

# 课前热场

## 自我介绍

老猫。一个在IT行业奋战十几年的老兵，曾就职于摩托罗拉物流研发中心任系统架构师，软通动力任系统架构师、高级企业讲师。也曾自主创业，开发过互联网金融和在线教育等产品。在电子商务、电信、企业管理、政务管理等行业有较丰富的经验，擅长设计和开发高并发、高可用、海量数据的分布式系统。

## 大厂面试问题

存储引擎的InnoDB与MyISAM的区别，优缺点，使用场景？

说说 MySQL 优化之道？

UndoLog和RedoLog的区别和联系？

MySQL索引的数据结构是什么，及为什么使用这种数据结构？

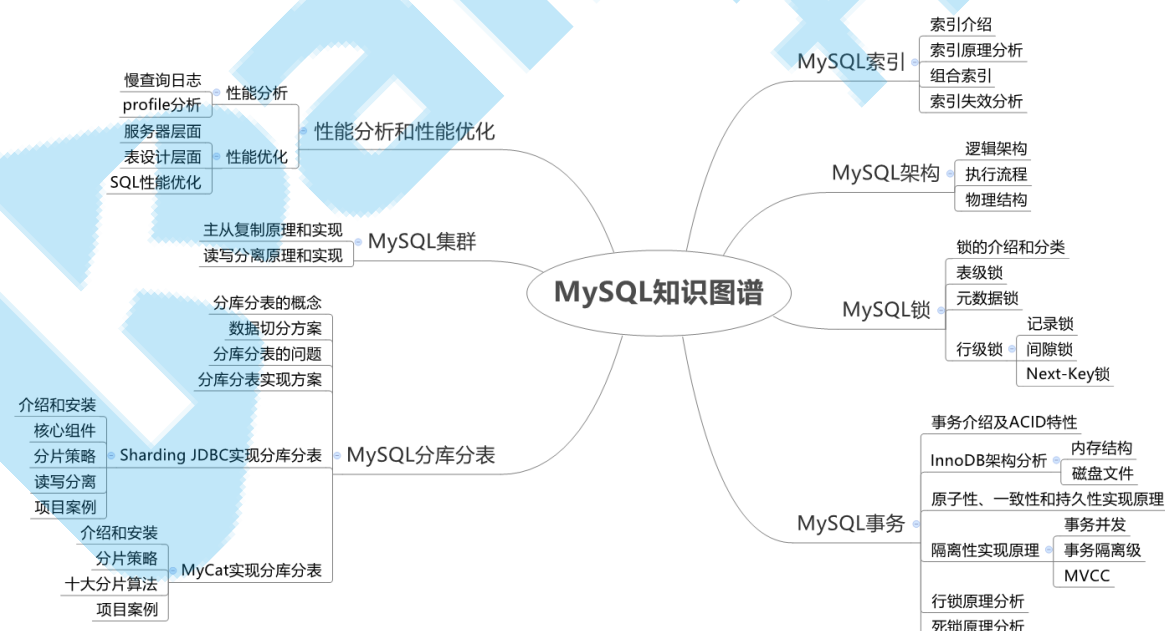
索引失效的场景有哪些？

什么是死锁和死锁的排查和解决？

RC和RR的实现原理及区别和使用场景？

分库与分表带来的分布式困境与应对之策？

.....



## 课堂主题

Mysql架构、索引介绍及原理

## 课堂目标

掌握Mysql的各组件及各组件的功能

理解Mysql简版执行流程和详细执行流程

掌握Mylsam和InnoDB的区别并说明使用场景

掌握Mysql日志文件及主要日志文件的作用

理解Mysql的数据文件及作用

使用命令查看mysql日志

配置my.cnf开启二进制日志、通用查询日志、慢查询日志等

掌握索引、分类、优劣势

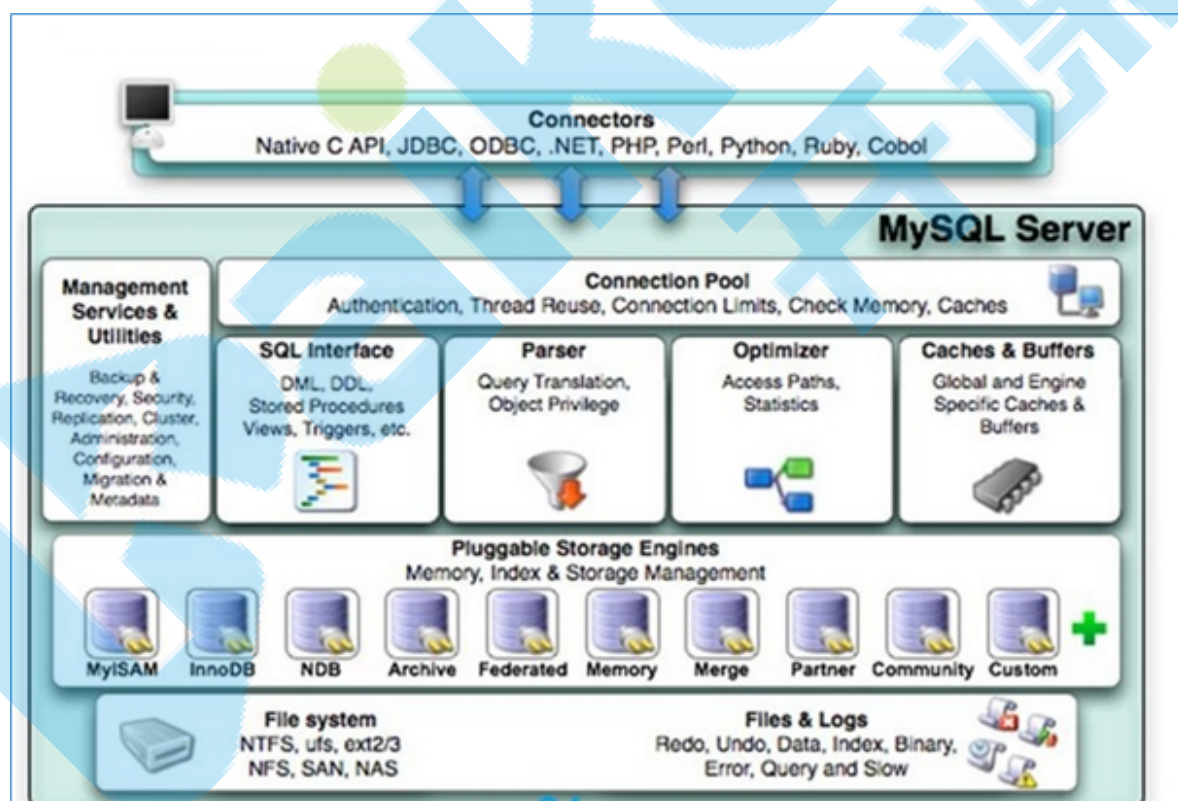
使用命令创建、查看、删除索引

理解索引的原理和存储结构

## 一、MySQL架构篇

### 逻辑架构

#### 逻辑架构图



连接器 (Connectors)

系统管理和控制工具 (Management Services & Utilities)

连接池 (Connection Pool)

SQL Layer MySQL业务层

SQL接口 (SQL Interface)

接收SQL DML DDL

## 解析器 (Parser)

select \* from t1

词法分析 分词 ----- 》形成语法树

语法分析 分析：符合SQL的语法 SQL的语法：SQL 92 limit MYSQL自己的语法

elect \* from t1 语法错误 syntnx error ..

形成正确语法树

## 查询优化器 (Optimizer)

mysql 觉得你写的SQL 不是完美的

优化什么呢?

索引 只使用一个 使用最优 explain

多表关联 小表驱动大表

where 从左到右 MySQL 找过滤力度最大的 先执行

where id=1 and sex='男'

where sex='男' and id=1

explain

从右到左 Oracle

## 查询缓存 (Cache和Buffer)

把查询结果存起来

select \* from tuser where id=1

SQL --- > hash后的值 唯一 则 表示有

Map

key value

hash(select语句) 查询结果

1、sql有变化 就值不唯一

2、数据有变化，缓存清除

MySQL8.0后不再使用

## 存储引擎 (Pluggable Storage Engines)

以表为单位

creat table xxx() engine=InnoDB/Memory/MyISAM

MySQL的存储引擎是针对表进行指定的。(engine=InnoDB\myisam)

存储引擎	说明
MyISAM	高速引擎，拥有较高的插入，查询速度， <b>但不支持事务、不支持行锁</b> 、支持3种不同的存储格式。包括静态型、动态型和压缩型。
InnoDB	<b>5.5版本后MySQL的默认数据库，支持事务和行级锁定，事务处理、回滚、崩溃修复能力和多版本并发控制的事务安全</b> ，比MyISAM处理速度稍慢、支持 <b>外键 (FOREIGN KEY)</b>
ISAM	MyISAM的前身，MySQL5.0以后不再默认安装
MRG_MyISAM (MERGE)	将多个表联合成一个表使用，在超大规模数据存储时很有用
Memory	<b>内存存储引擎，拥有极高的插入，更新和查询效率</b> 。但是会占用和数据量成正比的内存空间。只在内存上保存数据，意味着数据可能会丢失
Falcon	一种新的存储引擎，支持事物处理，传言可能是InnoDB的替代者
Archive	将数据压缩后进行存储，非常适合存储大量的独立的，作为历史记录的数据，但是只能进行插入和查询操作
CSV	CSV 存储引擎是基于 CSV 格式文件存储数据(应用于跨平台的数据交换)

xtraDB存储引擎是由Percona公司提供的存储引擎，该公司还出品了Percona Server这个产品，它是基于MySQL开源代码进行修改之后的产品。

阿里对于Percona Server服务器进行修改，衍生了自己的数据库（alisql）。

• **InnoDB和MyISAM存储引擎区别：**

	Innodb	Myisam
存储文件	.frm 表定义文件 .ibd 数据文件和索引文件	.frm 表定义文件 .myd 数据文件 .myi 索引文件
锁	表锁、行锁	表锁
事务	支持	不支持
CRDU	读、写	读多
count	扫表	专门存储的地方（加where也扫表）
索引结构	B+ Tree	B+ Tree
外键	支持	不支持

存储引擎的选型：

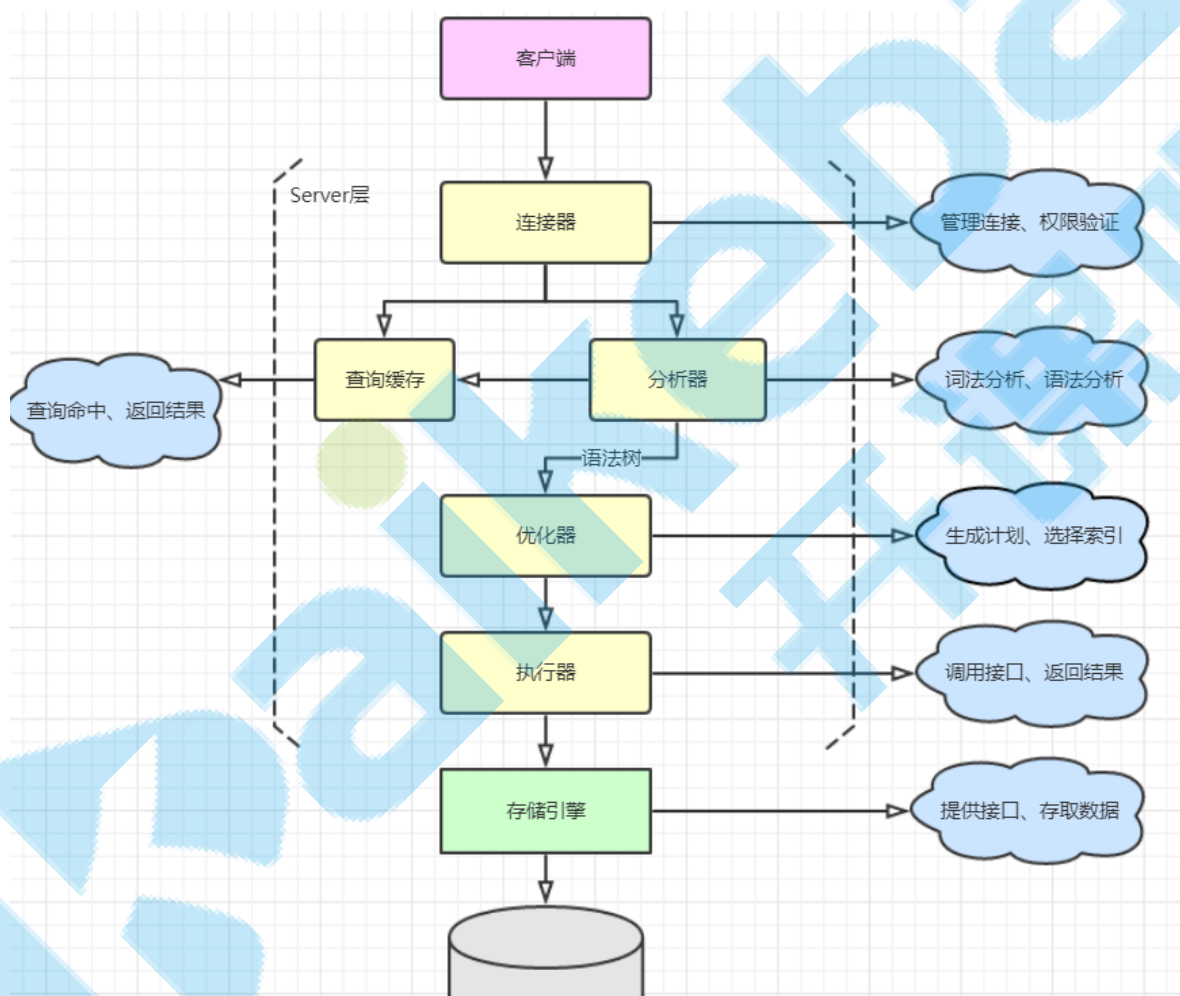
**InnoDB**：支持事务处理，支持外键，支持崩溃修复能力和并发控制。如果需要**对事务的完整性要求比较高**（比如银行），**要求实现并发控制**（比如售票），那选择InnoDB有很大的优势。如果需要**频繁的更新、删除操作**的数据库，也可以选择InnoDB，因为支持事务的提交（commit）和回滚（rollback）。

**MyISAM**：插入数据快，空间和内存使用比较低。如果表主要是**用于插入新记录和读出记录**，那么选择MyISAM能实现处理高效率。如果应用的完整性、并发性要求比较低，也可以使用。

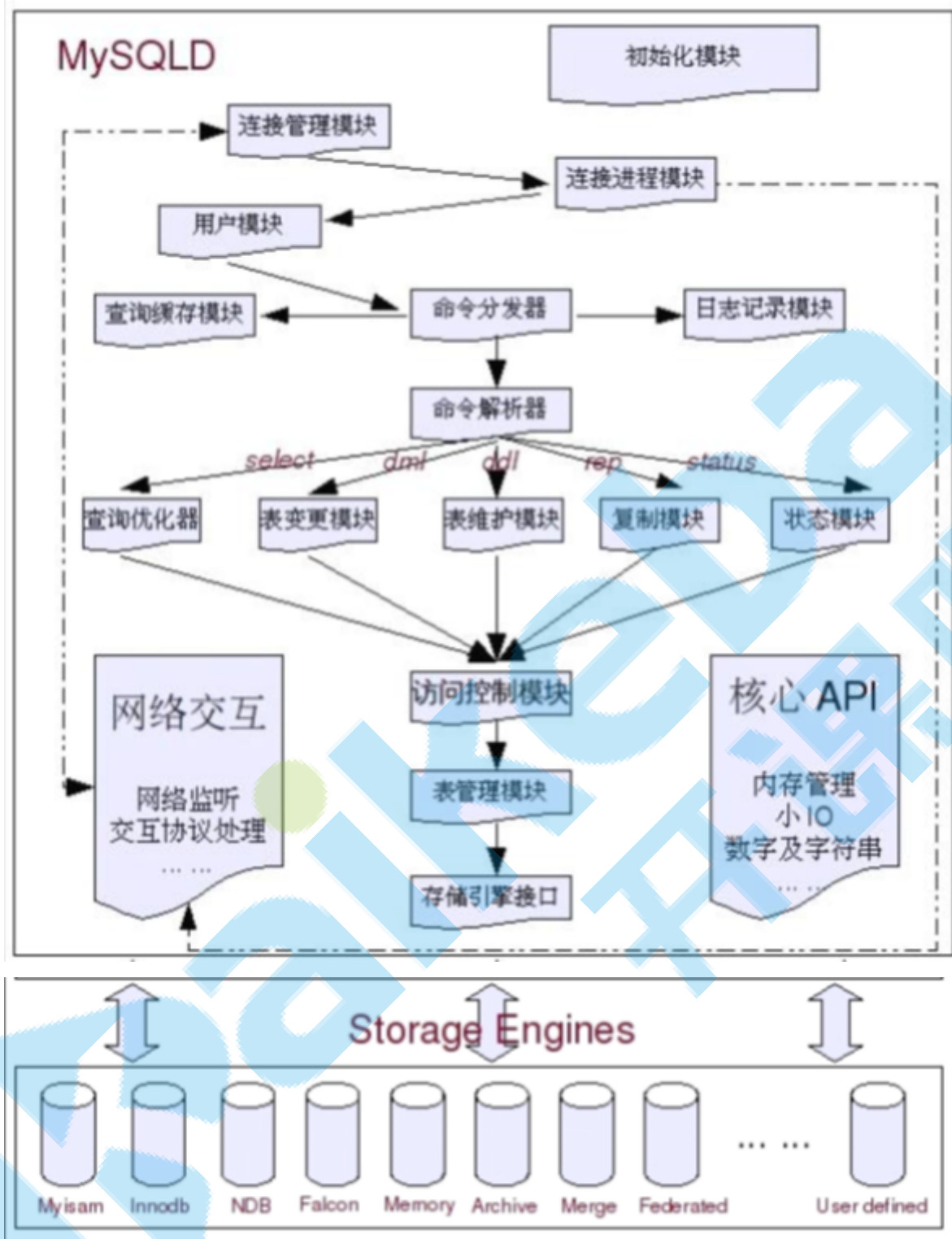
**MEMORY**：所有的数据都在内存中，数据的处理速度快，但是安全性不高。如果需要**很快的读写速度**，对数据的安全性要求较低，不需要持久保存，可以选择MEMORY。它对表的大小有要求，不能建立太大的表。所以，这类数据库只使用在相对较小的数据库表。

注意，同一个数据库也可以使用多种存储引擎的表。如果一个表要求比较高的事务处理，可以选择InnoDB。这个数据库中可以将查询要求比较高的表选择MyISAM存储。如果该数据库需要一个用于查询的临时表，可以选择MEMORY存储引擎。

## 简版执行流程图



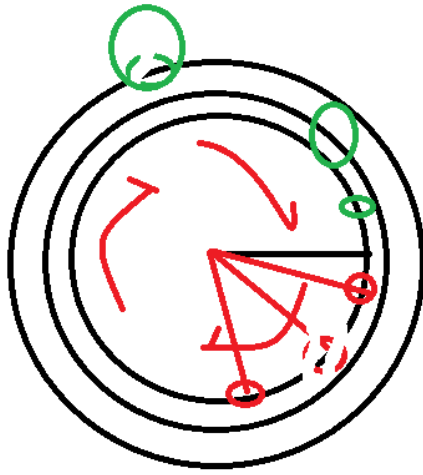
## 详细执行流程图



## 物理结构

- MySQL是通过文件系统对数据和索引进行存储的。
- MySQL从物理结构上可以分为日志文件和数据索引文件。
- MySQL在Linux中的数据索引文件和日志文件都在/var/lib/mysql目录下。
- 日志文件采用顺序IO方式存储、数据文件采用随机IO方式存储。





顺序IO

首地址

偏移量

优势：记录速度快 只能追加

劣势：浪费空间

适合：日志

随机IO

记录地址 0x2345

优势：省空间

劣势：相对慢

适合：数据+索引

## 日志文件

### 错误日志 (errorlog)

默认是开启的，而且从5.5.7以后无法关闭错误日志，错误日志记录了运行过程中遇到的所有严重的错误信息，以及 MySQL 每次启动和关闭的详细信息。

### 二进制日志 (bin log)

记录数据变化

binlog记录了数据库所有的ddl语句和dml语句，但不包括select语句内容，语句以事件的形式保存，描述了数据的变更顺序，binlog还包括了每个更新语句的执行时间信息。如果是DDL语句，则直接记录到binlog日志，而DML语句，必须通过事务提交才能记录到binlog日志中。生产中开启

数据备份、恢复、主从

### 通用查询日志 (general query log)

啥都记录 耗性能 生产中不开启

### 慢查询日志 (slow query log)

SQL调优 定位慢的 select

默认是关闭的。

需要通过以下设置进行开启：

```
#开启慢查询日志
slow_query_log=ON
#慢查询的阈值
long_query_time=3
#日志记录文件如果没有给出file_name值，默认为主机名，后缀为-slow.log。如果给出了文件名，但不是绝对路径名，文件则写入数据目录。
slow_query_log_file=file_name
```

记录执行时间超过`long_query_time`秒的所有查询，便于收集查询时间比较长的SQL语句

**重做日志 (redo log)**

**回滚日志 (undo log)**

**中继日志 (relay log)**

```
# 大小写不敏感
lower_case_table_names=1
# 默认字符集
character-set-server=utf8

#bin-log
log-bin=mysql-bin
# general query log
#general_log=on
#general_log_file=/var/lib/mysql/gen-log.log
# slow query log
#slow_query_log=ON
#long_query_time=3
#slow_query_log_file=/var/lib/mysql/slow-log.log

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

看日志开启情况：

```
show variables like 'log_%';
```

## 数据文件

```
SHOW VARIABLES LIKE '%datadir%';
```

InnoDB数据文件

- **.frm文件**：主要存放与表相关的数据信息,主要包括**表结构的定义信息**
- **.ibd**：使用**独享表空间**存储**表数据和索引**信息，一张表对应一个ibd文件。



- **ibdata文件**：使用**共享表空间**存储**表数据和索引**信息，所有表共同使用一个或者多个ibdata文件。

## MyIsam数据文件

- **.frm文件**：主要存放与表相关的数据信息,主要包括**表结构的定义信息**
- **.myd文件**：主要用来存储**表数据信息**。
- **.myi文件**：主要用来存储**表数据文件中任何索引的数据树**。

# 二、MySQL索引篇

## 索引介绍

### 索引是什么

官方介绍索引是帮助MySQL**高效获取数据的数据结构**。更通俗的说，数据库索引好比是一本书前面的目录，能**加快数据库的查询速度**。

### 索引的优势和劣势

优势：

可以提高数据检索的效率，降低数据库的IO成本，类似于书的目录。 -- 检索

通过索引列对数据进行排序，降低数据排序的成本，降低了CPU的消耗。 --排序

- 被索引的列会自动进行排序，包括【单列索引】和【组合索引】，只是组合索引的排序要复杂一些。
- 如果按照索引列的顺序进行排序，对应order by语句来说，效率就会提高很多。
- where 索引列 在存储引擎层 处理 索引下推 ICP
- 覆盖索引 select 字段 字段是索引

劣势：

索引会占据磁盘空间

索引虽然会提高查询效率，但是会降低更新表的效率\*\*。比如每次对表进行增删改操作，

MySQL不仅要保存数据，还有保存或者更新对应的索引文件。

## 索引的分类

### 单列索引

### 组合索引 \*

### 全文索引

### 空间索引

### 位图索引 Oracle

## 索引的使用

## 创建索引

- 单列索引之普通索引

```
CREATE INDEX index_name ON table(column(length)) ;  
ALTER TABLE table_name ADD INDEX index_name (column(length)) ;
```

- 单列索引之唯一索引

```
CREATE UNIQUE INDEX index_name ON table(column(length)) ;  
alter table table_name add unique index index_name(column);
```

- 单列索引之全文索引

```
CREATE FULLTEXT INDEX index_name ON table(column(length)) ;  
alter table table_name add fulltext index_name(column)
```

- 组合索引

```
ALTER TABLE article ADD INDEX index_titme_time (title(50),time(10)) ;
```

## 删除索引

```
DROP INDEX index_name ON table
```

## 查看索引

```
SHOW INDEX FROM table_name \G
```

## 索引原理分析

### 索引的存储结构

#### 索引存储结构

- 索引是在**存储引擎中实现**的，也就是说不同的存储引擎，会使用不同的索引
- **MyISAM和InnoDB存储引擎**：只支持**B+ TREE索引**，也就是说**默认使用BTREE，不能够更换**
- **MEMORY/HEAP存储引擎**：支持HASH和BTREE索引

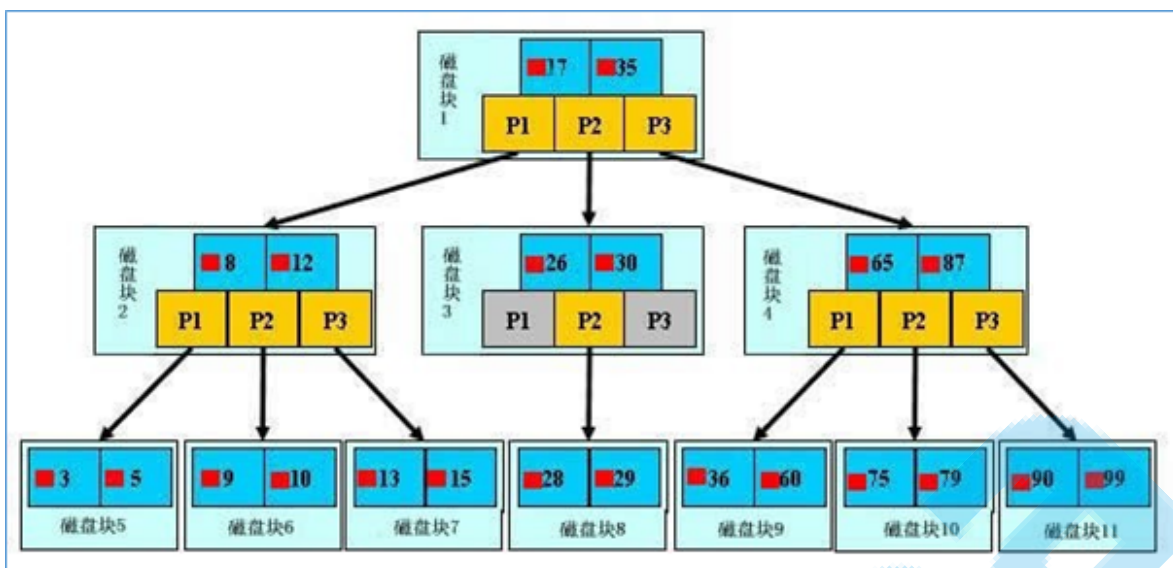
### B树和B+树

数据结构示例网站：

<https://www.cs.usfca.edu/~galles/visualization/Algorithms.html>

#### B树图示

B树是为了磁盘或其它存储设备而设计的一种多叉（下面你会看到，相对于二叉，B树每个内结点有多个分支，即多叉）平衡查找树。多叉平衡



- B树的高度一般都是在2-4这个高度，树的高度直接影响IO读写的次数。
- 如果是三层树结构---支撑的数据可以达到20G，如果是四层树结构---支撑的数据可以达到几十T

B和B+的区别

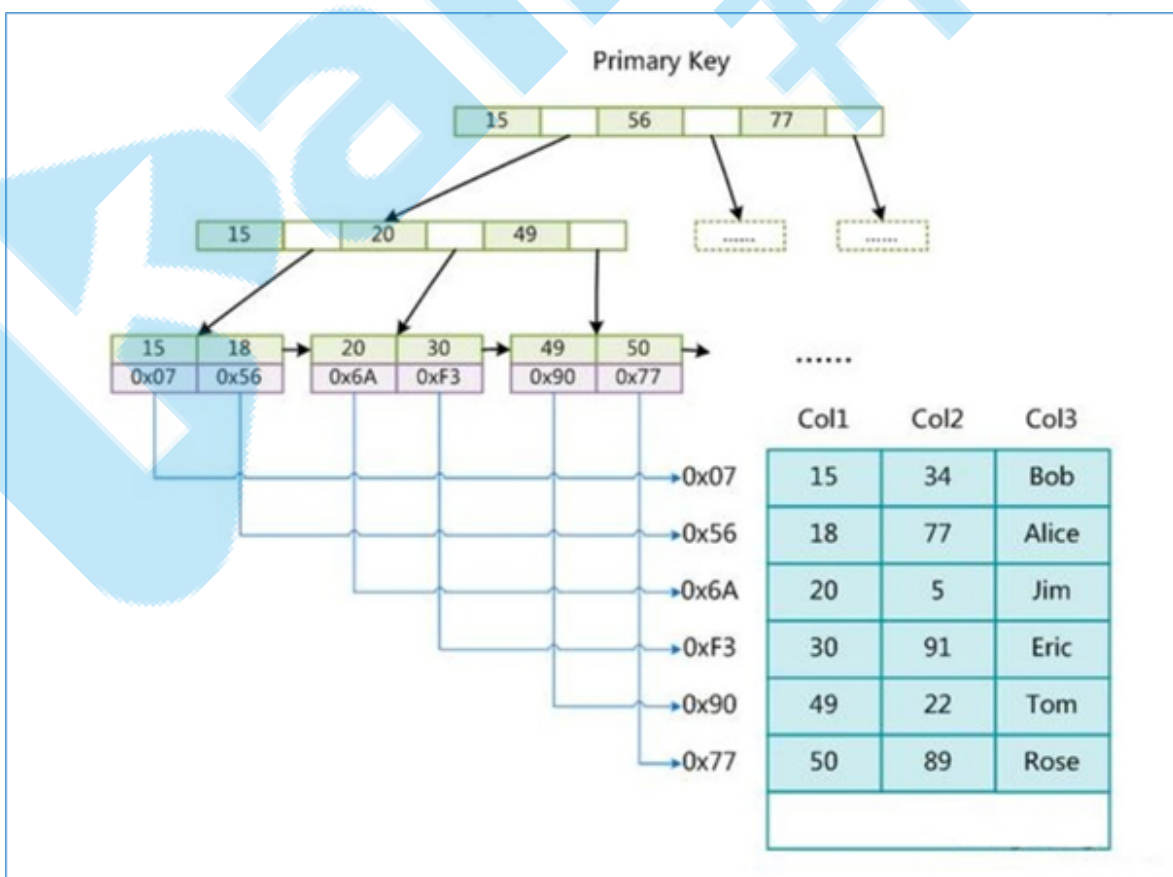
B树和B+树的最大区别在于**非叶子节点是否存储数据**的问题。

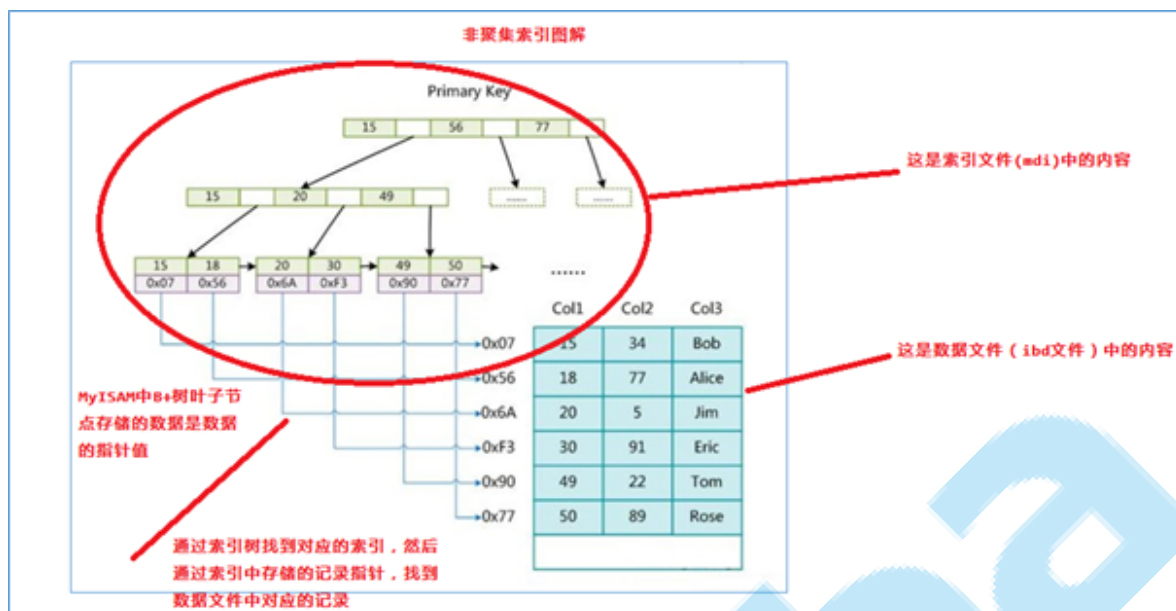
B树是非叶子节点和叶子节点都会存储数据。

B+树只有叶子节点才会存储数据，而且存储的数据都是在一行上，而且这些数据都是有指针指向的，也就是有顺序的。

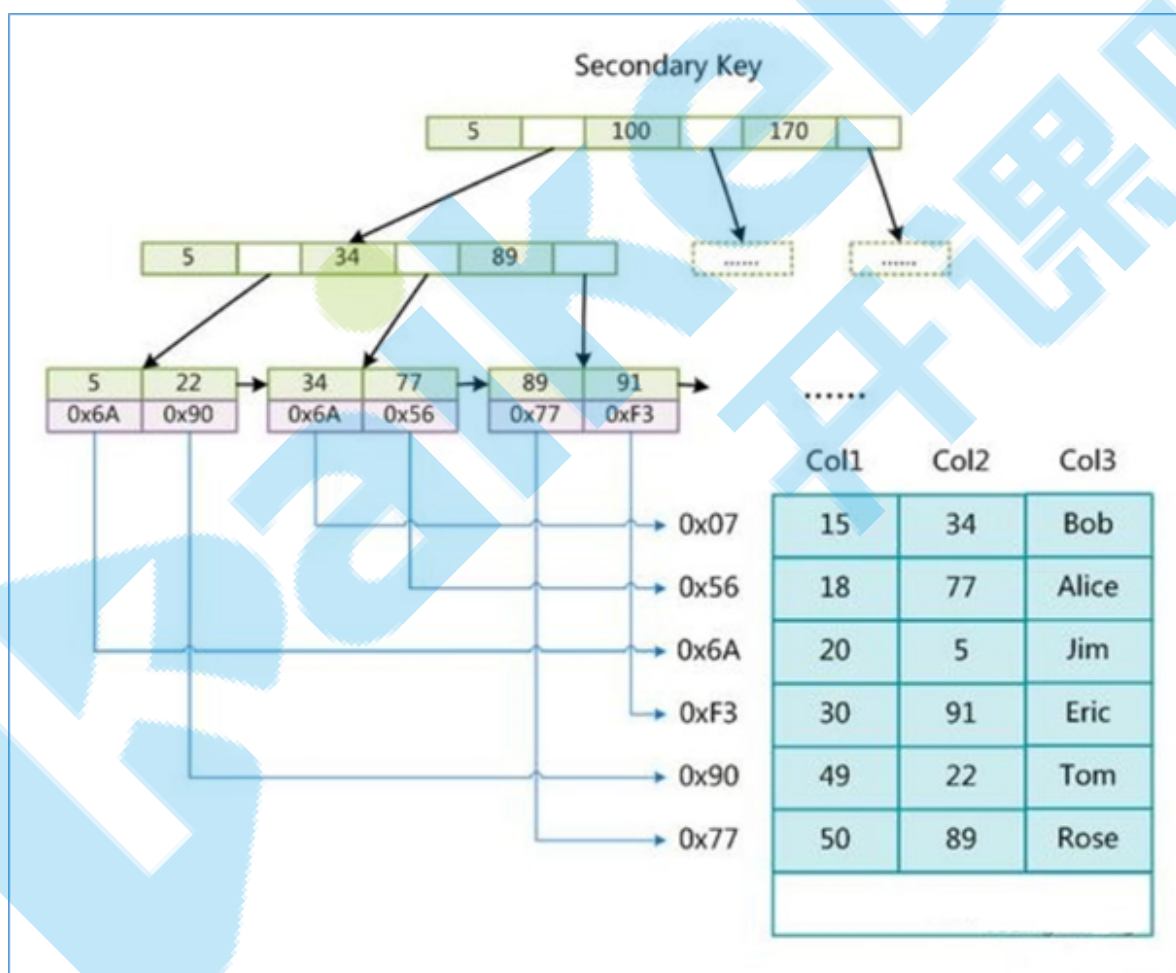
## 非聚集索引 (MyISAM)

### 主键索引



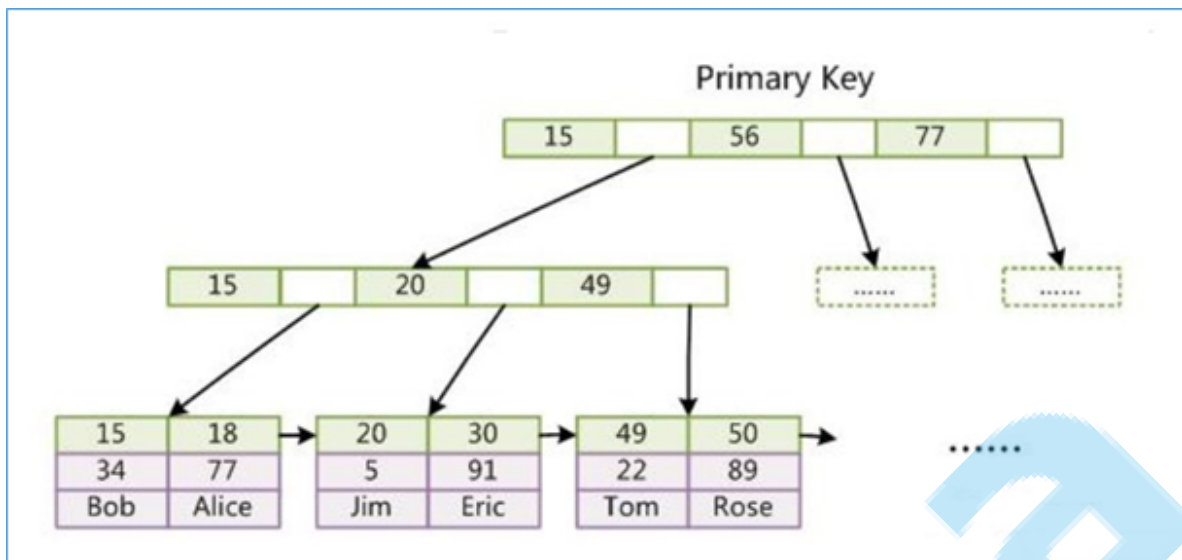


## 辅助索引（次要索引）



## 聚集索引（InnoDB）

### 主键索引



主键：

1、建主键

2、没建主键

找唯一字段 当主键

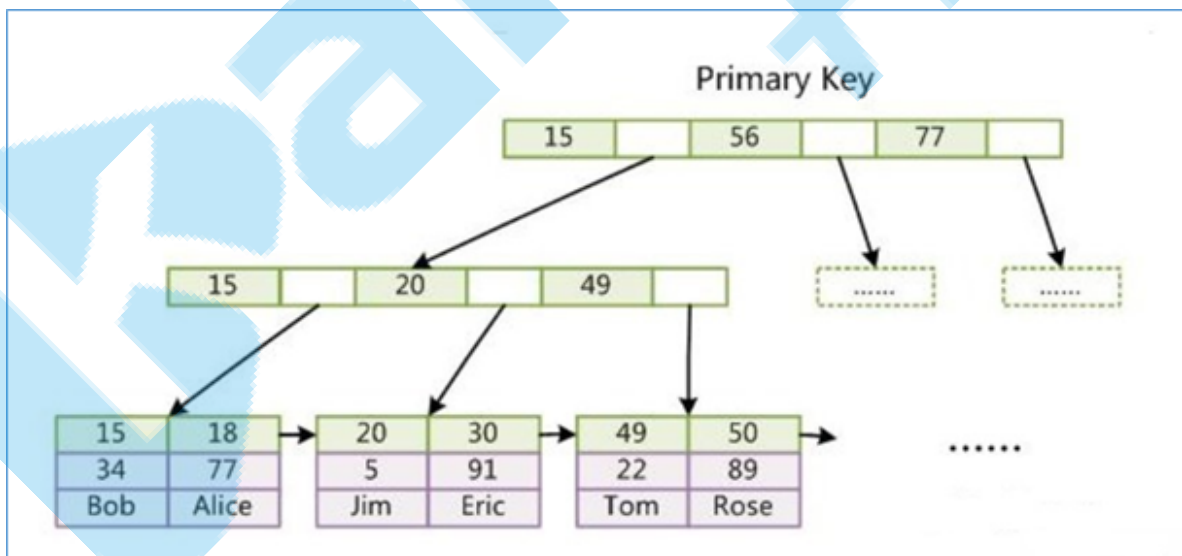
自动生成伪列 当主键

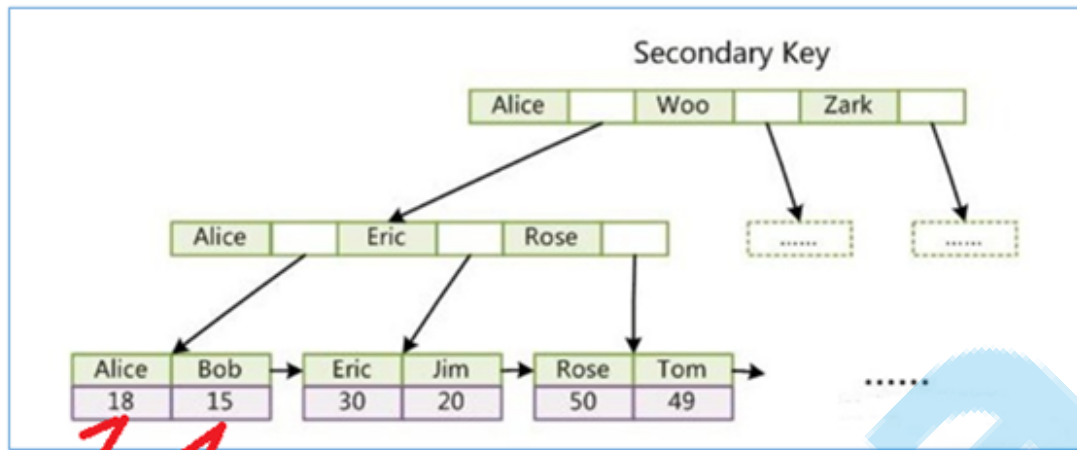
主键创建

自增整数

不要用大字符串比如 uuid ---- 》雪花算法 snowflakes

**辅助索引（次要索引）**





存储的是主键索引中的主键值，不是地址值

结论：如果是非主键查询，则需要搜索两次索引树（一次是name辅助索引树，一次是主键索引树），最终取出来数据。

select \* from t where id=15

select \* from t where name='Alice'

回表(从辅助索引树上找到主键后在主键索引树下找到数据)

select name from t where name='Alice' 给name做了索引

select id,name from t where name='Alice' 覆盖索引

select \* from t where name='Alice' 只找一棵索引树？

形成索引树

利用组合索引 完成覆盖索引（利用组合索引完成在辅助索引树的遍历，不回表）

