南開大學
Nankai University

优化查询速度——数据库索引

数据库系统上机

计算机学院&网络空间安全学院 乜鹏

https://dbis.nankai.edu.cn/2019/0417/c12139a128118/page.htm

声明：上机课程的内容偏向举例，通俗化，一些术语的准确定义请查看理论课程。

01

数据库索引介绍

投票 最多可选3项

南开大学 Nankai University

你知道查询速度慢的原因有哪些么？

A 服务器**CPU/**内存资源不足

B 查询列没有建立索引或未命中索引

C 查询数据量过大

D 查询过程中出现锁或死锁

投票 最多可选3项

南开大学 Nankai University

你认为哪些数据结构可以用在索引中?

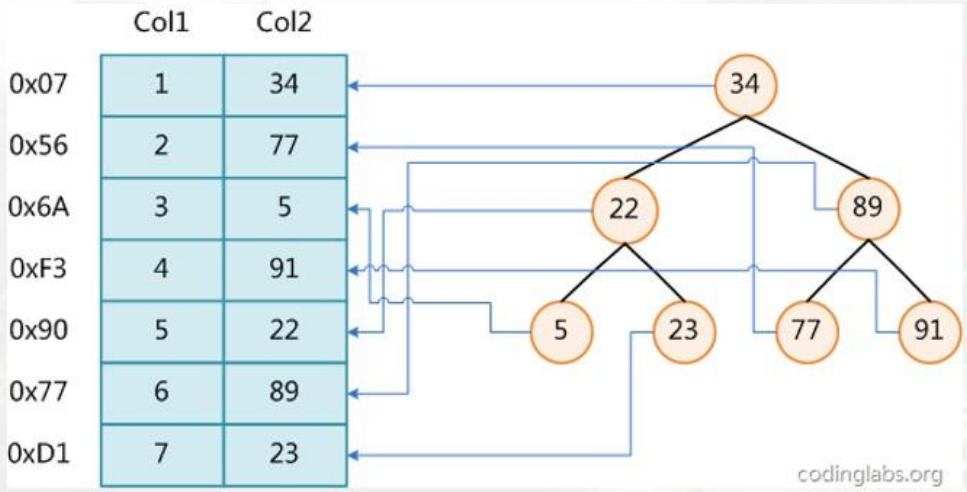
A **B树 / B+树**

B 哈希表

C 有向无环图

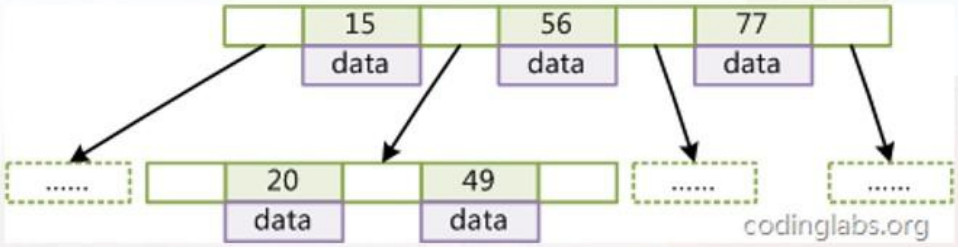
D 优先队列

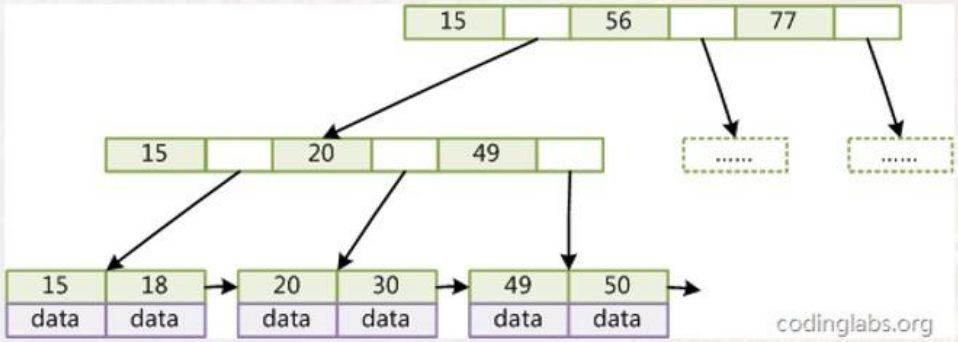
索引概述



当我们针对 Col2 查询时，
可以维护一个 BST，每个节点分别包含索引键值和一个指向对应数据记录物理地址的指针。
查询优化：O(n) -> O(log n)

B树与B+树





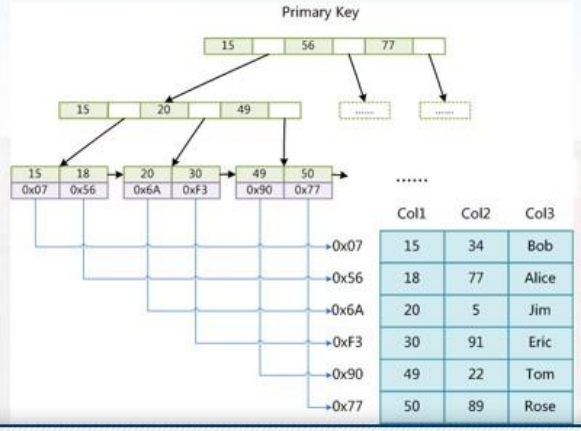
M 阶 B 树的特点：

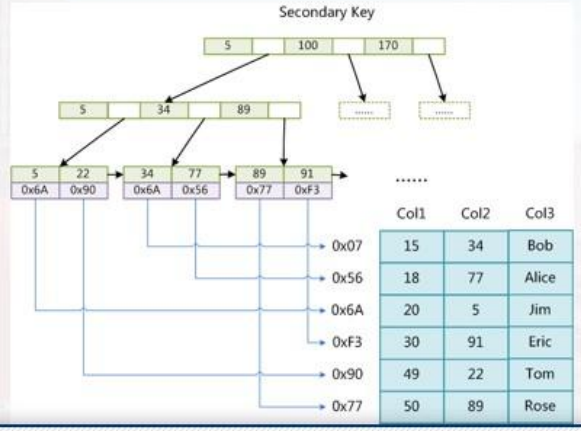
1) 每个结点至多有M个孩子；
2) 除根结点和叶结点外，其它每个结点至少有M/2个孩子；
3) 根结点至少有两个孩子（除非该树仅包含一个结点）；
4) 所有叶结点在同一层；
5) 有K个关键字的非叶结点恰好包含K+1个孩子；

B+ 树的一些区别：

1) 非叶子结点的子树指针个数与关键字（节点中的元素个数）个数相同
2) 所有关键字都在叶子结点出现
3) 只有叶子节点有Data域

索引实现——MyISAM索引实现





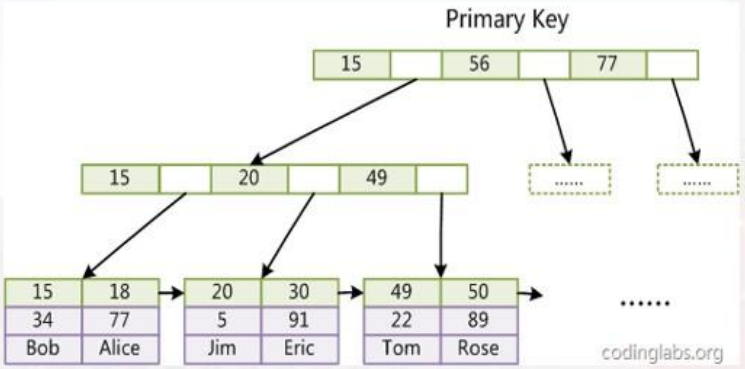
MyISAM引擎使用B+Tree作为索引结构，叶节点的data域存放的是数据记录的地址。

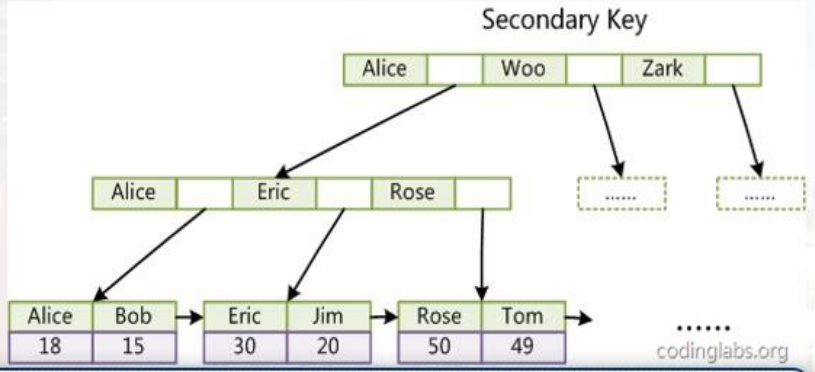
这种索引称为非聚集索引，表数据存储顺序与索引顺序无关。

在MyISAM中，主索引和辅助索引（Secondary key）在结构上没有任何区别，只是主索引要求key是唯一的，而辅助索引的key可以重复。

MyISAM索引文件和数据文件是分离的。

索引实现——InnoDB索引实现





InnoDB 的数据文件本身就是索引文件。

表数据文件本身就是按B+Tree组织的一个索引结构，这棵树的叶节点data域保存了完整的数据记录。这种索引叫做聚集索引。因为InnoDB的数据文件本身要按主键聚集，所以InnoDB要求表必须有主键（没有显示指定会默认分配到可以唯一标识的列，不存在会建立隐含字段）。

InnoDB的辅助索引data域存储相应记录主键的值而不是地址，辅助索引搜索需要检索两遍索引：首先检索辅助索引获得主键，然后用主键到主索引中检索获得记录。

索引种类

南开大学 Nankai University

主键索引	• 要求关键字不能重复，也不能为NULL；同时增加主键约束。
唯一索引	• 要求关键字不能重复；同时增加唯一约束。
普通索引	• 对关键字没有要求。
全文索引	• 关键字的来源不是所有字段的数据，而是从字段中提取的特别关键词。

注：关键字可以是某个字段，也可以是多个字段，多个字段的索引称之为复合索引。

投票 最多可选3项

南開大学 Nankai University

MySQL中唯一索引的字段内容可以为NULL吗?

A 不可以

B 可以，但是只能有一个

C 可以，且允许有多个**NULL**同时存在

02

索引如何建立与删除

创建表时添加

南開大學 Nankai University

```
59 create table student (
60   stu_id int unsigned not null auto_increment,
61   xing varchar(8) not null default '',
62   ming varchar(32) not null default '',
63   stu_sn char(10) not null default '',
64   stu_desc text,
65   primary key (`stu_id`), -- 主索引
66   unique index `ui` (`stu_sn`), -- 唯一索引
67   index `xingming` (`xing`, `ming`), -- 复合，普通索引
68   fulltext index `desc` (`stu_desc`) -- 全文索引
69 ) engine=myisam charset=utf8;
```

索引可以起名字，但是主键索引不能起名字，因为一个表仅仅可以有一个主索引，其他索引可以出现多个。名字可以省略，mysql会默认生成，通常使用字段名来充当。

更新表时添加

南開大學 Nankai University

```
create table student_2 (
  stu_id int unsigned not null,
  xing varchar(8) not null default '',
  ming varchar(32) not null default '',
  stu_sn char(10) not null default '',
  stu_desc text
) engine=myisam charset=utf8;

alter table student_2
  add primary key (`stu_id`), -- 主索引
  add unique index `ui` (`stu_sn`), -- 唯一索引
  add index `xingming` (`xing`, `ming`), -- 复合，普通索引
  add fulltext index `desc` (`stu_desc`); -- 全文索引
```

如果表中存在数据，数据符合唯一或主键的约束才可能创建成功。auto_increment属性，依赖于一个KEY。

删除索引

南開大學 Nankai University

```
90  alter table student_2
91    drop primary key,
92    drop index `ui`,
93    drop index `xingming`,
94    drop index `desc`;
```

使用sql语句的方式删除索引，auto_increment依赖于KEY，删除主键后，auto_increment也会丢失

Explain 执行计划

在select语句前使用 explain，来获取该查询语句的执行计划，而不是真正执行该语句。

```
mysql> explain select * f
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: emp
         type: ref
possible_keys: empno
          key: empno
      key_len: 4
          ref: const
         rows: 1
        Extra:
1 row in set (0.07 sec)
```

```
mysql> explain select * from emp where empno=1234567\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: emp
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1800000
        Extra: Using where
1 row in set (0.00 sec)

mysql>
```

All，全表扫描

从存储引擎中获取180W条记录才能获取查询结果

删除索引前后对比

03

索引的使用原则

投票 最多可选3项

南开大学 Nankai University

建立索引有很多好处，应该为所有字段加上索引。

A 是

B 不是

C 不清楚

投票 最多可选3项

南开大学 Nankai University

索引加快查询速度，应该为查询字段都加上索引。

A 是

B 不是

C 不清楚

索引的使用场景

南开大学 Nankai University

因为索引虽然加快了查询速度，但索引也是有代价的：
索引文件本身要消耗存储空间，同时索引会加重插入、删除和修改记录时的负担，另外，MySQL在运行时也要消耗资源维护索引，因此索引并不是越多越好。一般两种情况下不建议建索引。

1. 表记录比较少，例如一两千条甚至只有几百条记录的表，没必要建索引。一般以2000为界。
2. 索引的选择性较低。不重复的索引值（也叫基数，Cardinality）与表记录数（#T）的比值。比如性别这种属性，建立索引的意义不大。

列独立原则

南開大學 Nankai University

如果需要某个字段上使用索引，则需要在字段参与的表达中，保证字段独立在一侧。

```
select * from emp where empno=1234567;
explain select * from emp where empno=1234567\G
select * from emp where empno=1234567-1;
explain select * from emp where empno=1234567-1\G
select * from emp where empno+1=1234567;
explain select * from emp where empno+1=1234567\G
```

Empno 列已经建立主键索引，哪些查询语句可以使用索引？（不定项）

A: 第一个

B: 第二个

C: 第三个

左原则（1）

Like：匹配模式必须要左边确定不能以通配符开头。

```
-- 可以使用索引
select * from emp where ename like 'abc%';
explain select * from emp where ename like 'abc%'\G
-- 可以使用索引
select * from emp where ename like 'ab%c';
explain select * from emp where ename like 'ab%c'\G
-- 不能使用索引，使用通配符开头 %任意字符任意数量，  _任意一个字符
select * from emp where ename like '%abc';
explain select * from emp where ename like '%abc'\G
-- 不能使用索引，使用通配符开头 %任意字符任意数量，  _任意一个字符
select * from emp where ename like '_abc';
explain select * from emp where ename like '_abc'\G
-- 不能使用索引，使用通配符开头 %任意字符任意数量，  _任意一个字符
select * from emp where ename like '_abc%';
explain select * from emp where ename like '_abc%'\G
```

投票 最多可选3项

南开大学 Nankai University

已知 **title** 列已经建立索引，请问两条查询语句能否使用索引?

A 都可以

B 只有第一个可以

C 只有第二个可以

D 都不可以

```
SELECT * FROM employees.titles
WHERE left(title, 6) = 'Senior';

SELECT * FROM employees.titles
WHERE title like 'Senior%';
```

左原则（2）

南開大學 Nankai University

复合索引：一个索引关联多个字段，仅仅针对左边字段有效果。

```
145
146  alter table emp add index `multi_index` (ename, empno);
147
```

```
mysql> explain select * from emp where ename like 'abc%'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: emp
         type: range
possible_keys: multi_index
          key: multi_index
      key_len: 62
          ref: NULL
         rows: 91
        Extra: Using where
1 row in set (0.00 sec)
```

```
mysql> explain select * from emp where empno = 1234567\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: emp
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1800000
        Extra: Using where
1 row in set (0.00 sec)
```

OR的使用

南開大學 Nankai University

必须要保证 OR 两端的条件都存在可以用的索引，该查询才可以使用索引。

```
mysql> explain select * from emp where ename like 'abc%' OR empno > 1800000\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: emp
         type: index_merge
possible_keys: multi_index,empno
          key: multi_index,empno
      key_len: 62,4
          ref: NULL
         rows: 204296
        Extra: Using sort_union(multi_index,empno); Using where
1 row in set (0.07 sec)

mysql>
```

都有可用的索引

为后面的条件增加可以使用的索引

MySQL智能选择

南开大学 Nankai University

```
mysql> explain select * from emp where empno > 1800000\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: emp
         type: range
possible_keys: empno
          key: empno
      key_len: 4
          ref: NULL
         rows: 204205
        Extra: Using where
1 row in set (0.00 sec)

mysql> explain select * from emp where empno > 1600000\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: emp
         type: ALL
possible_keys: empno
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1800000
        Extra: Using where
1 row in set (0.00 sec)

mysql>
```

可以

不可以

即使满足了上面说原则，MySQL也能弃用索引。

弃用索引的主要原因：查询即使使用索引，会导致出现大量的随机IO，相对于从数据记录的第一条遍历到最后一条的顺序IO开销，还要大。

小结

南開大學 Nankai University

1. 不要过度索引。索引越多，占用空间越大，反而性能变慢；
2. 只对WHERE子句中频繁使用的建立索引；
3. 尽可能使用唯一索引，重复值越少，索引效果越强；
4. 使用短索引，如果char(255)太大，应该给它指定一个前缀长度，大部分情况下前10位或20位值基本是唯一的，那么就不要对整个列进行索引；
5. 充分利用左前缀，这是针对复合索引，因为WHERE语句如果有AND并列，只能识别一个索引(获取记录最少的那个)，索引需要使用复合索引，那么应该将WHERE最频繁的放置在左边。
6. 索引存在，如果没有满足使用原则，也会导致索引无效。

索引的使用场景

南开大学 Nankai University

1. 索引检索：检索数据时使用索引。

2. 索引排序：如果order by 排序需要的字段上存在索引，则可能使用到索引。

3. 索引覆盖：复合索引拥有的关键字内容，覆盖了查询所需要的全部数据，此时，就不需要在数据区获取数据，仅仅在索引区即可。覆盖就是直接在索引区获取内容，而不需要在数据区获取。

全文索引

南開大學 Nankai University

1. Mysql 的全文索引不支持中文，所以在实际中几乎不使用。
2. 全文索引推荐使用 Lucene 或 ElasticSearch。
3. 全文索引与信息检索密切相关，感兴趣的同学可以选修相关课程《信息检索》。

轮到你了!