实验报告

实验完成情况

完成了对 long， timestamp，decimal，date四个类型的测试，测试结果如下

```
filelen: 504
fileTailOffset: 416
filelen: 504
fileTailOffset: 416
```

a int64	b int64
1	1509095563
2	1262399562
3	1654921478
4	185012677
5	2042332856
6	622819451
7	844820549
8	666477209
9	1738431465
10	1973293514
10 rows	2 columns

```
filelen: 229
fileTailOffset: 165
filelen: 229
fileTailOffset: 165
```

a date
1997-10-03
2025-09-16
1997-02-23
2019-03-28
2002-09-27
2019-01-02
1993-03-08
2014-04-29
2006-01-02
2017-10-06
10 rows

```
D SELECT * FROM "/home/postgres/pixels/data/output/1737355132.pxl";
PIXELS_SRC is /home/postgres/mini-pixels
PIXELS_HOME is /home/postgres/mini-pixels
pixels properties file is /home/postgres/mini-pixels/pixels-cxx.properties
filelen: 235
fileTailOffset: 165
filelen: 235
fileTailOffset: 165
```

a timestamp
2004-10-14 04:32:47
2001-02-10 01:07:47
2004-08-13 08:33:04
2011-01-11 02:24:51
2023-06-26 00:54:27
2014-10-26 03:03:38
2010-03-09 04:49:53
2010-09-22 04:39:34
2011-06-08 04:49:51
2004-12-29 00:43:17
10 rows

```
pixels properties file is /home/postgres/mini-pixels/pixels-cxx.properties
filelen: 372
fileTailOffset: 290
filelen: 372
fileTailOffset: 290
```

a decimal(10,2)
34476661.13
9581641.68
97486758.64
2451781.32
330310.41
28381028.71
36915791.60
78169018.65
25024491.12
64608051.20
75169058.60
32558787.76
59501512.29
51011771.52
47060971.30
15 rows

其中，date遇到了Segmentation fault的问题，迫于时间问题，截至目前还未找到问题来源（可能是vector free的问题），bug主要是当文件行数设置过大时会出现segmentation fault，但当n<=10时可以正常读入。

实验完成细节

Date

考虑到Date的底层存储为int，我们可以参考int部分的代码，首先在构造函数中为整型指针分配内存，以存储日期的天数。然后在add(std::string &val)函数中，通过解析字符串形式的“yyyy-mm-dd”，计算与1970-01-01的天数差并存储到dates数组。add(bool)和add(int)则用于根据不同类型将数据写入数组。ensureSize会在需要时成倍扩容并复制已存在的数据。close和析构函数中释放内存。最后在set函数中根据索引更新days值并维护isNull信息 。

Timestamp

考虑到Date的底层存储为long,同理date，我们可以参考long部分的代码，首先在构造函数中根据编码级别决定是否使用运行长度编码。在write函数中，将数据按像素步长写入，若超过步长则调用newPixel切换到下一个像素。writeCurPartTimestamp函数根据isNull判断值是否为0。newPixel函数在开启运行长度编码时使用encoder对数据进行编码，否则直接写入。最后通过close释放资源，getColumnChunkEncoding返回相应的编码类型。

Decimal

如果完整实现需要考虑正负数和四舍五入，需要在vector文件中手动实现，实现思路是：add(std::string &val)函数先解析字符形式的整数部分和小数部分，再根据scale进行放大或截断，并在需要时执行正负方向的四舍五入。确保数组长度不足时会自动扩容并复制已有数据。

DecimalColumnWriter 在将 long 值写入输出流时，没有对正负值的特殊区分，直接从 DecimalColumnVector 读取 long 型整数并根据字节序及 nullsPadding 设置将数据写出。

实验总结

从补全代码角度来看，由于有已经实现好的代码可以参考，所以难度并不大，主要花费时间在理解lab框架，知道每一步在干什么。

总的来说还是非常不错的lab，特别是锻炼了debug（尤其是gdb能力），在此特别感谢白锦峄同学和沈海涛同学对我的帮助，提供了很多debug方向的建议。

仓库连接