上课内容：

1.回顾为什么MySQL底层使用b+树？

2.linux服务器中查询MySQL索引文件

3.MySQLb+树能够存放多少字节数据

4.索引为什么会失效？注意那些事项？

5.联合索引为什么需要遵循左前缀原则？

6.分表分库为什么能够提高数据库查询效率？

7.MySQL是如何定位慢查询

# 回顾为什么MySQL底层使用b+树？

B+树索引具有范围查找和前缀查找的能力，相当于二分查找。

Hash索引只能支持等于查询，无法支持范围查询

# MySQLb+树能够存放多少字节数据

由于存储介质的特性，磁盘本身存取就比主存慢很多，再加上机械运动耗费，磁盘的存取速度往往是主存的几百分分之一，因此为了提高效率，要尽量减少磁 盘I/O。为了达到这个目的，磁盘往往不是严格按需读取，而是每次都会预读，即使只需要一个字节，磁盘也会从这个位置开始，顺序向后读取一定长度的数据放 入内存。这样做的理论依据是计算机科学中著名的局部性原理：

当一个数据被用到时，其附近的数据也通常会马上被使用。

程序运行期间所需要的数据通常比较集中。

由于磁盘顺序读取的效率很高（不需要寻道时间，只需很少的旋转时间），因此对于具有局部性的程序来说，预读可以提高I/O效率。

预读的长度一般为页（page）的整倍数。页是计算机管理存储器的逻辑块，硬件及操作系统往往将主存和磁盘存储区分割为连续的大小相等的块，每个存 储块称为一页（在许多操作系统中，页得大小通常为4k），主存和磁盘以页为单位交换数据。当程序要读取的数据不在主存中时，会触发一个缺页异常，此时系统 会向磁盘发出读盘信号，磁盘会找到数据的起始位置并向后连续读取一页或几页载入内存中，然后异常返回，程序继续运行。

一般操作系统一页大小为4kb；getconf PAGE\_SIZE

局部性原理与磁盘预读  
计算机科学中著名的局部性原理当一个数据被用到时， 其附近的数据也通常会马上被使用。  
为了提高效率，磁盘往往不是严格按需读取，而是每次都会预读，即使只需要一个字节，磁盘也会从这个位置开始，顺序向后读取一定长度的数据放入内存。  
这里的-定长度叫做页，也就是操作系统操作磁盘时的基本单位。一般操作系统中.页的大小是4Kb (getconf  PAGE\_SIZE)  
总结  
如果一个B树或B+树中的节点的大小

假设一个节点为1页

如果从磁盘读取超过1页大小，根据局部性原理与磁盘预读 会读出2页大小  
如果从磁盘读取小于1页大小，根据局部性原理与磁盘预读 会读出1页大小

根据以上规则，如果读取整哈是页的倍数，这样就可以不用浪费，所以B+树的每一个节点是页的倍数是最佳的。

在MySQL中我们的InnoDB页的大小默认是16k，当然也可以通过参数设置：

show variables like 'innodb\_page\_size';

16384/1024=16kb;

MySQLb+树能够存放多少字节数据

假设一行为1kb，那么一页可以读取16行数据，一个叶子节点可以存放16条数据

那么非叶子节点存放多少条数据？非叶子节点存放索引值(bigint 8b)和指针(6b)

那么一页16\*1024/(8+6)=1170指针

B+树 高度为2 1170\*16=18720 条数据

B+树 高度为3 1170\*1170\*16=21902400 条数据

所以在InnoDB中B+树高度一般为1-3层，它就能满足千万级的数据存储。在查找数据时一次页的查找代表一次IO，所以通过主键索引查询通常只需要1-3次IO操作即可查找到数据。

# 索引为什么会失效？注意那些事项？

Explain命令在解决数据库性能上是第一推荐使用命令，大部分的性能问题可以通过此命令来简单的解决，Explain可以用来查看SQL语句的执行效 果，可以帮助选择更好的索引和优化查询语句，写出更好的优化语句。

|  |
| --- |
| CREATE TABLE `user\_details` (  `id` int(11) DEFAULT NULL,  `user\_name` varchar(50) DEFAULT NULL,  `user\_phone` varchar(11) DEFAULT NULL,  PRIMARY KEY (id)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  ALTER TABLE `user\_details` ADD INDEX user\_name\_index ( `user\_name` )  insert into user\_details values(1,'yushengjun1','15921009245');  insert into user\_details values(2,'yushengjun2','15921009245');  insert into user\_details values(3,'yushengjun3','15921009245');  insert into user\_details values(4,'yushengjun4','15921009245');  insert into user\_details values(5,'yushengjun5','15921009245');  insert into user\_details values(6,'yushengjun6','15921009245');  EXPLAIN select \* from user\_details WHERE id=1  EXPLAIN select \* from user\_details WHERE id=1 and user\_name='yushengjun1';  EXPLAIN select \* from user\_details WHERE id like '%sss'  EXPLAIN select \* from user\_details WHERE id like '%1'  EXPLAIN select \* from user\_details WHERE user\_name like '1%'  EXPLAIN select \* from user\_details WHERE user\_name =1;  EXPLAIN select \* from user\_details WHERE user\_name ='1'; |

1. 索引无法存储null值
2. 前导模糊查询不能利用索引(like '%XX'或者like '%XX%')
3. 索引失效的几种情况

1.如果条件中有or，即使其中有条件带索引也不会使用(这也是为什么尽量少用or的原因)

要想使用or，又想让索引生效，只能将or条件中的每个列都加上索引

2.对于多列索引，不是使用的第一部分，则不会使用索引

3.like查询以%开头

4.如果列类型是字符串，那一定要在条件中将数据使用引号引用起来,否则不使用索引

5.如果mysql估计使用全表扫描要比使用索引快,则不使用索引

# 联合索引为什么需要遵循左前缀原则？

对于多列索引，不是使用的第一部分，则不会使用索引

|  |
| --- |
| CREATE TABLE `user\_details` (  `id` int(11) DEFAULT NULL,  `user\_name` varchar(50) DEFAULT NULL,  `user\_phone` varchar(11) DEFAULT NULL,  PRIMARY KEY (id,user\_name)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  insert into user\_details values(1,'yushengjun1','15921009245');  insert into user\_details values(1,'yushengjun2','15921009245');  insert into user\_details values(2,'yushengjun1','15921009245');  insert into user\_details values(2,'yushengjun2','15921009245');  insert into user\_details values(3,'yushengjun1','15921009245');  insert into user\_details values(3,'yushengjun2','15921009245');  EXPLAIN select \* from user\_details WHERE id=1  EXPLAIN select \* from user\_details WHERE id=1 and user\_name='yushengjun1';  EXPLAIN select \* from user\_details WHERE user\_name='yushengjun1';  EXPLAIN select \* from user\_details WHERE user\_name='yushengjun1' and id=1  (1,yushengjun1 1,yushengjun2),(2,yushengjun1 2,yushengjun2),(3,yushengjun1 3,yushengjun2) |

# 分表分库为什么能够提高数据库查询效率？

分表分库为什么提高查询的效率？因为会将一张表的数据拆分成多个n张表进行存放，让后在使用第三方中间件（MyCat或者Sharding-JDBC）并行同时查询，让后在交给第三方中间进行组合返回给客户端。

# MySQL是如何定位慢查询