizer)基于统计信息的优化方法：

- 需要收集统计信息
- 表有多少行，占用多少数据块
- 列有多少个Null值、不同值
- 列的最大值和最小值，及值的分布情况
- 索引的层次、结点数、叶结点数，及行的分布状况(Cluster)
- 根据一定算法算出一个成本值，选择成本值最低的执行方法，不一定使用索引。

查看SQL执行计划

- 什么是执行计划?
- 如何查看sql的执行计划：
  - 在sqlplus里设置set autotrace traceonly
  - 在PL/SQL Developer里，选中SQL语句，按F5
  - 通过explain plan for …
  - 通过查询select * from v$sql_plan
- 如何阅读执行计划?

避免在列做操作

典型的获取方式：

- select * from jsstmp1 where substr(object_name,1,4) = 'jss'
- select * from jsstmp1 where object_id/30 < 100
- select * from jsstmp1 where to_char(created,'yyyy-mm-dd')='2012-10-15'

正确的例子：

- select * from jsstmp1 where object_name like 'jss%'
- select * from jsstmp1 where object_id < 100*30
- select * from jsstmp1 where created = to_date ('2012-10-15' ,'yyyy-mm-dd')

避免列的**隐式**类型转换

错误的方式：

« select * from jsstmp1 where object_name = 10

正确的例子：

« select * from jsstmp1 where object_name = '10'

对查询列的范围限制

例如，查询指定日期前的记录：

« select * from jsstmp1 where created < to_date('2012-10-15','yyyy-mm-dd')

不足之处：由于条件范围限定太广，可能导致查询出现全表扫描。

改写后的例子：

« select * from jsstmp1 where created < to_date ('2012-10-15' ,'yyyy-mm-dd') and created > to_date('2012-7-1','yyyy-mm-dd')

语句改写：<>、OR、IN

<>：select * from jsstmp1 where status <> 'UNKNOWN'

OR：select * from jsstmp1 where status = 'VALID' or status = 'INVALID'

IN：select * from jsstmp1 where status in ('VALID','INVALID')

更近一步的例子：

« Select * from jsstmp1 where (created < to_date('2002-10-15','yyyy-mm-dd') and status = 'VALID') or (object_name='TOBJ1' and object_id >10)

改写为：

« Select * from jsstmp1 where (created < to_date('2001-10-15','yyyy-mm-dd') and status = 'VALID')

« union

« Select * from jsstmp1 where (object_name='TOBJ1' and object_id >10)

减少对象访问次数

原SQL语句：

« Select count(0) ct from jsstmp1 where object_type= 'TABLE'
union all

« Select count(0) from jsstmp1 where object_type= 'INDEX'

改写为：

« Select sum(decode(object_type, 'TABLE',1,0)) VT,
sum(decode(object_type, 'INDEX',1,0)) IT from jsstmp1 where
object_type in('TABLE', 'INDEX')

分清场景应用IN、EXISTS

总的原则：

- 小表驱动大表，使用EXISTS
- 大小驱动小表，使用IN

受CBO生成的执行计划影响较大，因此实际执行时，仍应以实际操作时最优为准则。

- 最好选择是改为表连接

索引:查询语句的关键因素

核心：HOW、WHY、 WHEN

索引的类型

逻辑分类：

- 单列索引
- 复合索引
- 主键/唯一索引
- 非唯一索引
- 函数索引

物理分类：

- B*Tree索引
- 降序索引
- Bitmap索引
- 分区/非分区索引
- 全文索引
- ……

快递员的烦恼

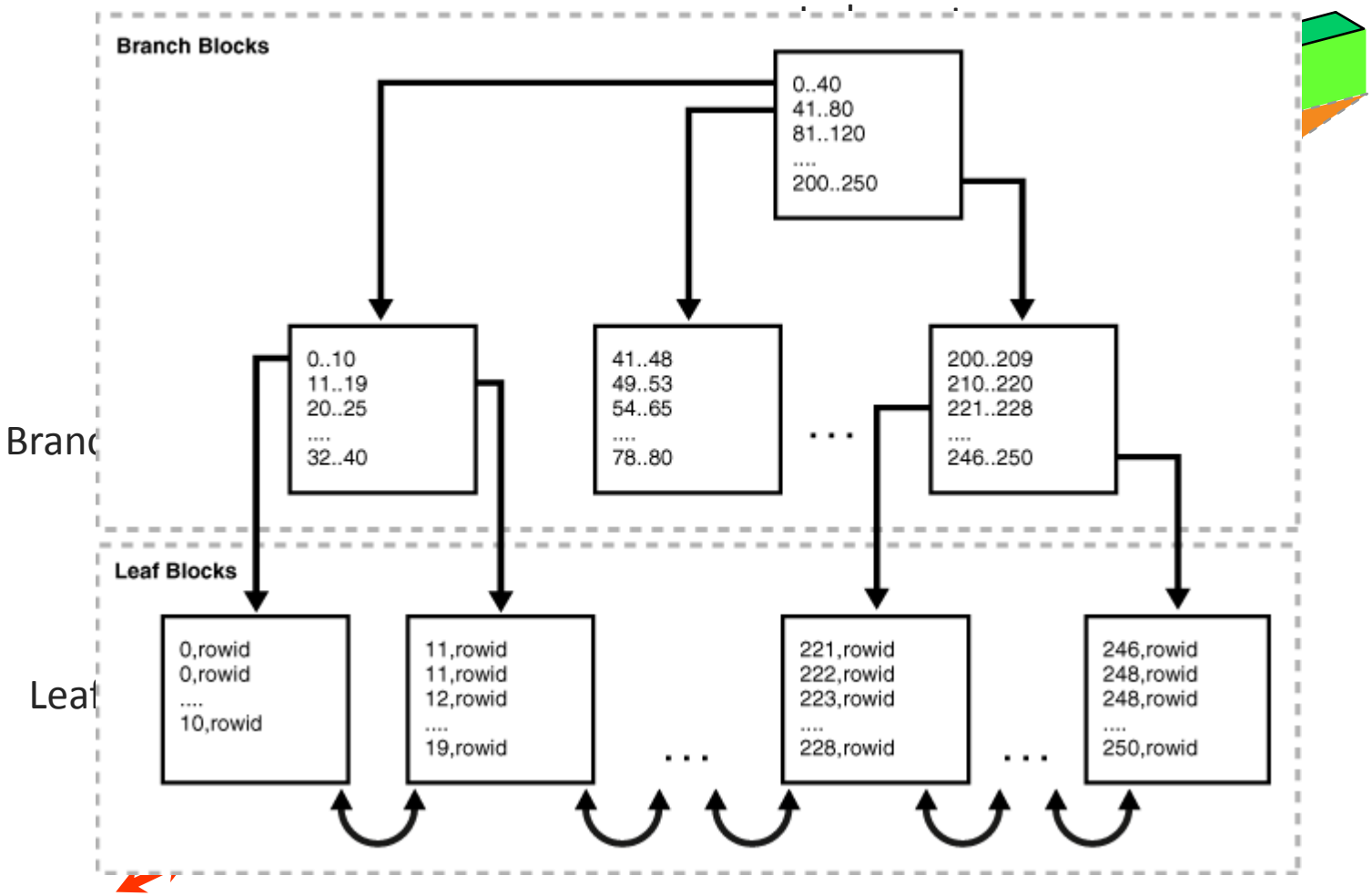
要向5号楼的702和9号楼的501发快递，怎么办？

快递员的烦恼(2)

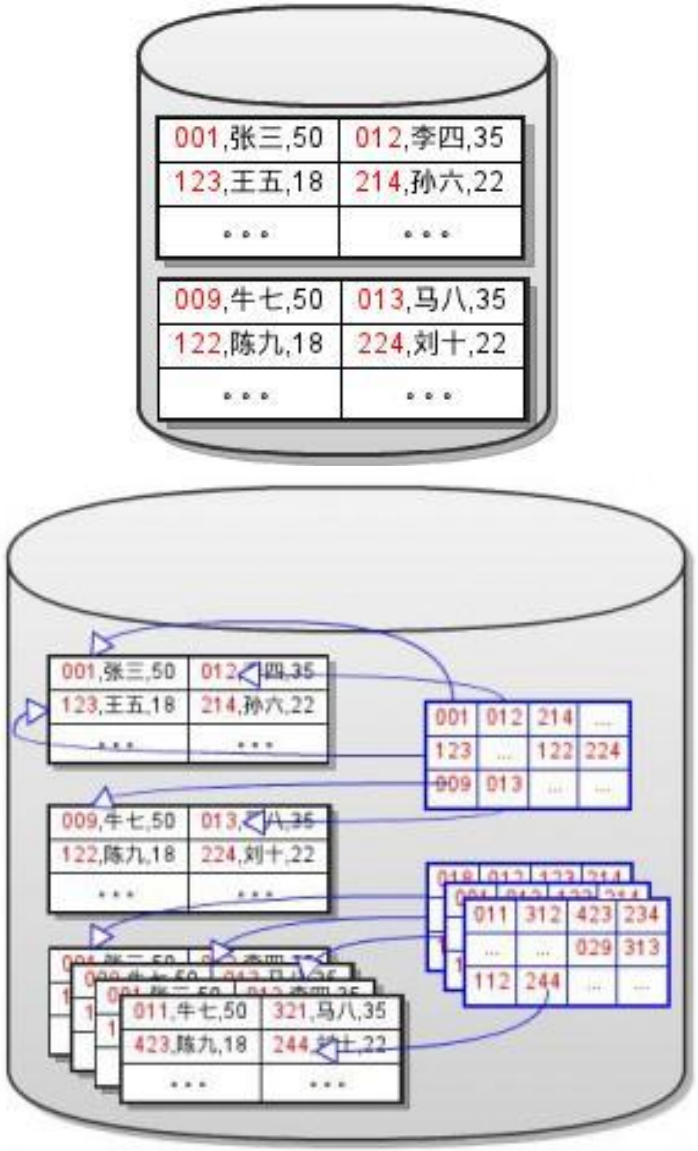
要是有个楼层图就好了！

B*Tree索引结构

为什么要设计成这样？



索引产生缘自于**快速查询数据**的需求



实际操作数据时。

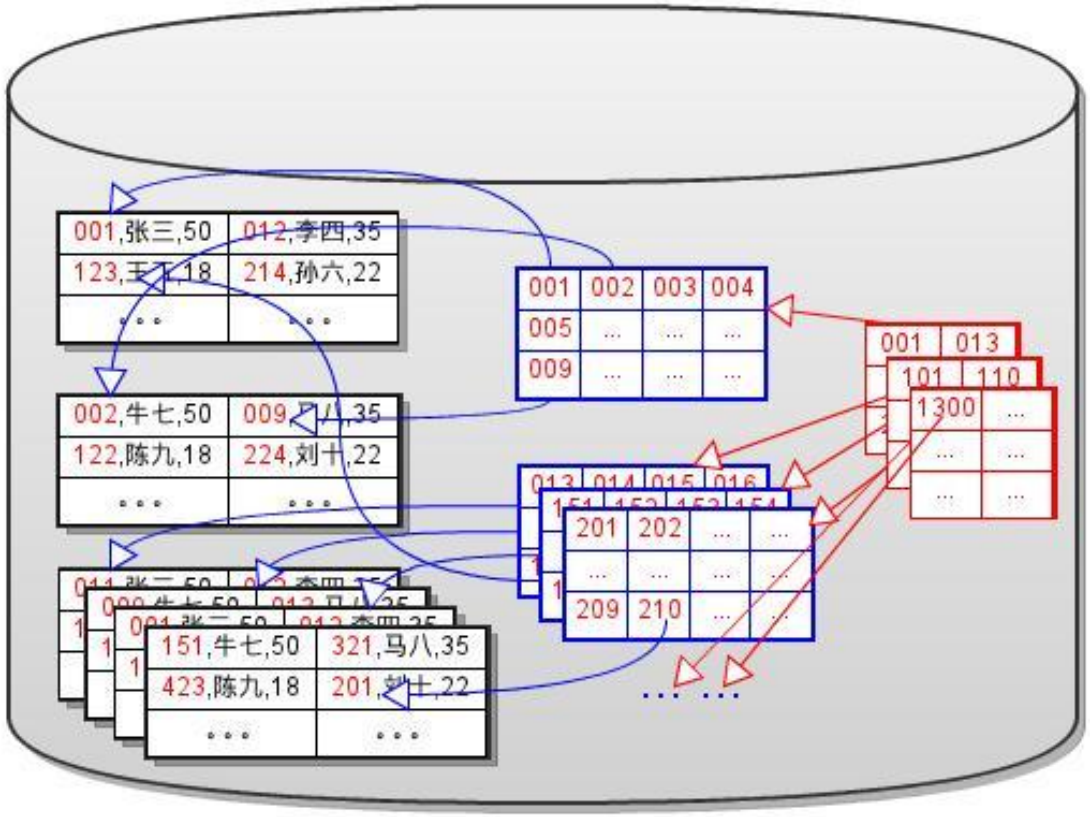
我们会发现多数情况下，并不需要访问所有数据，而只需要某部分行的某些列。

为减少 无用数据 的访问。

于是我们考虑将键值独立起来存储，每个键值再附加一个指针，指向该记录的位置。

索引的祖先 – dense index
当我们再访问数据时，就是先访问独立块进行快速的数据过渡。

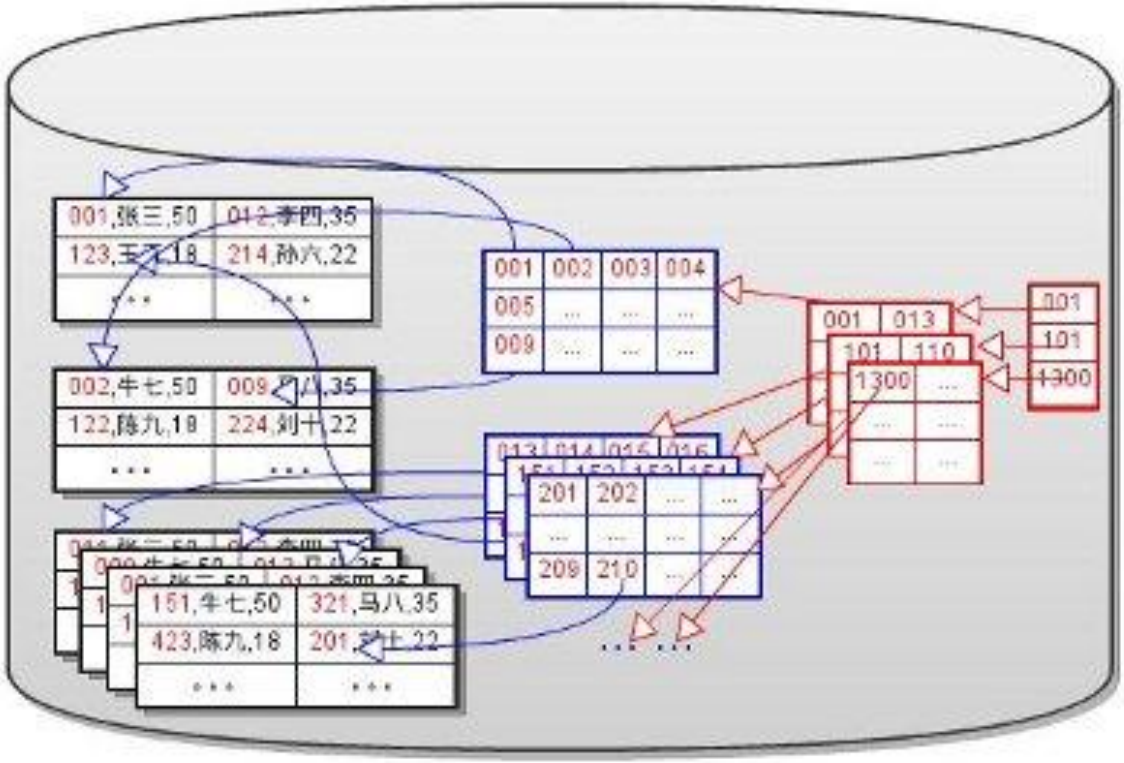
索引的进化



通过 排序 和 查找算法 减少I/O访问。

按照顺序存储索引块，因为这个索引块中不存储行的地址，因此更小访问也会更快。
这个索引就是sparse index。

索引继续进化

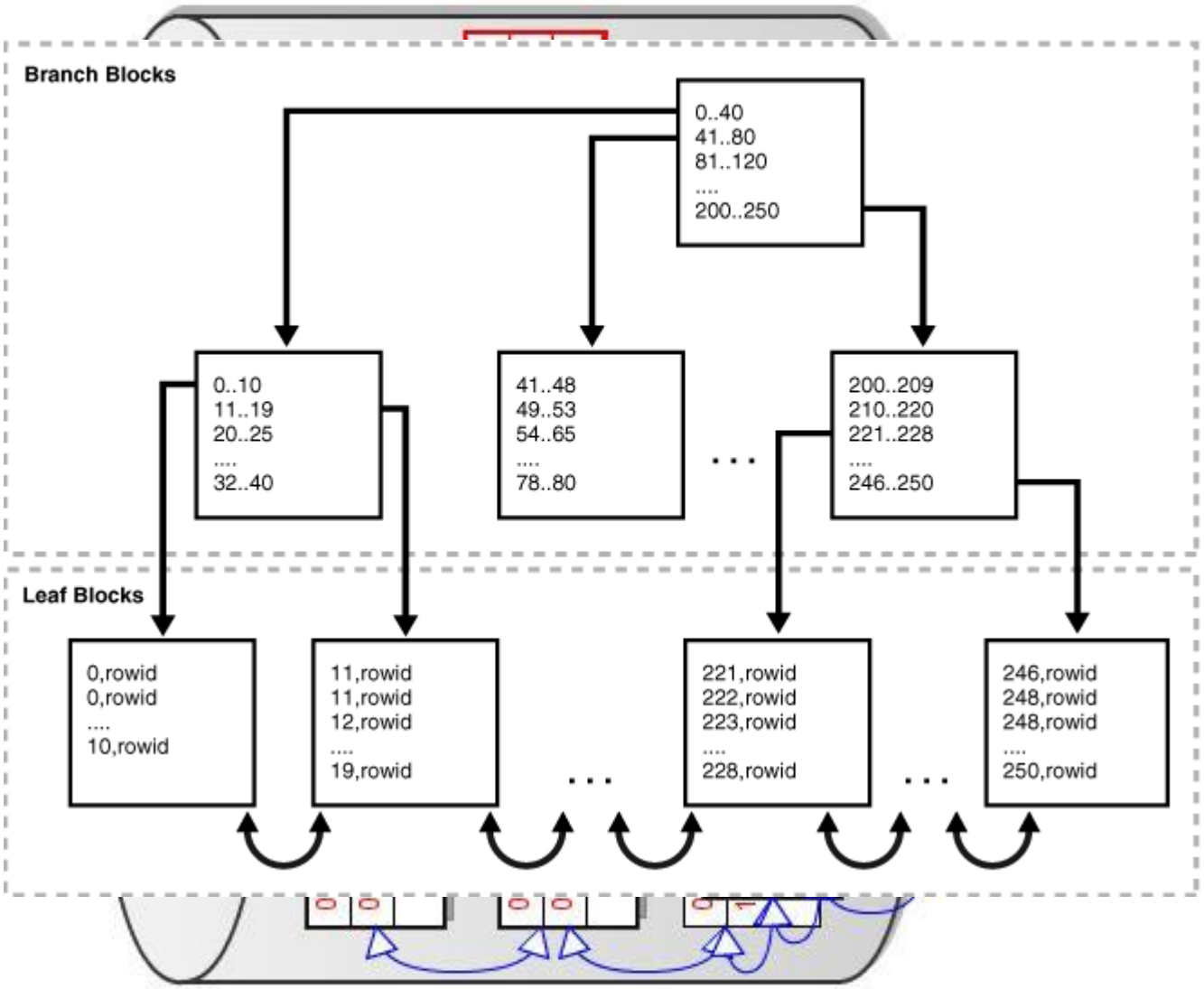


多层 **Sparse Index**。

因为Sparse Index本身是有序的，就可以为Sparse Index再创建Sparse Index，一层一层创建，直到最上层仅只需要一个块。

最上层的块，就是根(root)
最底层的块，就是叶(leaf)

通过索引**访问**数据



看着眼熟不。

换个角度再看看

对比看看

可能用到索引的操作

- 查询语句中有WHERE条件；
- 多表关联查询；
- 执行MIN()、MAX()类型聚集函数；
- 执行ORDER BY对结果集排序；
- 查询的列本身就在索引内；

索引的特点

优点：

- 提高查询速度
- 提高内存使用效率

缺点：

- 降低写效率
- 占用更多磁盘空间

什么情况下 不用索引 效果更佳?

索引要点

索引不是越多越好

» 表中必须创建 主键，索引键长度应尽可能的短小

» “WHERE”用不到的列不建索引

» 非特殊条件、STATUS/SEX等低相异值列不建索引

» “NULL”列不建索引

» 查询条件有多列时尽可能创建复合索引

这只是开始...

细节并未完全呈现!

与性能密切 **相关** 的话题

- 执行计划
- 等待事件
- 对象结构
- 软硬件环境
- 系统配置
- 业务需求
- ……

Q&A