Redis

**3.1:考虑 用户基数 1亿,日活用户 1kw的 一个APP系统,如果要开发签到功能,要求支持大并发,如何存储签到数据最节省空间?请测算如果采用你给方案,1个月后 需要多少空间存储这些数据?(用户Id Long型数据,均匀分布在 0-400000000之间,日活用户30%签到),使用哪些命令存储签到数据, 获取 某用户的某些天是否签到 使用什么命令?**

当用户点击签到按钮，在redis数据库中使用hash结构进行存储签到数据；

获取到用户信息得到用户的id；

创建当前date转化为 2020-4-29这种字符串格式；

key存储用户标识（userId+当天日期date） value里存储签到数据（用户id值，签到数值已签到为“1”); hset(user.getId()+date,”id”,user.getId());  
hset(user.getId()+date,**"signIn"**,1);

基数 1亿,日活用户 1kw，日活用户30%签到

300W人数据签到

一个hash key为大概占40个字节左右 用户id键名加键值为16bt sigiIn字段键名加键值大概10bt 总共大概为80bt左右

需要得存储空间为： 80\*3000000\*30/1024/1024

大概为 250MB

获取数据：

判断想知道用户哪些天是否签到;

系统得到用户id信息 如果想知道该用户2020-4-29是否签到；

hget获取key(用户标识Id加日期，）字段sigiIn所对应得值 是否为1

Hget(user.getId()+2020-4-29,signIn)

得到数据是否为1 为1则用户该日期已经签到

不存在则表示用户该日期没有进行签到。

**3.2:如果要用Redis List 做1:先入先出队列;2:后入先出队列,应该使用那几个命令?**

一般使用list结构做队列,

先入先出：lpush生产消息，rpop消费消息

先入后出: lpush生产消息 lpop消费消息

**3.3:请使用Redis 做数据库 (Jedis做为Driver) 平配合Servlet /JSP/JSTL 做一个学生数据管理功能.**

A:学生数据包含:

Id 字符串类型长度 40,

姓名 (name)字符串类型长度 40,

出生日期(birthday) 日期类型,

备注 (description)字符串类型长度 255,

平均分(avgscore) 整数类型,

B:功能:

按照平均分倒序分页(每页10条记录),展示学生数据所有字段,每行有 修改和删除 链接;

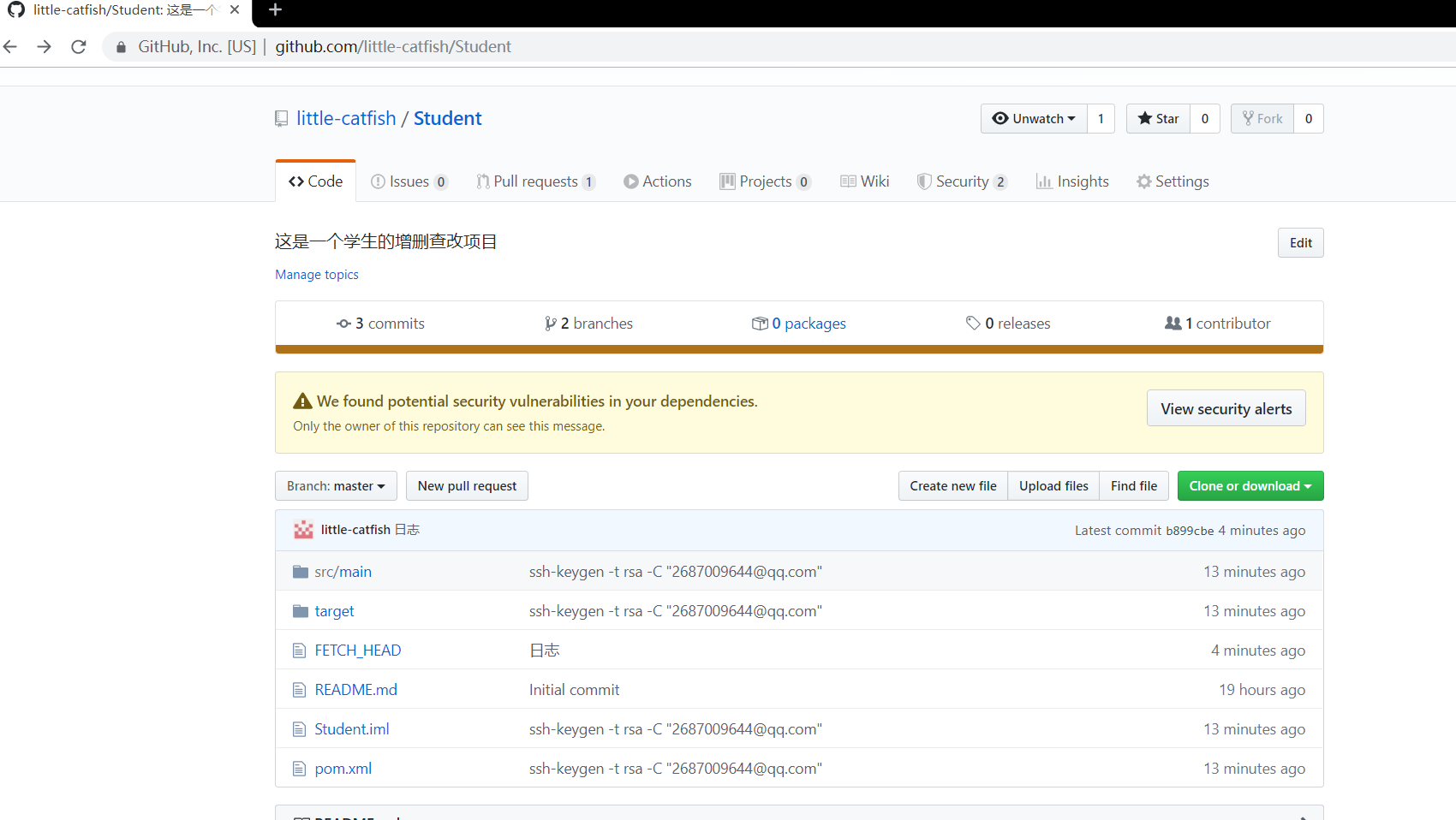
点击修改链接能够修改该学生数据;

点击删除链接能删除一个学生数据;

列表页下方有分页链接,点击能够跳转到相应的页数;

列表页上方有新增链接,点击录入一个学生数据.

此题代码提交到github ,要求使用maven



将你们的应用部署到购买的阿里云服务器,并给出访问地址

<http://114.55.36.191:8080/student>

**做的功能有：**

平均分倒序分页查询

添加学生信息

修改学习信息以及修改学生信息时候，学生信息回显在界面

删除学生信息以及一次性删除选中学生信息

**界面截图：**



Mysql

**4.1 列举Mysql支持的数据类型和最大值或者长度.**

TINYINT       (0，255)            小整数值

SMALLINT      (0，65 535)         大整数值

MEDIUMINT     (0，16 777 215)      大整数值

INT或INTEGER   (0，4 294 967 295) 大整数值

BIGINT (0，18 446 744 073 709 551 615) 极大整数值

FLOAT   0，(1.175 494 351 E-38，3.402 823 466 E+38) 单精度浮点数值

DOUBLE   0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308) 双精度浮点数值

CHAR         0-255字节          定长字符串

  VARCHAR      0-255字节          变长字符串

  TINYBLOB     0-255字节        不超过 255 个字符的二进制字符串

    TINYTEXT     0-255字节        短文本字符串

    BLOB         0-65535字节      二进制形式的长文本数据

    TEXT         0-65535字节      长文本数据

    MEDIUMBLOB   0-16 777 215字节 二进制形式的中等长度文本数据

    MEDIUMTEXT   0-16 777 215字节 中等长度文本数据

    LOGNGBLOB    0-4 294 967 295字节 二进制形式的极大文本数据

    LONGTEXT     0-4 294 967 295字节 极大文本数据

    VARBINARY(M)                   允许长度0-M个字节的定长字节符串，值的长度+1个字节

    BINARY(M)    M                 允许长度0-M个字节的定长字节符串

DATE       4        1000-01-01/9999-12-31 YYYY-MM-DD    日期值

 TIME       3        '-838:59:59'/'838:59:59' HH:MM:SS    时间值或持续时间

 YEAR       1         1901/2155               YYYY       年份值

 DATETIME   8       1000-01-01 00:00:00/9999-12-31 23:59:59 YYYY-MM-DD HH:MM:SS 混合日期和时间值

 TIMESTAMP  4       1970-01-01 00:00:00/2037 年某时 YYYYMMDD HHMMSS 混合日期和时间值，时间戳

**4.2 列举Mysql 常用的 字符串函数,日期函数,聚合函数.**

**聚合函数名称 描述**

COUNT() 返回满足SELECT条件的记录总和数

SUM() 通常为数值字段或表达式列作统计，返回一列的总和

AVG() 通常为数值字段或表达式列作统计，返回一列的平均值

MIN() 可以为数值字段、字符字段或表达式列作统计，返回最大的值

MAX() 可以为数值字段、字符字段或表达式列作统计，返回最小的值

**字符串函数名称 描述**

SELECT CHAR\_LENGTH('狂神说坚持就能成功'); /\*返回字符串包含的字符数\*/

SELECT CONCAT('我','爱','程序'); /\*合并字符串,参数可以有多个\*/

SELECT INSERT('我爱编程helloworld',1,2,'超级热爱'); /\*替换字符串,从某个位置开始替换某个长度\*/

SELECT LOWER('KuangShen'); /\*小写\*/

SELECT UPPER('KuangShen'); /\*大写\*/

SELECT LEFT('hello,world',5); /\*从左边截取\*/

SELECT RIGHT('hello,world',5); /\*从右边截取\*/

SELECT REPLACE('狂神说坚持就能成功','坚持','努力'); /\*替换字符串\*/

SELECT SUBSTR('狂神说坚持就能成功',4,6); /\*截取字符串,开始和长度\*/

SELECT REVERSE('狂神说坚持就能成功'); /\*反转\*/

**日期函数 描述**

SELECT CURRENT\_DATE(); /\*获取当前日期\*/SELECT CURDATE(); /\*获取当前日期\*/

SELECT NOW(); /\*获取当前日期和时间

\*/SELECT LOCALTIME(); /\*获取当前日期和时间

\*/SELECT SYSDATE(); /\*获取当前日期和时间\*/

/\*获取年月日,时分秒\*/

SELECT YEAR(NOW());

SELECT MONTH(NOW());

SELECT DAY(NOW());

SELECT HOUR(NOW());

SELECT MINUTE(NOW());

SELECT SECOND(NOW());

**4.3 判断某列为 NULL或者不为NULL的写法,**

是空为 is null，不为空则是 is not null

select \* from table where column is null;

select \* from table where column is not null;

**4.4 解释IF, IFNULL函数用法**

IFNULL函数：它接受两个参数，如果不是NULL，则返回第一个参数。 否则，IFNULL函数返回第二个参数。两个参数可以是文字值或表达式。

IF函数：如果要返回基于TRUE或FALSE条件的值，而不是NULL，则应使用IF函数。

**4.5 写出你知道的Show 命令**

1.显示mysql中所有数据库的名称.

|  |  |
| --- | --- |
| 1 | show databases; |

2.显示当前数据库中所有表的名称

|  |  |
| --- | --- |
| 1  2  3 | show tables;  show tables from database\_name; |

3.显示表中列名称

|  |  |  |
| --- | --- | --- |
| 1 | show columns from database\_name.table\_name; |  |

4.查看某MySQL用户的使用权限

|  |  |
| --- | --- |
| 1 | show grants for user\_name; |

5.显示create database 语句是否能够创建指定的数据库,并可以查看到创建库语句的SQL信息。

|  |  |
| --- | --- |
| 1 | show create database database\_name; |

6.显示create table 语句是否能够创建指定的数据表,并可以查看到表创建语句的SQL信息。

|  |  |
| --- | --- |
| 1 | show create table table\_name; |

7.显示安装以后可用的存储引擎和默认引擎。

|  |  |
| --- | --- |
| 1 | show engines; |

8.显示最后一个执行的语句所产生的错误、警告和通知

|  |  |
| --- | --- |
| 1 | show warnings; |

9.只显示最后一个执行语句所产生的错误

|  |  |
| --- | --- |
| 1 | show errors; |

10.显示系统中正在运行的所有进程，也就是当前正在执行的查询.

|  |  |
| --- | --- |
| 1 | show processlist\G |

**4.6数据导出导入命令**

### **导出数据库文件**

**常用命令**

|  |  |
| --- | --- |
| 1 | mysqldump -uroot -pMyPassword databaseName tableName1 tableName2 > /home/foo.sql |

mysqldump -u 用户名 -p 数据库名 数据表名 > 导出的文件名和路径

**导出整个数据库**

|  |  |
| --- | --- |
| 1 | mysqldump -u root -p databaseName > /home/test.sql   (输入后会让你输入进入MySQL的密码) |

**mysql导出数据库一个表，包括表结构和数据**  
mysqldump -u 用户名 -p 数据库名 表名> 导出的文件名和路径

|  |  |
| --- | --- |
| 1 | mysqldump -u root -p databaseName tableName1 > /home/table1.sql |

如果需要导出数据中多张表的结构及数据时，表名用空格隔开

|  |  |
| --- | --- |
| 1 | mysqldump -u root -p databaseName tableName01 tableName02 > /home/table.sql |

**仅导出数据库结构**

|  |  |
| --- | --- |
| 1 | mysqldump -uroot -pPassWord -d databaseName > /home/database.sql |

**仅导出表结构**

|  |  |
| --- | --- |
| 1 | mysqldump -uroot -pPassWord -d databaseName tableName > /home/table.sql |

### **将语句查询出来的结果导出为.txt文件**

|  |  |
| --- | --- |
| 1 | mysql -uroot -pPassword database1 -e "select \* from table1" > /home/data.txt |

### **数据导入**

常用source 命令   
进入mysql数据库控制台，mysql -u root -p   
mysql>use 数据库   
使用source命令，后面参数为脚本文件(.sql)

|  |  |
| --- | --- |
| 1 | mysql>source /home/table.sq |

**4.7 写SQL建立数据库,建立学生数据表student (使用第三题中描述的字段),**

CREATE DATABASE Student;

CREATE TABLE student(

id VARCHAR(40),

NAME VARCHAR(40),

birthday DATE,

description VARCHAR(225),

avgscore INT

);

**4.8 写SQL修改student表增加一列 入学年份(字段名为 :clazz) 类型 int**

ALTER TABLE student ADD clazz INT;

**4.9 Explain 命令用法和 输出的解释**

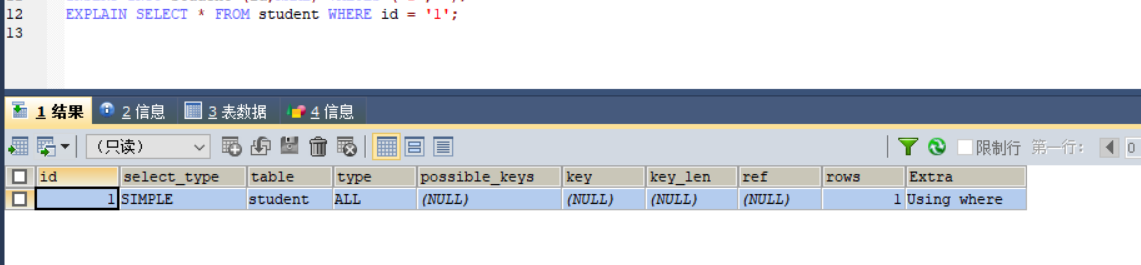
在student 表插入一条数据后

insert into student (id,name) values ('1','');

执行如下sql:

explain select \* from student where id = '1'\G

请解释结果表格含义(主要是type字段)

**结果为：**



SIMPLE:简单SELECT(不使用UNION或子查询)

Table 输出的行所引用的表

ALL:对于每个来自于先前的表的行组合,进行完整的表扫描。

|  |  |
| --- | --- |
| ****possible\_keys**** | 指出MySQL能使用哪个索引在该表中找到行 |

|  |  |
| --- | --- |
| ****key**** | 显示MySQL实际决定使用的键(索引)。如果没有选择索引,键是NULL。 |

|  |  |
| --- | --- |
| ****key\_len**** | 显示MySQL决定使用的键长度。如果键是NULL,则长度为NULL。 |
| ****ref**** | 显示使用哪个列或常数与key一起从表中选择行。 |
|  |  |

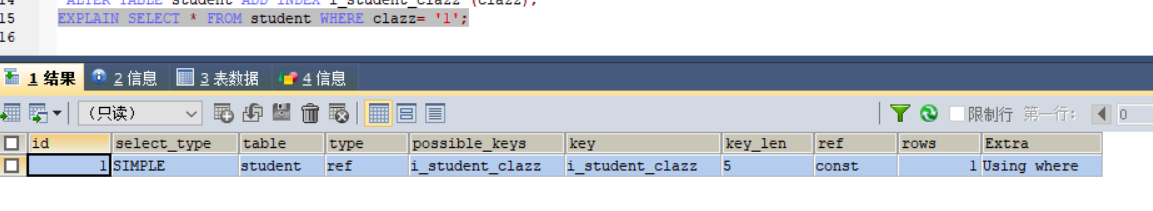
|  |  |
| --- | --- |
| ****rows**** | 显示MySQL认为它执行查询时必须检查的行数。多行之间的数据相乘可以估算要处理的行数。 |

Using where:WHERE 子句用于限制哪一个行匹配下一个表或发送到客户。

alter table student add index i\_student\_clazz (clazz);

再执行 explain select \* from student where clazz= '1'\G

再解释结果表格含义

**结果为：**



SIMPLE:简单SELECT(不使用UNION或子查询)

Table 输出的行所引用的表

ref:对于每个来自于前面的表的行组合,所有有匹配索引值的行将从这张表中读取。

|  |  |
| --- | --- |
| ****possible\_keys**** | 指出MySQL能使用哪个索引在该表中找到行 |

|  |  |
| --- | --- |
| ****key**** | 显示MySQL实际决定使用的键(索引)。如果没有选择索引,键是NULL。 |

|  |  |  |
| --- | --- | --- |
| ****key\_len**** | 显示MySQL决定使用的键长度。如果键是NULL,则长度为NULL。 | |
| ****ref**** | | 显示使用哪个列或常数与key一起从表中选择行。 |
| ****rows**** | 显示MySQL认为它执行查询时必须检查的行数。多行之间的数据相乘可以估算要处理的行数。 | |

Using where:WHERE 子句用于限制哪一个行匹配下一个表或发送到客户。

**4.10根据你所学知识,现已知student表记录行数超过1kw,为了加快如下对表student的查询sql 应如何建立索引:**

select \* from student where clazz <1990 and birthday > '2009-01-01'

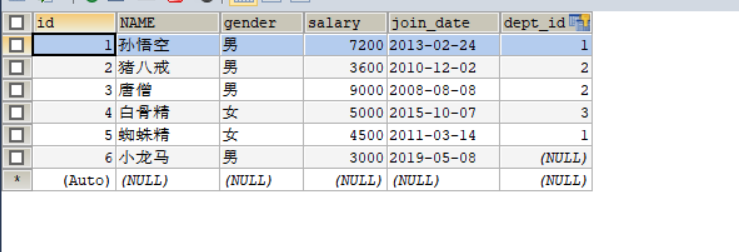
select \* from student where birthday > '2009-01-01'

在student表中clazz字段和birthday字段建立索引

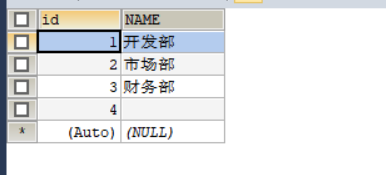
alter table student add index i\_student\_clazz (clazz)

alter table student add index i\_student\_birthday (birthday)

**4.11 请用SQL 示例 解释left join ,right join ,inner join 并截图 验证你执行的结果.**

emp表：

dept表：

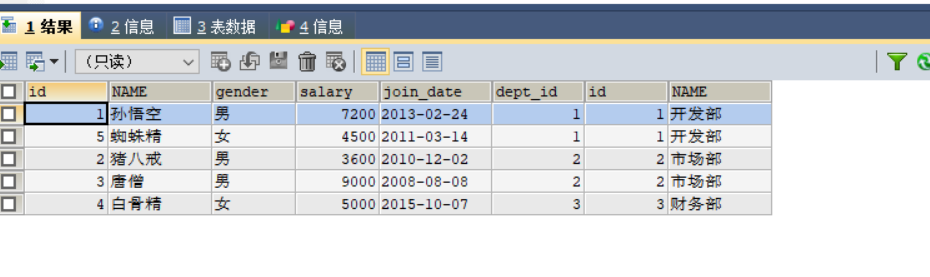


显式内连接：inner join

\* 语法： select 字段列表 from 表名1 [inner] join 表名2 on 条件

SELECT \* FROM emp INNER JOIN dept ON emp.`dept\_id` = dept.`id`;

只会查询俩个表有关联关系的数据，当关联关系字段为空时，则查询不到；

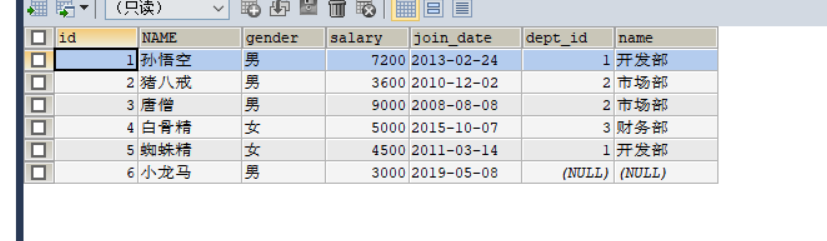


左外连接：

\* 语法：select 字段列表 from 表1 left [outer] join 表2 on 条件；

SELECT t1.\*,t2.`name` FROM emp t1 LEFT JOIN dept t2 ON t1.`dept\_id` = t2.`id`;

\* 查询的是左表所有数据以及其交集部分。



右外连接：

\* 语法：select 字段列表 from 表1 right [outer] join 表2 on 条件；

SELECT \* FROM dept t2 RIGHT JOIN emp t1 ON t1.`dept\_id` = t2.`id`;

\* 查询的是右表所有数据以及其交集部分。

