MySQL

1. +号：只有一个功能-运算符 如果两个操作数都为数值做加法，如果其中一个为字符型，试图将字符型转换为数值，如果成功则作加法运算，如果失败，将其转为0；如果一方为null结果为null。
2. Distinct 去重。Select distinct age from student；
3. Concat（str1,str2,....）连接符，select concat（name，age）from student；
4. Ifnull（expr1,expr2）判断expr1是否为null 如果为null 输出expr2 否则输出expr1；
5. Not的用法 select \* from employees where not(department\_id > 90 and department\_id <= 100) or department\_id > 150
6. Like 用法：一般和通配符搭配使用 通配符：%任意多个字符，包含0个字符 \_任意单个字符。转义符 \ 或者 指定一个字符为转义符 然后用escape 比如 \_#\_% escape ‘#’
7. 安全等于<=> 可以连接null值和普通值，可读性差 is null 只能判断null值
8. 启动mysql服务-》net start 服务名 停止服务-》net stop 服务名
9. 登陆mysql服务-》 mysql -h 主机名 -P 端口号 -u用户名 -p密码 退出 exit或ctrl+c
10. Isnull（字段）如果是null返回1 不是 返回0；
11. 排序 order by asc升序 desc降序 order by 子句可以支持单个字段/多个字段/表达真实/函数/别名，order by 子句一般放在查询语句的最后面，limit子句除外
12. 字符函数：

* length(‘张三丰hahaha’) 中文占三个字节utf-8 gbk两个字节 这儿返回字节数
* Concant
* Upper（） lower（）转换为大写/小写
* Substr（‘abcdefg’，6）-》fg下标从1开始
* Substr（‘abcdefg’，1，3）-》abc

Instr instr(‘abcdefg’,’def’)->4 返回子串第一次出现的下标，找不到返回0

* Trim（‘ aaaa ’）去掉前后空格
* Trim（‘a’ from‘aaababaaaa’）去掉前后a ->bab
* Lpad lpad(‘aaa’,10,’+’) 左填充 +++++++aaa
* Rpad（aaa，10，‘+’）右填充 aaa+++++++
* Replace 替换 replace（‘aaabbbccc’，’a’,’b’）bbbbbbccc

1. 数学函数
2. Round(1.65) 绝对值四舍五入 然后加符号
3. Round（1.657，2）保留两位小数
4. Ceil（1.12） 向上取整 返回>=该参数的最小整数
5. Floor 向下取整，返回<=该参数的最大整数
6. Truncate（1.65，1）-》1.6 截断
7. Mod（10，3）1 取余
8. 日期函数
9. Now() 当前时间
10. Curdate（） 返回当前系统日期，不包含时间
11. Curtime（） 返回当前时间，不包含日期
12. Year（date）month(date) 获取指定时间的年/月
13. Str\_to\_date 加ing日期格式的字符转换成指定格式的日期

Str\_to\_date(‘9-13-1999’,’%m-%d-%Y’)

1. date\_format 将日期转换成字符

Date\_format(‘2018/6/6’,%Y年%m月%d日’) 2018年6月6日

G.datediff(date1,date2) 两个日期相差的天数

1. 日期格式符



1. 其他函数
2. version() 版本
3. Database（）；
4. User（）当前用户
5. 流程控制函数
6. if(10>5,’da’,’xiao’);
7. Case函数使用一 switch case的效果

Case 要判断的字段或表达式

When 常量1 then 要显示的值1或语句1；

When 常量2 then 要显示的值2或语句2

.......

Else 要显示的值n或语句n

End

Case函数使用二 类似于多重if

Case when 条件1 then 要显示的值1或语句1

When 条件2 then 要显示的值2

Else 要显示的值n

End

1. 分组函数
2. Sum() 求和 数值型 忽略null
3. Avg()平均值 数值型 忽略null值
4. Max()最大值 参数类型（可排序的就行）
5. Min() 最小值 任何类型 忽略null
6. Count() 计算个数 任何类型 忽略null count(\*)效率高
7. 可以和distinct搭配使用

Select sum(distinct salary),avg(salary),max(salary),min(salary),count(salary) from employees

1. 分组查询

Group by

Select 分组函数,列（要求出现在group by的后面） from 表 where 条件 group by 分组的列表 order by 列 需要注意的是查询列必须特殊，要求是分组函数和group by后出现的字段

Select max(salary) ,job\_id from employees group by job\_id

添加分组后的筛选

Select count(\*),department\_id from employees group by department\_id having count(\*) > 2

数据源 位置 关键字

分组前筛选 原始表 group by子句的前面 where

分组后筛选 分组后的结果集 group by 子句的后面 having

分组函数做条件肯定是分组后筛选

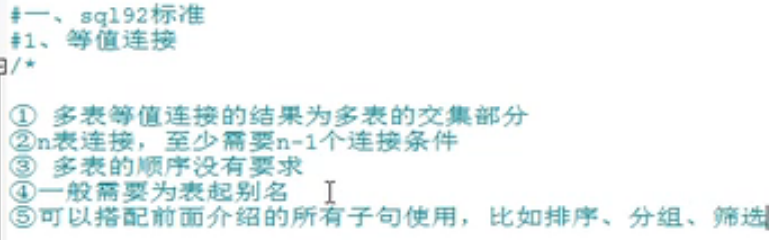
能分组前筛选就分组前筛选

按多个字段分组

Select avg(salary),department\_id,job\_id from employees group by department\_id,job\_id 效果是这两个字段值都一样的分为一组

1. 连接sql92

等值连接



Select last\_name,department\_id from employees,departments

Where employees.department\_id = department\_id.department\_id

Select count(\*) ,city from departments d,locatings l

Where d.locating\_id = l.location\_id group by city

非等值连接

Select salary,grade\_level from employees e,job\_grades g

Where salary between g.lowest\_sal and g.highest\_sal and g.grade\_level = ‘A’

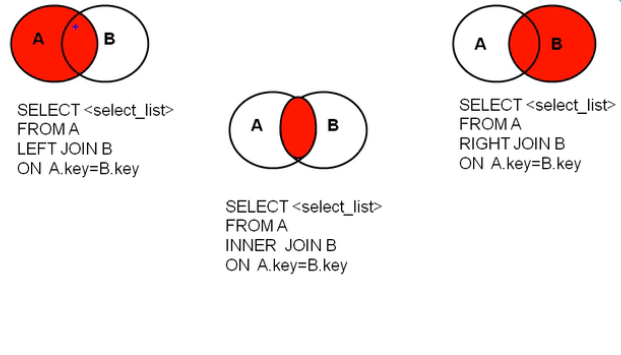
自连接

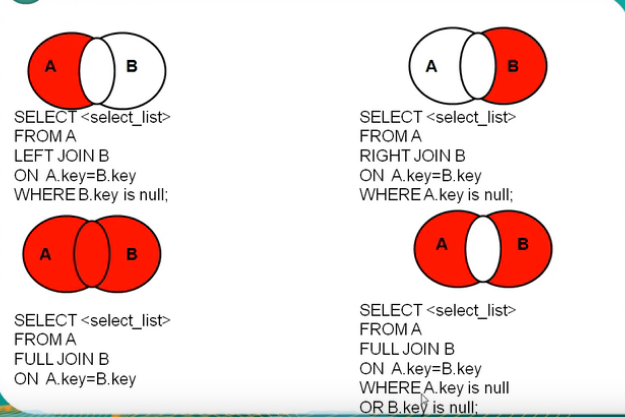
Select e.employee\_id,e.last\_name,m.employee\_id,m.last\_name

From employees e,employees m

Where e.employee\_id = m.employee\_id

1. sql99 from后面的表为主表





1. 子查询

含义：出现在其他语句中的select语句，称为子查询

示例：

Select first\_name from employees where department\_id in(

Select department\_id from departments where location\_id = 1700

)

分类：

按子查询出现的位置：

Select 后面 ： 仅仅支持标量子查询

From 后面 支持表子查询

Where或having后面 标量子查询/列子查询/行子查询

Exists 后面（相关子查询）

按结果集的行列数

标量子查询（结果集只有一行一列）

列子查询（结果集一列多行）

行子查询（结果集一行多列）

表子查询（结果集一般为多行多列）

Where或having后面

特点：

子查询放在小括号内

子查询一般放在条件的右侧

标量子查询一般搭配着单行操作符使用 > < = <>...

列子查询一般搭配着多行操作符使用 in any some all

子查询的执行优先于主查询执行，主查询的条件用到了子查询的结果



Select \* from employees where salary > any(

Select salary from employees

) 等同于 salary > min(select salary from employees)

行子查询：

Select \* from employees where (employee\_id ,salary) = (

Select min(employee\_id),max(salary) from employees

)

select后面

Select d.\*,(

Select count(\*) from employees e

Where e.department\_id = d.department\_id

) qty from departments d;

from后面

Select ag\_dep.\*,grade\_level

From(

Select avg(salary) ag,department\_id from employees group by department\_id

) ag\_dep

Join job\_grades g on ag\_dep.ag between g.lowest\_sal and g.highest\_sal

exists后面 相关子查询

Select exists(完整的查询语句) 结果1或0

有员工的部门名

Select department\_name from departments d

Where exists(

Select \* from employees e where d.department\_id = e.department\_id

)

1. 联合查询

关键字Union

要求多条查询语句的查询列数是一致的

要求多条查询语句的查询的每一列的类型和顺序最好一致

Union关键字会去重，union all不会去重

1. 插入

语法：

方式一：Insert into表名(列名1,列名2,...) values(值1,值2,....)

方式二：insert into 表名 set 列名=值，列=值，。。。

注意：

插入的值的类型要与列的类型一致或兼容

可以为null的列如何插入值？可以列括号里不加这个列或者值括号里这个列对应的值放null

列的顺序可以调换，但是对应的值跟着调换

列和值的数量必须一致

可以省略列名，默认所有列（但是比如主键id是自增长，可忽略），而且列的顺序和表中列的顺序一致

两种方式的比较：

方式一支持插入多行，方式二不支持

Insert into students values (值1，值2，...),(值3，值4，...);

方式一支持子查询，方式二不支持

insert into students select \* from student2

1. 修改

语法：

Update 表名 set 列=新值，列=新值。。。 where 筛选条件

Update 表1 别名，表2 别名 set 列= 值，。。。

Where 连接条件 and 筛选条件

Update 表1 别名 inner|left|right join 表2 别名

On 连接条件

Set 列=值，。。。。

Where 筛选条件

1. 删除

方式一：delete from 表名 where 筛选条件

多表删除

Delete 表一别名，表二别名 （这里注意的是删除哪个表的数据就在delete后面放哪个表的别名）

From 表一 别名，表二 别名

Where 连接条件

And 筛选条件

Delete 表1别名，表2别名

From 表1 别名

Inner|left|right join 表2 别名 连接条件

Where 筛选条件

方式二： truncate table 表名 不能接删除条件，删除整个表数据

1. 数据库创建

语法：create database 库名；

Create database if not exists students

1. 数据库修改

一般不修改

更改库的字符集 alter database students character set gbk;

1. 数据库删除

Drop database if exists students

1. 表的创建

语法：create table 表名(

列名 列的类型(长度) 约束,

列名 列的类型(长度) 约束,

列名 列的类型(长度) 约束,

列名 列的类型(长度) 约束,

列名 列的类型(长度) 约束

)

1. 表的修改

语法：

Alter table 表名 add|drop|modify|change column 列名【类型 约束】

修改表名 alter table student rename to students

修改列名 alter table student change column studentName name vachar; column可以不要

修改列的类型或约束

Alter table students modify column name char

添加删除列

Alter table students add column age int

Alter table students drop column age

1. 表的删除

Drop table if exist students;

1. 表的复制

仅仅复制表的结构

Create table copy like students

复制表结构+数据

Create table copy2 select 列1，列2，列3 from students where 筛选条件

仅仅复制某些字段结构

Create table copy3

Select id，name from students where 1=2

1. 数据类型
   1. 整型：

Tinyint smallint mediumint int/integer bigint

字节1 2 3 4 8

如何设置有无符号

Create table tab\_int(

T1 int,

T2 int unsigned, 无符号

T3 int(7) zerofill 如果位数不够零填充，变成无符号

)

如果插入的数值超出了范围，会报out of range，会插入临界值

如果不设置长度会有默认长度，如果要填充满位数用zerofill

* 1. 小数

浮点型 float 4字节 double 8字节

定点型 dec(M,D)/decimal(M,D) 精度更高。D表示小数点后位数，会四舍五入

M表示总的位数（整数位+小数位），超出范围会写成临界值

* 1. 字符型

较短的文本

Char 固定长度 效率更高

Vachar 可变长度 效率低些

较长的文本

Text

Blob（较大的二进制）

其他：

Binary/ varbinary 用于保存较短的二进制

Enum 用于保存枚举

Set 用于保存集合

Create table tab\_char(

EOne enum(‘a’,’b’,’c’), 如果不在这个范围，值存不进去

SOne set(‘a’,’b’,’c’,’d’)

)

* 1. 日期型



Datetime 和timestamp的区别

Timestamp支持的时间范围较小，取值范围：19700101080001-2038年的某个时间 datetime的取值范围 1000-1-1 00:00:00 9999-12-31 23:59:59

timestamp和实际时区有关，更能反映实际的日期，而datetime则只能反映出插入时的当地时区

Timestamp的属性受mysql版本和sqlmode的影响很大

1. 常见约束

含义：用于限制表中的数据，为了保证表中的数据䣌准确和可靠性

分类：六大约束

Not null：非空，用于保证该字段的值不能为空

Default：默认，用于保证该字段有默认值

Primary key：主键，该字段的值具有唯一性，非空

Unique：唯一，保证该字段的值唯一性，可以为空，但也只有一个值能为空

Check：检查约束（mysql中不支持）

Foreign key：外键，用于限制两个表的关系，用于保证该字段的值必须来自于主表的关联列的值。在从表添加外键约束用于引用主表中某列的值。

添加约束分类：

列级约束：六大约束语法上都支持，但外键约束没效果

表级约束：除了非空和默认，其他都支持

Create table 表名(

字段名 字段类型 列级约束，

字段名 字段类型，

表级约束

)

列级约束：

Create table stuinfo(

Id int primary key, 主键

stuName varchar(20) not null, 非空

Gender char(1) check(gender = ‘男’ or gender = ‘女’), 检查

Seat int unique, 唯一

Age int default 18, 默认

majorId int foreign key references major(id) 外键

)

表级约束：

Create table stuinfo(

Id int,

stuName varchar(20),

Gender char(1) ,

Seat int ,

Age int ,

majorId int,

Constraint pk primary key(id),

Constraint uq unique(seat),

Cpmstraint ck check(gender = ‘男’ or gender = ‘女’),

Constraint fk\_stuinfo\_major foreign key(majorid) references major(id) 外键 有效生成外键

) constraint 约束名 可以不加

外键：

要求在从表设置外键关系

从表的外键列的类型和主表的关联列的类型要求一致或兼容，名称无要求

主表的关联列必须是一个key（一般是主键或唯一）

插入数据时，先插入主表，再插入从表

删除输入时，先删除从表，再删除主表

修改表时添加约束

添加

Alter table stuinfo modify column stuname vachar(20) not null

添加主键

列级约束：alter table stuinfo modify column id int primary key

表级约束：alter table stuinfo add primary key(id);

修改表示删除约束

删除非空约束

Alter table stuinfo modify column stuname varchar(20) null

删除主键

Alter table stuinfo drop primary key 主键名

删除唯一

Alter table stuinfo drop index seat

标识列 auto\_increment

又称为自增长列，可以不用手动的插入值，系统提供默认的序列值。

标识列不一定必须和主键搭配，但要求是一个key

一个表只能有一个标识列

表示列的类型必须是数值型

标识列可以通过 set auto\_increment\_increment = 3;设置步长为3

1. TCL 事务控制语言

事务：事务由单独单元的一个或多个sql语句组成，在这个单元中，每个mysql语句是相互依赖的。整个单独单元作为一个不可分割的整体，如果单元中某条sql语句一旦执行失败或产生错误，整个单元将会回滚。所有受到影响的数据将返回事务开始以前的状态，如果单元中所有sql语句均执行成功，则事务顺利执行。

事务的ACID属性

原子性：事务是一个不可分割的工作单位，事务中的操作要么都发生，要么都不发生。

一致性：事务必须使数据库从一个一致性状态变换到另外一个一致性状态

隔离性：一个事务的执行不能被其他事务干扰，即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰

持久性：一个事务一旦被提交，它对数据库中数据的改变就是永久性的，接下来的其他操作和数据库故障不应该对其有任何影响

事务的创建

隐式事务：事务没有明显的开启和结束的标记 比如 insert update delete语句

显示事务：事务具有明显的开启和结束的标记

前提：必须先设置自动提交功能为禁用

Set autocommit = 0; 仅针对当前事务有效

步骤1：开启事务

Set aotocommit = 0;

Start transaction;可写可不写

步骤2:编写事务中的sql语句（select insert update delete）

步骤3：结束事务

Commit 提交

Rollback 回滚

对于同时运行的多个事务，当这些事务访问数据库中相同的数据时，如果没有采取必要的隔离机制，就会导致各种并发问题。

脏读：对于两个事务T1,T2,T1读取了已经被T2更新但还没有被提交的字段之后，若T2回滚，T1读取的内容就是临时且无效的

不可重复读：对于两个事务T1,T2,T1读取了一个字段，然后T2更新了该字段之后再次读取同一个字段，值就不同了

幻读：对于两个事务T1,T2,T1从一个表中读取了一个字段，然后T2在该表中插入了一些新的行之后，如果T1再次读取同一个表，就会多出几行

1. 事务隔离性

数据库系统必须具有隔离并发运行各个事务的能力，使它们不会相互影响，避免各种并发问题。

数据库提供4中事务隔离级别

Read uncommitted（读未提交数据） 允许事务读取未被其他事务提交的变更，脏读，不可重复度和幻读的问题都会出现

Read commited （读已提交数据） 只允许事务读取已经被其他事务提交的变更，可以避免脏读，但不可重复读和幻读问题仍然可能出现

Repeatable read （可重复读） 确保事务可以多次从一个字段中读取相同的值，在这个事务持续期间，禁止其他事务对这个字段进行更新，可以避免脏读和不可重复读，但幻读的问题仍然存在

Serializable（串行化）确保事务可以从一个表中读取相同的行，在这个事务持续期间，禁止其他事务对该表执行插入，更新和删除操作，所有并发问题都可避免，但性能十分低下。

Oracle支持2种事务隔离级别：read commited，serializable。

Mysql支持以上4中，默认为repeatable read

设置方式 set session transaction isolation level read uncommitted

1. 存储引擎

概念：在mysql中的数据用各种不同的技术存储在文件（或内存）中。

通过show engines; 来查看mysql支持的存储引擎。

在mysql中庸的最多的存储引擎有：innodb，myisam， memory等。其中innodb支持事务

1. 系统变量

系统变量由系统提供，不是用户定义，属于服务器层面

1. 查看所有的系统变量

Show global | session variables like ‘’

1. 查看满足条件的部分系统变量

Show global | session variables like ‘’

1. 查看指定的某个系统变量的值

Select @@global | session 系统变量名

1. 为某个系统变量赋值

Set global | session 系统变量名=值

如果是全局级别，则需要加global，如果是会话级别，则需要加session，如果不屑，默认session

全局变量：

作用域：服务器每次启动将为所有的迁居变量赋初始值，针对所有的会话有效，但不能跨重启

会话变量：

作用域：仅仅针对于当前会话有效

1. 自定义变量

变量是用户自定义的，不是系统提供的

作用域：针对于当前会话有效，同于会话变量的作用域

应用在begin end 里面或外面都可以

使用：

1. 声明并初始化

Set @用户变量名= 值

Set @用户变量名：=值或

Select @用户变量名：=值

1. 赋值

通过set或select 就是a中的用法

通过select into

Select 字段 into 变量名 from 表

1. 查看变量值

Select @变量名

局部变量

作用域：仅仅在定义它的begin end中有效

应用在begin end 中

声明：

declare 变量名 类型

declare 变量名 类型 default 值；

赋值：

Set 局部变量名 = 值；或

Set 局部变量名 := 值；或

Select @局部变量名:=值

Select 字段 into 局部变量名 from 表

使用：select 局部变量名；

1. 存储过程

含义：一组预先编译好的sql语句的集合，理解成批处理语句

创建语法：

Create procedure 存储过程名（参数列表）

Begin

存储过程体

End

注意：

参数列表包含三部分

参数模式 参数名 参数类型

比如：

In statuname varchar(20)

参数模式：

In：该参数可以作为输入，也就是该参数需要调用方传入值，可省略

Out：该参数可以作为输出，也就是该参数可以作为返回值

Inout：该参数既可以作为输入又可以作为输出，也就是该参数既需要传入值，又可以返回值

如果存储过程体仅仅只有一句话，begin end 可以省略

存储过程体中的语句结尾需要加分号

存储过程的结尾可以使用delimiter 重新设置

语法

Delimiter//

调用语法

Call 存储过程名(实参列表)；

调用in call sq1（‘值’）

调用out set @var；call sq1(@var)

调用inout set @var = 值；call sp1（@var）；select @var；

查看语法

Show create procedure 存储过程名

删除语法

Drop procedure if exist 存储过程名

1. 函数

存储过程可以由0个返回，也可以有多个

函数有且只有一个返回

创建语法：

Create function 方法名(参数列表) returns 返回数据类型

Begin

方法体

End

参数列表里的参数格式为 参数名 参数类型，参数名 参数类型.....

使用return关键字 返回值

调用语法

Select 函数名(实参列表)

查看语法

Show create function 函数名；

删除语法

Drop function if exist 函数名

1. 流程控制

分支结构

If函数 if(条件，值1，值2)

Case结构

Case 表达式或字段

When 值1 then 语句1

When 值2 then 语句2

...

Else 语句n

End case

Case

When 条件1 then 语句1

When 条件2 then 语句2

...

Else 语句n

End case

If结构

If 条件1 then 语句1

Elseif 条件2 then 语句2

....

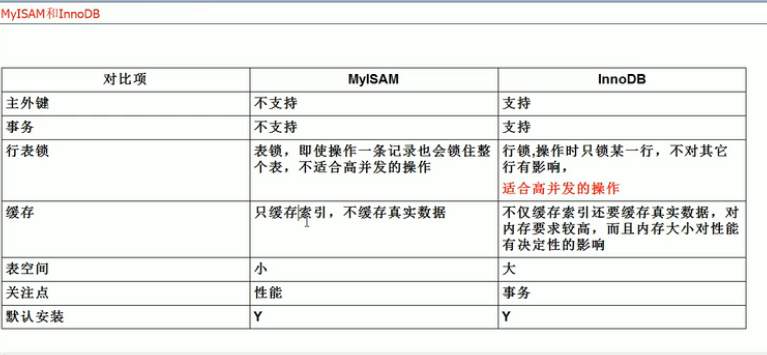
Else 语句n

End if

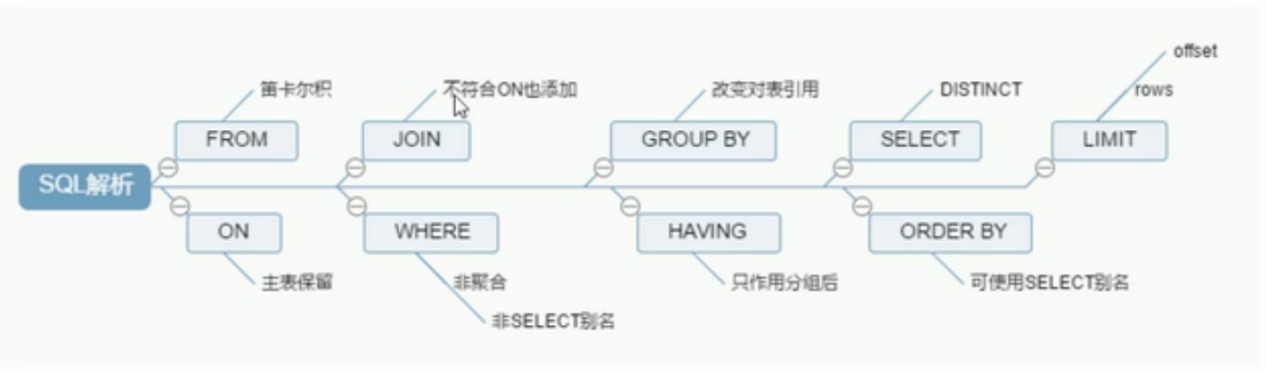
循环结构：语法如下图



1. 存储引擎MyISAM和InnoDB比较



1. Percona 为mysql数据库服务器进行了改造
2. sql解析



1. 排好序的快速查找数据结构-》索引

在数据库之外，数据库系统还维护着满足特定查找算法的数据结构，这些数据结构以某种方式引用（指向数据），这样就可以在这些数据结构上实现高级查找算法。这种数据结构就是索引。

1. 索引语法

创建：create[unique] index indexName on tableName(columnname(length))

alter tableName add [unique] index [indexName] on (columnname(length))

加上unique表示唯一索引

删除：drop index [indexName] on tableName

查看：show index from tableName;

1. 调优Explain

语法：Explain + SQL语句

作用：

表的读取顺序

数据读取操作的操作类型

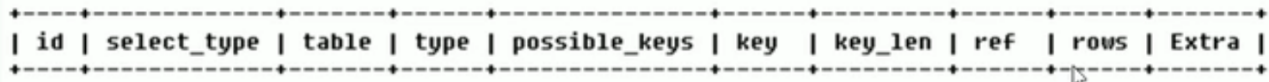
哪些索引可以使用

哪些索引被实际使用

表之间的引用

每张表有多少行被优化器查询

执行计划包含的信息：



id：select查询的序列号，包含的一组数字，表示查询中执行select子句或操作表的顺序。值有3种：id相同 执行顺序从上到下；id不同 如果是子查询，id的序号会递增，值越大越先执行 id相同不同同时存在 id如果相同，可以认为是同一组，从上往下执行 在所有组中，id值越大越先执行 derived=衍生 derived2 表示id为2的那个衍生出来的表

select\_type:值有6种，分别是 simple/ primary/ subquery/ derived/ union/ union result，主要用于区别普通查询，联合查询，子查询等复杂查询。

simple：简单的select查询，不包含子查询或者union

primary：查询中若包含任何复杂的子部分，最外层查询则被标记为primary

subquery：在select或where列表种包含了子查询

derived：在from列表种包含的子查询被标记为derived(衍生) mysql会递归执行这些子查询，把结果放在临时表里

union：若第二个select出现在union，则被标记为union，若union包含在from子句的子查询种，外层select被标记为derived

union result：从union表获取结果的select

table：显示哪张表

type：访问类型，有8种值，显示查询使用了何种类型，从最好到最差依次是：system>const>eq\_ref>ref>range>index>All

system：表只有一行记录，等于系统表

const：表示通过索引一次就找到了，const用于比较primary key或者unique索引。因为只匹配一行，所以很快，如将主键置于where条件种，mysql就能将该查询转换为一个常量。

eq\_ref：唯一性索引扫描，对每个索引键，表中只有一条记录与之匹配，常见于主键或唯一索引扫描

ref：非唯一性索引扫描，返回匹配某个单独值的所有行。本质上也是一种索引访问，它返回所有匹配某个单独值的行，然后，它可能会找到多个符合条件的行，它应该属于查找和扫描的混合体。

range：只检索给定范围的行，使用一个索引来选择行。key列显示使用了哪个索引，一般就是在你的where语句种出现了between/</>/in等的查询，这种范围扫描索引扫描比全表扫描要好。

index：全索引扫描，只遍历索引树。虽然all和index都是读全表，但index是从索引中读取，all是从硬盘中读的。

all：全表扫描

possible\_keys：显示可能应用在这张表中的索引，一个或多个。查询设计到的字段上若存在索引，则该索引将被列出，但不一定被查询实际使用。

key：实际使用的索引，如果为null，则没有使用索引，查询中若使用了覆盖索引，则该索引仅出现在key列表中。

覆盖索引：一个索引包含或覆盖select所有的字段，顺序不一定需要一致。

key\_len:表示索引中使用的字节数，可通过该列计算查询中使用的索引的长度。在不顺势精确性的情况下，长度越短越好，key\_len显示的值为索引字段的最大可能长度，并非实际使用长度，它是根据表定义计算而得，不是通过表内检索出的

ref：显示索引的那一列被使用了，如果可能的话，是一个常数。哪些列或常量被用于查找所有列上的值。

rows：根据表统计信息及索引使用情况，大致估算出找到所需的记录所需要读取的行数。

extra：额外信息

Using filesort：说明mysqlhi对数据使用一个外部的索引排序，而不是按照表内的索引顺序进行读取。mysql中无法利用索引完成的排序操作成为文件排序

Using temporary：使用了临时表表村中间结果，mysql在对查询结果排序时使用临时表。常见排序order by 和分组查询 group by

Using index 表示相应的select操作中使用了覆盖索引，避免访问了表的数据行，效率不错，如果同时出现using where 表名索引被用来执行索引键值的查找，如果没有同时出现using where 表名索引用来读取数据而非执行查找动作。

Using where 使用了where过滤

using join buffer 使用了连接缓存

impossible where where子句的值总是false，不能用来获取任何元组

select tables optimized away 在没有groupby子句的情况下，基于索引优化min/max操作或者对于MyISAM存储引擎优化count(\*)操作，不必等到执行阶段再进行计算，查询执行计划生成的阶段即完成优化。

distinct：优化distinct操作，在找到第一匹配的元组后即停止找同样值的动作

1. 优化

小表驱动大表，即小的数据集驱动大的数据集。

EXISTS:SELECT ... FROM TABLE WHERE EXISTS(subquery)

将主查询的数据放到子查询中做条件验证，根据验证结果来决定主查询的数据结果是否得以保留

select \* from A where id in(select id from B)

等价于:

for select id from B

for select \* from A where A.id = B.id

当B表的数据集小于A表的数据集时，in优于exists

select \* from A where exists (select 1 from B where B.id = A.id)

等价于

for select \* from A

for select \* from B where B.id = A.id

当A表的数据集小于B表的数据集时，exists优于in

A表与B表的ID字段应该建立索引。

1. Mysql两种排序方式，文件排序或扫描有序索引排序，mysql能为排序与查询使用相同的索引。orderby里面如果既有升序又有降序，会产生filesort

