数据库

2017年8月14日
8:29

一、索引

1.1 索引的概念

索引是存储引擎用于快速找到记录的一种数据结构。

1.2 索引分类

①. B-Tree索引：使用B-Tree数据结构来存储数据。

1) InnoDB存储引擎的B-Tree索引的实现方式

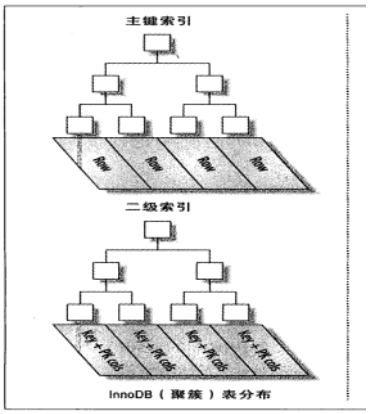
a) InnoDB采用的是B+Tree的数据结构来存储数据的。

b) InnoDB根据主键引用被索引的行

c) 索引的实现

二级索引的叶子结点包含了索引列的值和主键值，主键索引的叶子结点存储了行数据。

查找规则：二级索引表进行查找找到对应的主键值，在主键索引中通过主键值查找到行数据。

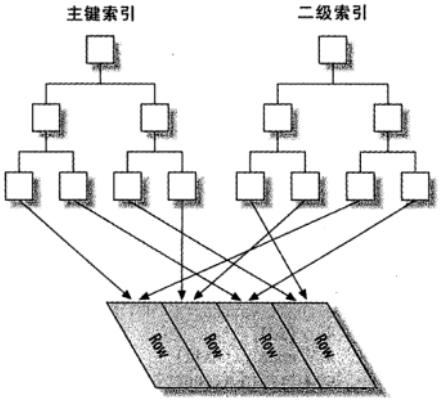


2) MyISAM存储引擎的B-Tree索引的实现方式

a) MyISAM索引通过数据的物理位置引用被索引的行

b) 索引的实现方式

主键索引和二级索引没有区别，都属于非聚簇索引。叶子结点存储了行数据的物理位置。



3) 优点

a) B-Tree对索引列是顺序组织存储的，所以很适合查找范围数据

b) B-Tree索引适用于全键值、键值范围或键前缀查找。其中键前查找只适用于根据最左前缀的查找。

②. 哈希索引

哈希索引基于哈希表实现的，只有精确匹配索引所有列的查询才有效。对所有的索引列计算一个哈希码，哈希码是一个较小的值，哈希索引将所有的哈希码存储在索引中，同时在哈希表中指向每个数据的指针。

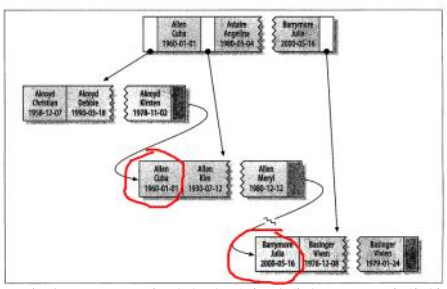
只支持等值比较查询，不支持范围查询。（因为不是有序的存储）

1.3 索引使用规则

前提情景：

```
CREATE TABLE People (
   last_name  varchar(50)    not null,
   first_name varchar(50)    not null,
   dob        date           not null,
   gender     enum('m', 'f')not null,
   key(last_name, first_name, dob)
);
```

索引为（last_name,first_name,dob）



1、全值匹配：和索引中的所有列进行匹配。查找姓名为Cuba Allen,出生于1960-01-01的人。

2、匹配最左前缀：查找所有姓为Allen的人。只能使用索引的第一列。

3、匹配列前缀：某一列的的值的开头部分。只能索引的第一列。查找所有姓以J开头的人。
4、匹配范围值：只能使用索引的第一列。查找姓为Allen和Barrymore之间的人。
5、精确匹配某一列并范围匹配另外一列。查找所有姓为Allen，并且名字是字母K开头的人。

1.3 使用规则总结

索引对多个值进行排序的依据是定义索引时列的顺序。

①. 如果不是按照索引的最左列开始查找，则无法使用索引。

②. 不能跳过索引中的列。

③. 如果查询中有某个列是范围查询，则其右边所有列都无法使用索引优化查找。

二、高性能的索引策略

B-Tree索引，按照顺序存储数据，所以MySQL可以用来做order by和group by操作。

2.1 独立的列

索引列不能是表达式的一部分，也不能是函数的参数。

错误的做法：

```
mysql> SELECT actor_id FROM sakila.actor WHERE actor_id + 1 = 5;

mysql> SELECT ... WHERE TO_DAYS(CURRENT_DATE) - TO_DAYS(date_col) <= 10;
```

正确的做法：

```
select actor_id from sakila.actor where actor_id = 4;
```

2.2 前缀索引

BLOB,TEXT的列，必须使用前缀索引。这样需要考虑索引的选择性。选择性越高，过滤的数据行数越高。

2.5 聚簇索引

数据存储方式。

①. 数据行实际上存放在索引的叶子页中。聚簇表示数据行和相邻的键值紧凑的存储在一起。因此一个表只有一个聚簇索引。

②. 只有表包含聚集索引时，表中的数据才按顺序存储，如果表没有聚集索引，则其数据行存储在一个成为堆的无序结构中。

③. InnoDB通过主键聚集数据。InnoDB支持聚簇索引，MyISAM不支持聚簇索引。

存在问题：

数据插入可能导致页分裂问题。占用更多的磁盘空间。

二级索引访问需要两次索引查找，而不是一次。

2.6 非聚集索引

非聚集索引并不是物理上的排列数据，即索引中的逻辑顺序并不等同于表中行的物理顺序，索引是指向表中行的位置的指针，这些指针本身是有序的，通过这些指针可以在表中快速定位的数据。

如果一张表包含一个非聚集索引但没有聚集索引，则新的数据将被插入到最末一个数据页中，然后非聚集索引将被更新。如果也包含聚集索引，该聚集索引将被用于查找新行将要处于什么位置，随后，聚集索引、以及非聚集索引将被更新。

2.7 索引的理解

聚簇索引和非聚簇索引只是数据存储的一种方式。对于InnoDB和MyISAM存储引擎来说，InnoDB支持聚簇和非聚簇两种存储结构；MyISAM只支持非聚簇索引。（书中提到的二级索引就是非聚簇索引）两种存储引擎实现两种存储结构的方式是不同的。

InnoDB的二级索引的优点，减少了当出现行移动或者数据页分裂时候二级索引的维护工作。

只有当索引的列顺序和order by子句的顺序完全一致，并且所有列的排序方向都一样时，mysql才能够使用索引来对结果进行排序。

三、数据库锁

3.1 共享锁

共享锁（S锁）表示对数据进行读操作。因此多个事务可以同时为一个对象加共享锁。

sql : select * from ad_plan lock in share mode;

3.2 排它锁

排它锁也叫写锁（X）。对数据进行写操作。

sql : select * from ad_plan for update;

对于锁，通常会用一个矩阵来描述他们之间的冲突关系。

	S	X
S	+	-
X	-	-

+代表兼容，-代表不兼容

sql : select * from information_schema.innodb_locks：可以查看锁。

MySQL有三种锁的级别：页级，表级，行级。

MyISAM：表级锁。分为共享锁和排他锁。

InnoDB：行级锁和表级锁。行级锁是加在索引数据项上的，如果不使用索引进行查询会加表级锁。

三、面试题

1、数据库引擎，InnoDB和MYISAM的区别？

㈠. InnoDB：支持事务处理。支持行级锁。支持外键。不支持全文索引。不保存表的行数，因此count()需要进行一次全表扫描。

㈡. MyISAM：不支持事务处理。表级锁。不支持外键。支持全文索引。保存表的行数，count()只需要读取相应的字段值就可以了。

2、索引实现方式？B+树有什么特点？二叉树，B树和B+树的区别？B+树的实现方式？为什么不用红黑树?为什么使用B+实现？

㈠. 索引实现方式：B+树。

㈡. B+树：多路搜索树。每个结点的数据是有序的，每个结点的指针指向的数据时大于这个数据的结点必须小于相邻右侧节点的值。叶子结点存储了所有的数据信息，且是有序的。

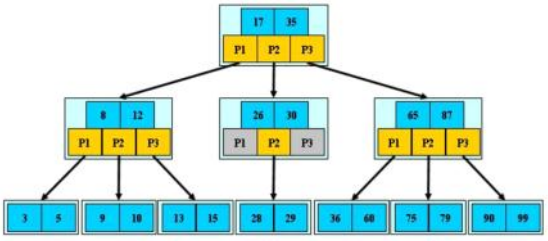
㈢. 二叉树，B树和B+树的区别？

①. 二叉搜索树：所有非叶子结点最多有两个儿子；每个结点存储一个关键字；非叶子结点的左指针指向小于该关键字的子树，右指针指向大于其关键字的子树。

②. B-树：一种多路搜索树

1) 所有非叶子结点中包含n个关键字，且是有序排列，因此查找关键字，不必根据叶子结点来定位。

2) 从根结点开始，对结点内的关键字序列进行二分查找，如果命中就结束，否则对儿子结点进行查找。



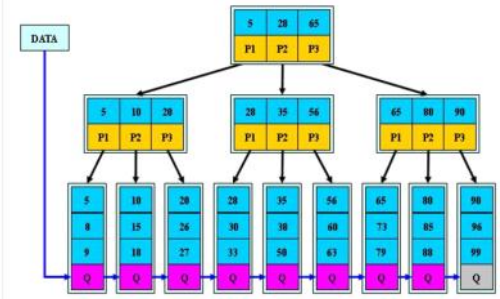
③. B+树：B-树的一种变体，也是一种多路搜索树。

1) 所有关键字都出现在叶子结点的链表中，且链表中的关键字是有序的；不可能在非叶子结点命中

2) 非叶子结点相当于叶子结点的索引，叶子结点相当于存储数据的数据层。

3) 内部结点并没有指向关键字具体信息的指针，因此内部结点相对B-Tree更小。因此磁盘容纳的关键字数量越多，一次性读入内存中的需要查找的关键字也就越多，IO读写次数降低。

4) 任何关键字的查找必须走一条从根结点到叶子结点的路。所有关键字的路径长度相同，导致每个数据的查找效率相当。



㈣. 为什么不用红黑树？为什么使用B+树实现呢？

①. 红黑树是二叉查找树结果，查找的时间复杂度与树的深度有关。降低树的深度自然对查找效率有所提高。二叉查找树的深度过大而造成磁盘I/O读写过于频繁，进行导致查询效率低下，因此采用多路搜索树可以降低树的深度，提高 查询效率。

3、索引的最左匹配原则？

索引对多个值进行排序的依据是定义索引时列的顺序。因此进行查找时，必须从索引的最左列开始，否则不能使用索引；不能跳过索引列；如果查询列为范围查找，则右边的索引列不能使用。

4、事务特性，隔离界别，mysql的默认隔离级别？

㈠. 事务是数据库中一个单独的执行单元。用户定义的一个数据库的操作序列。

㈡. 事务的四大特性分别是原子性，一致性，隔离性，持久性。

i. 原子性是指事务中的操作时不可分割的，要么全部执行，要么都不执行。

ii. 一致性是指几个并发执行的事务，其结果必须与某一顺序执行的结果一致。

iii. 隔离性：事务之间是封闭的。事务执行的中间结果必须对其他事务时透明地。

iv. 持久性：事务一旦提交，对数据库的影响是永久的。

㈢. 事务的隔离级别有四种：不提交的读，提交的读，可重复的读，串行化。

i. 不提交的读：事务中的修改，即使没有提交，对其他事务也是可见的。导致脏读。

ii. 提交的读：一个事务从开始直到提交之前，所做的任何修改对其他事务都是不可见的。

iii. 可重复的读：同一个事务中多次读取同样的记录的结果是一致的。产生幻读问题：当某个事务在读取某个范围内的记录时，另外一个事务又在该范围内插入了新的记录，当之前的事务再次读取某个范围内的记录时，产生幻读。

iv. 串行化：事务串行执行。

㈣. MYSQL默认的事务隔离级别是：可重复读。

5、如何确定数据库中自己建立的索引被执行了呢？

explain语句查看具体的执行过程。

6、聚簇索引和非聚簇索引的区别？

㈠. 聚簇索引：该索引中的键值的逻辑顺序决定了表中相应行的物理顺序。

㈡. 非聚簇索引：该索引中索引的逻辑顺序与磁盘上行的物理存储顺序不同。

7、什么是视图？

视图是从数据库的基本表中选取出来的数据组成的逻辑窗口。与基本表不同，它是一个虚表。数据库中只存放视图的定义，而不存放视图包含的数据项。

8、数据库范式

第一范式：每一列都是不可分割的基本数据项。

第二范式：每一个非主属性完全函数依赖于候选键。

第三范式：每一个非主属性都不传递依赖于候选键

第四范式：每一个主属性都不传递依赖于候选键。

9. 你对mysql有什么了解

10. 说一下数据库事务的四个特性，为什么mysql事务能保证失败回滚

11. mysql数据库的锁有多少种，怎么编写加锁的sql语句

12. mysql什么情况下会触发表锁

13. 页锁、乐观锁、悲观锁

项目中如何实现事务？

https://www.nowcoder.com/discuss/26180