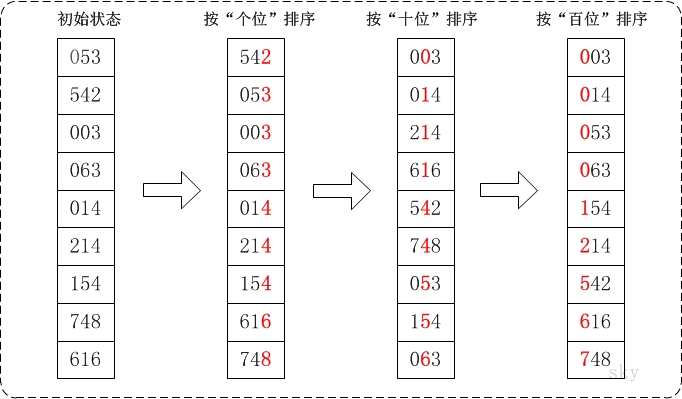
新笔记

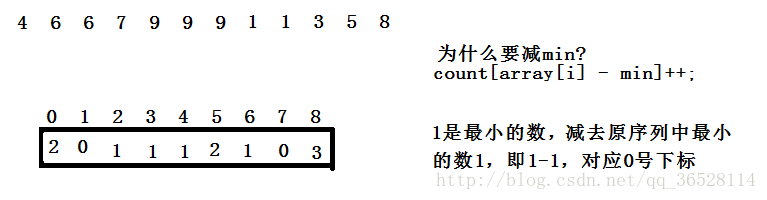
基数排序

先按个位排序，再按十位排序，再按百位排序



计数排序

相同的数对应的数组中的值++，统计每个数有多少个



JVM命令

jinfo:查看Java虚拟机各种配置参数和系统属性

jmap：监控内存内的Java对象，分析和定位频繁GC,堆配置参数和各代中堆内存的使用情况

jstack：用于显示指定进程内线程的信息，监控分析死锁

waiting to lock 等待某个资源的锁

locked 锁住了某个资源

jstat:监控进程GC

数据库 mysql为例

查询字段所属的表

information\_schema数据库中columns表存了每个字段所属的数据库和表

查询某个字段在哪些表中有可以用如下的语句

SELECT \* FROM information\_schema.columns WHERE ABLE\_SCHEMA = '数据库名称' AND COLUMN\_NAME = '字段名称';

分析sql的工具：除了explain,profile

触发器：

查看触发器 show trigger

触发器类型：insert,update,delete

#创建触发器，user表中插入数据时，log中插入一条日志

CREATE TRIGGER log\_trigger AFTER INSERT

ON user\_t FOR EACH ROW

INSERT INTO log\_t(`datetime`, operation) VALUES(NOW(), "insert");

INSERT INTO user\_t(user\_name, user\_password) VALUES("小明","qwe");

#删除触发器

DROP TRIGGER IF EXISTS log\_trigger;

存储过程：

临时表

sql学习总结

mysql介绍

结构化查询语言

1.数据定义语言（DDL：Data Definition Language ） ： create 、drop 、alert

2.数据查询语言（DQL：Data Query Language） ： select 、where 、group by 、having 、order by

3.数据操作语言（DML：Data Manipulation Language） ： insert 、update 、delete

数据库定义语言（DDL）

基础

使用数据库：USE 数据库名;

查看数据库： SHOW DATABASES;

使用数据库： USE 数据库名;

展示表： SHOW TABLES;

展示表结构： DESC 表名

create：创建结构

创建数据库： DROP DATABASE 数据库名;

创建表

CREATE TABLE 表名(

字段名 类型 [约束],

字段名 类型 [约束],

字段名 类型 [约束]

);

auto\_increment : 列自增

复制表结构： CREATE TABLE 表名 LIKE 被复制表名;

复制表结构和数据：CREATE TABLE 表名 SELECT \* FROM 被复制表名 WHERE 1;

alert：修改结构

修改字符集： ALTER DATABASE 表名 CHARACTER SET 字符集;

查看表结构： DESC 表名; 或 SHOW COLUMNS FROM 表名;

改字段名： ALTER TABLE 表名 CHANGE 字段名 新字段名 新字段名类型;

改字段类型/约束： ALTER TABLE 表名 MODIFY 字段名 新字段类型;

添加字段： ALTER TABLE 表名 ADD 字段名 字段类型;

删除字段： ALTER TABLE 表名 DROP 字段名;

改表名： ALTER TABLE 表名 RENAME TO 新表名;

添加列级约束： ALTER TABLE 表名 MODIFY 字段名 类名 约束名;

添加表级约束： ALTER TABLE 表名 [ADD CONSTRAINT 约束名] 约束类型(字段名) [外键的引用];

删除列级约束：ALTER TABLE 表名 MODIFY 字段名 类名;

删除表级约束：ALTER TABLE 表名 DROP 约束类型 约束名;

drop primary key

drop index 约束名

drop foreign key 约束名

修改字段默认值: alter table 表名 alter column 字段名 default 默认值

drop：删除结构

删除数据库： CREATE DATABASE 数据库名;

删除表 ： drop table 表名

删除视图：drop view 视图名

view：视图

作用：sql语句较复杂且使用频繁，不妨将该查询封装（只保存sql逻辑）到视图中（一张虚拟的表），拿出来就能用

表数据改变，对应的视图也会改变，对视图进行增删改，对应表数据也会改变（一般不对视图做改动）

可以在视图上建视图

使用：就当一张表来使用

创建：只保存sql逻辑，表数据改变，对应的视图也会改变，可以在视图上建视图

CREATE VIEW 视图名 AS SELECT 查询语句;

修改：

方式1：CREATE OR REPLACE VIEW 视图名 AS SELECT 查询语句;

方式2：ALTER VIEW 视图名 AS SELECT 查询语句;

删除： DROP VIEW 视图名1,视图名2;

查看视图结构： DESC 视图名; 或 SHOW CREATE VIEW 视图名;

数据库查询语言（DQL）

别名写法：

字段名|表名 as 别名

字段名|表名 别名

select：查询

SELECT @@ROWCOUNT 返回上个sql受影响的行数

查看表索引： SHOW INDEX FROM 表名;

distinct：去重

order by 字段名：排序

将查询结果根据字段值进行排序

asc升序

desc降序

group by 字段名：分组

将表中数据根据字段值进行分组

having子句：通过指定条件过滤不符合条件的分组（和group by一起使用）

where 条件：根据条件过滤

条件中不能包含聚合函数，但能包含带聚合函数的子查询

case when then end:多个条件判断

case

when 条件判断 then 结果值

when 条件判断 then 结果值

else 结果值

as 最后结果的别名

end

筛选条件

+号会将字符运算转为数值运算，无法转换为数字则转为0，拼接字符串要使用concat()

条件判断符：> 、< 、 = 、!= 、<> 、>= 、<=、<=>

逻辑判断符：AND 、OR 、NOT 、&& 、|| 、!

模糊查询：LIKE 、BETWEEN AND 、NOT BETWEEN AND、IN 、IS NULL

可以使用 != && || ! <=>但不推荐，可读性差

= 只能判断普通内容

is 只能判断null 、note null

null和任何比较都返回false

例：SELECT \* FROM staff WHERE test <> 'test9' test字段为null的不会被查出来(因为判断结果false) 所以需要另外处理null的情况

修改后：SELECT \* FROM staff WHERE test <> 'test9' or test != null

联合查询

union ：将多条查询语句的结果合并成一张表（类似or，可读性高）

列名和列数要一致，默认自动去重

union all不去重

聚合函数：不能和where一起使用

count(\*) 、count(1) ：统计表中记录数

count(字段值)：统计特定字段的记录数(会忽略null值)

ucase(字段值)：转为大写

datediff（日期1, 日期2)：日期1 - 日期2

Charindex() 返回查找字符串的索引，sql索引从1开始，无则返回0，CHARINDEX(查找字符，被查字符，开始位置)

SubString() 截取字符串中的一部分字符。SUBSTRING（字符串，开始位置，截取长度）

Stuff() 将指定长度的字符用新串替代。STUFF(字符串,开始位置,长度,替代字符串)

Soundex() 返回表示字符串声音的一串字符

coalesce(参数1，参数2，参数3...):返回第一个不是null的参数

子查询

where (子查询语句) + 判断 ：单行单列（将结果作为值进行条件判断）  
 from (子查询语句) ：多行多列（将结果作为临时表供外层sql使用）  
 where 字段值in (子查询语句) ：多行单列（将结果作为一个范围）  
 exists（子查询语句） ：子查询结果返回true,则执行外层sql,否则不执行

where 字段值 > any (子查询语句) ：子查询结果中有一条数据满足条件即条件成立(true)  
 where 字段值 > all (子查询语句) ：子查询结果中有所有数据满足条件即条件成立(true)  
 比较运算符子查询：子查询中可以使用的比较运算符如 “>” “<” “= ” “!=”

some：是any的别称(很少用)

多表连接

内连接：inner join | join 左连接和右连接结果的交集

外连接：内连接结果+主表中没有的字段

左连接：left join 左表为主表

右连接：right join 右表为主表

全连接：full join 左连接和右连接结果的并集，mysql中没有 full join，可以使用union来完成

交叉连接：cross join 笛卡尔积，不要使用

级联查询

一对一查询

一对多查询

多对多查询

limit [起始索引], 数据个数：分页查询

省略起始索引，则从0开始

分页查询公式：limit (page-1)\*size size

数据操作语言（DML）

insert：插入数据

insert into 表名 values(字段值1 , 字段值2) 不写字段顺序默认是创建表时的顺序

insert into 表名(字段名1, 字段名2) values(字段值1, 字段值2)

delete：删除数据

删除表中符合条件的数据：DELETE FROM 商品 WHERE 价格>3000 删除记录

自增列保持不动

支持回滚

trustcate：删除表中所有数据

删除表中所有数据：trustcate 表名

自增列从1重新开始

不支持回滚

效率比delete高

update：更新数据

修改数据：UPDATE 表名 SET 字段值=字段名

查询表中符合条件的字段，插入到另一个表中： SELECT 字段名1, 字段名2, ... INTO 表名1 from 表名2

事务处理语言（TPL）

SQL中显式事务和隐式事物的区别：

隐式事务：在连接的基础上，每一条语句都以单独事务运行。

显式事务：存在开始/结束标志，例如:  
BEGIN TRANSACTION  
...具体事务

END TRANSACTION。

数据控制语言（DCL）

现授予用户USER1在学生表上的SELECT权限

grant to ：授予权限(select 、delete 、update的权限)

授予select权限：USE 数据库名 GO GRANT SELECT ON 表名 TO 用户名

revoke from：回收权限(select 、delete 、update的权限)

回收select权限：USE 数据库名 GO REVOKE SELECT ON 表名 FROM 用户名

其他

查询每门课成绩都>=80分的学生

思路1：查询 not in 分数<80分的学生 并去重

select distinct name from table where name not in( select name from t\_student where score <= 80)

思路2：查询 根据姓名分组 组中分数最小值>=80

select name from table group by name having min(score)>=80;

触发器

SQL server的触发器类型

AFTER触发器：要求执行某一操作INSERT、UPDATE、DELETE之后才能够触发。

INSTEAD OF触发器：表示并不执行其所定义的操作INSERT、UPDATE、DELETE，而仅仅是触发器本身，既可以是表上定义的INSTEAD OF触发器，或者是视图是定义的INSTEAD OF 触发器

sql基础

DQL

执行顺序（按照执行顺序，要设了别名才能用）

from，join on，where，group by，having，select，order by，limit

常见函数

SELECT 函数名(实参列表) [FROM 表名]

sql中索引从1开始，函数可以嵌套调用

单行函数：concat()、IFNULL()等

字符函数：LENGTH()、CONCAT()、CONCAT()、SUBSTR()、SUBSTR()、INSTR()、TRIM()、LPAD()、REPLACE()

数学函数：SELECT ROUND()、SELECT CEIL()、SELECT FLOOR()、SELECT TRUNCATE()、SELECT MOD()、SELECT RAND()

日期函数：NOW()、CURDATE()、CURTIME()、year()、month()、day()、hour()、minute()、second() STR\_TO\_DATE()、DATE\_FORMAT()、DATEDIFF()

其他函数：VERSION()、DATABASE()、USER()

流程控制函数：IF CASE

分组函数（统计函数，聚合函数）：sum()、avg()、max()、min()、count()

1.sum和avg只处理数值，max,min,count可以处理任何类型

2.分组函数都忽略Null值

3.可以和distinct搭配去重

4.count(\*)/count(1)统计行数

5.和分组函数一同查询的字段 必须是group by的字段

联合查询

union ：将多条查询语句的结果合并成一张表（类似or，可读性高）

列名和列数要一致，默认自动去重

union all不去重

视图：是一张虚拟表，可以普通表一样使用

以下视图不允许更新

包含分组函数、distinct、group by、having、union、union all

常量视图

select中包含子查询

join

from 一个不能更新的视图

where子句的子查询引用了from子句中的表

级联删除的外键

alter table 表名 add constraint 约束名 foreign key(字段名) references 主表名(字段名) on delete caseade;

级联制空

alter table 表名 add constraint 约束名 foreign key(字段名) references 主表名(字段名) on delete set null;

需要建索引的情况

主键自动建立唯一索引

查询条件中频繁出现的字段，要建立索引

外键关系，要建立索引

单值/组合索引选择：高并发下倾向于组合索引

查询需要排序的字段建立索引，会提高索引排序速度，因为索引就已经排序了

查询中统计/分组的字段,要建立索引

左连接 索引加右边，因为左表一定全都有，会全表扫描  
 右链接 索引加左边，同理

索引优化  
 join语句优化  
 尽量减少join语句中nestedloop的循环总次数，用小结果集驱动大结果集  
 优先优化nestedloop的内层循环  
 保证join语句被驱动表的join条件字段被索引  
 无法保证被驱动表的join条件字段被索引且内存资源充足时，不要吝啬joinBuffer的设置（缓存）

索引建议

对于单值索引，尽量选择针对当前query过滤性更好的索引

对于复合索引，当前query中过滤性最好的字段放在索引靠前的位置，尽量选择能够包含当前query中的where字段中更多字段的索引

尽可能通过分析统计信息和调整query的写法来达到选择合适索引的目的

永远小表驱动大表：条件既能放where也能放having，那么优先放where，(小表驱动大表)

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

等价于(B表数据小于A表时适用,in)

for selelct id from B

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

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

等价于（B表数据大于A表时适用,exists,替代 in）

for selelct \* from A

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

order by子句的排序优化

避免使用selelct \*

尽量使用index方式排序，避免使用fileSort方式排序

尽量再索引列上完成排序操作，遵循最佳左前缀

如果进行filesort，会有单路排序/双路排序

增大sort\_buffer\_size参数，

增大 max\_length\_for\_sort\_data参数，提高排序效率

group by子句的优化

先排序后分组，会产生临时表

order by中的优化都能用在group by中

where 高于having,能在where中限定的条件就不要去写having

sql优化

临时表

当需要重复引用大型表或常用表中的某个数据集时,创建临时表很有效，但要避免频繁创建和删除临时表，减少系统表资源的消耗

新建临时表时，如果一次性插入数据量很大，那么可以使用 select into 代替 create table，

使用临时表，在存储过程的最后务必将所有的临时表显式删除，先 truncate table ，然后 drop table ，这样可以避免系统表的较长时间锁定

其他

尽量避免大事务操作，提高系统并发能力。

尽量避免向客户端返回大数据量，若数据量过大，应该考虑相应需求是否合理

In太多改为临时表join查询

-- 创建临时表

CREATE TEMPORARY TABLE temp\_customer\_ids (customer\_id INT);

-- 插入值

INSERT INTO temp\_customer\_ids (customer\_id) VALUES (1), (2), (3), ..., (10000);

-- 查询

SELECT o.\*

FROM orders o

JOIN temp\_customer\_ids t ON o.customer\_id = t.customer\_id;

-- 删除临时表

DROP TABLE temp\_customer\_ids;

@RequestBody使用jackson反序列化生成对象

Jackson反序列化只会调用无参构造+字段setter,不会涉及static字段，

jackson反序列化步骤

加载类， 分配内存，解析json

执行无参构造

赋默认值，执行代码块

按照key找stter方法，通过反射赋值

New一个对象的过程

是否加载过类(需要加载类，生成静态变量，执行静态代码块)

分配内存

初始化默认值

执行构造函数

使用==比较枚举，枚举常量的引用地址相同即枚举相同

Objects.equals来比较Integer，实际上是调用了Integer.equals来比较值是否相等

拆包时可能存在空指针异常，如if(Integer==int) 可能报空指针，建议用Object.equals

反射为什么慢

反射是在运行期解析类名，找构造函数，访问权限，找setter方法等，每次生成对象都要解析一遍，这些在new的时候都是编译器处理的，所以比较慢

线程池中，长时间的任务占满了线程，导致短时间的任务在排队，用了很长时间，如何优化

1. 引入优先级队列

2.使用两个不同的线程池，处理使短时间任务和长时间任务不冲突