友乾营

专注数据技术的社群

# 多层科目任意组合汇总表的性能优化方案

20 期

www.raqsoft.com.cn/yqy

# 一张资产负债表引发的难题?

友乾营

## 指标计算多

每个单元格都是独立计算的指标，往往这样的计算指标多达上百个!

## 样式复杂

典型的中国式复杂报表格式（不规则的EXCEL表格）

| 查询条件 | |
|---|---|
| | 会计年度（必输） |
| | 公司代码（非必输） |
| | 利润中心（非必输） |
| | 过账期间（必输） |

### 资产负债表

| | | | | | |
|---|---|---|---|---|---|
| universe名称: | ZFIC | ZFIGL_M10_Q006_UN01 | 查询条件: | | |
| query名称: | ZFIG | ZFIGL_M10_Q006 | 会计年度（必输） | | |
| 模型名称: | ZFIG | 总账余额 | 过账期间（必输） | | |
| 编制单位: | | | 期间: 年月 | | |
| 项目 | 行次 | 期末余额 | 项目 | 行次 | 期末余额 |
| 流动资产: | 1 | ----- | 流动负债: | 73 | ----- |
| 货币资金 | 2 | =Sum([查询 1].[期末余额]) Where ( Left([科 | 短期借款 | 74 | =-Sum([查询 H101].[期末余额]) Where ( |
| △结算备付金 | 3 | ----- | △向中央银行借款 | 75 | ----- |
| △拆出资金 | 4 | ----- | △吸收存款及同业存放 | 76 | ----- |
| 以公允价值计量且其变动计 | 5 | =Sum([查询 1].[期末余额]) Where ( Left([科 | △拆入资金 | 77 | ----- |
| 衍生金融资产 | 6 | | 以公允价值计量且其变动计入当期 | 78 | |
| 应收票据 | 7 | =Sum([查询 1].[期末余额]) Where ( Left([科 | 衍生金融负债 | 79 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| 应收账款 | 8 | =Left([科目];8) = "12310201" Or Left([科 | 应付票据 | 80 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| 合同资产 | 9 | =Sum([查询 1].[期末余额]) Where ( Left([科 | 应付账款 | 81 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| 预付款项 | 10 | =Sum([查询 1].[期末余额]) Where ( Left([科 | 预收账款 | 82 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| △应收保费 | 11 | ----- | △卖出回购金融资产款 | 83 | ----- |
| △应收分保账款 | 12 | ----- | △应付手续费及佣金 | 84 | ----- |
| △应收分保合同准备金 | 13 | ----- | 应付职工薪酬 | 85 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| 应收利息 | 14 | =Sum([查询 1].[期末余额]) Where ( Left([科 | 其中: 应付工资 | 86 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| 应收股利 | 15 | =Sum([查询 1].[期末余额]) Where ( Left([科 | 应付福利费 | 87 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| 其他应收款 | 16 | =Sum([查询 1].[期末余额]) Where ( Left([科 | #其中: 职工奖励及福 | 88 | |
| △买入返售金融资产 | 17 | ----- | 应交税费 | 89 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| 存货 | 18 | =Sum([查询 1].[期末余额]) Where ( Left([科 | 其中: 应交税金 | 90 | ----- |
| 其中: 原材料 | 19 | =Sum([查询 1].[期末余额]) Where (Left([科 | 应付利息 | 91 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| 库存商品(产成品) | 20 | =Sum([查询 1].[期末余额]) Where ( Left([科 | 应付股利 | 92 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| 划分为持有待售的资产 | 21 | | 其他应付款 | 93 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| 一年内到期的非流动资产 | 22 | =Sum([查询 1].[期末余额]) Where ( Left([科 | △应付分保账款 | 94 | ----- |
| 其他流动资产 | 23 | =Sum([查询 1].[期末余额]) Where ( Left([科 | △保险合同准备金 | 95 | ----- |
| 流动资产合计 | 24 | =[流动资产合计－期末] | △代理买卖证券款 | 96 | ----- |
| 非流动资产: | 25 | ----- | △代理承销证券款 | 97 | ----- |
| △发放贷款及垫款 | 26 | ----- | 划分为持有待售的负债 | 98 | |
| 可供出售金融资产 | 27 | =Sum([查询 1].[期末余额]) Where ( Left([科 | 一年内到期的非流动负债 | 99 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |
| 持有至到期投资 | 28 | =Sum([查询 1].[期末余额]) Where ( Left([科 | 其他流动负债 | 100 | =-Sum([查询 1].[期末余额]) Where ( Left([科 |

# 源数据表结构与单个指标的计算逻辑

友乾营

**在源数据表中有一个字段称为科目，其长度总是固定的10位，科目字段的值实际上是一个分层的代码，而报表里上百个指标就是根据需求对不同层次科目数据的统计结果，具体做法是通过截取科目的前几位来确定层次，然后按需求自由组合，作为条件进行过滤，最后对金额字段进行累计汇总。**

长度固定为10位

**源数据表**

| Index | 年 | 月 | 科目 | 金额 |
|---|---|---|---|---|
| 6500 | 2017 | 2 | 7854140013 | -7302.0 |
| 6501 | 2017 | 2 | 5421070013 | -100.0 |
| 6502 | 2017 | 2 | 7854140013 | -200.0 |
| 6503 | 2017 | 2 | 5414071413 | 100.0 |
| 6504 | 2017 | 2 | 5414070713 | [illegible] |
| 6505 | 2017 | 2 | 7807098118 | [illegible] |
| 6506 | 2017 | 2 | 7807098111 | [illegible] |
| 6507 | 2017 | 2 | 7854140013 | [illegible] |
| 6508 | 2017 | 2 | 7854140013 | [illegible] |
| 6509 | 2017 | 2 | 5547630811 | 14.53 |
| 6510 | 2017 | 2 | 5414071413 | -30.0 |
| 6511 | 2017 | 2 | 5414141413 | -50000.0 |
| 6512 | 2017 | 2 | 5414071413 | -11.0 |
| 6513 | 2017 | 2 | 5414070713 | 97.17 |
| 6514 | 2017 | 2 | 8547350713 | -7.0 |
| 6515 | 2017 | 2 | 7854140013 | 1000.0 |

**6000万**

指标A对应的科目列表是[1001，1002]，代表累计所有前4位是1001、1002的科目，用SQL写出来就是：select sum(金额) from T1 where concat(年,月)<=? and (left(科目,4)="1001" or left(科目,4)="1002")

**报表片段**

| 项目 | 行次 | 期末余额 |
|---|---|---|
| 流动资产： | 1 | ----- |
| 货币资金 | 2 | =Sum([查询 1].[期末余额]) Where ( Left([科 |
| △结算备付金 | 3 | ----- |
| △拆出资金 | 4 | ----- |
| 以公允价值计量且其变动计 | 5 | =Sum([查询 1].[期末余额]) Where ( Left([科 |
| 衍生金融资产 | 6 | |
| 应收票据 | 7 | =Sum([查询 1].[期末余额]) Where ( Left([科 |
| 应收账款 | 8 | =Left([科目];8) = "12310201" Or Left([科 |
| 合同资产 | 9 | =Sum([查询 1].[期末余额]) Where ( Left([科 |
| 预付款项 | 10 | =Sum([查询 1].[期末余额]) Where ( Left([科 |
| △应收保费 | 11 | ----- |
| △应收分保账款 | 12 | ----- |
| △应收分保合同准备金 | 13 | ----- |
| 应收利息 | 14 | =Sum([查询 1].[期末余额]) Where ( Left([科 |
| 应收股利 | 15 | =Sum([查询 1].[期末余额]) Where ( Left([科 |
| 其他应收款 | 16 | =Sum([查询 1].[期末余额]) Where ( Left([科 |
| △买入返售金融资产 | 17 | ----- |
| 存货 | 18 | =Sum([查询 1].[期末余额]) Where ( Left([科 |
| 其中：原材料 | 19 | =Sum([查询 1].[期末余额]) Where (Left([科 |
| 库存商品(产成品) | 20 | =Sum([查询 1].[期末余额]) Where ( Left([科 |
| 划分为持有待售的资产 | 21 | |

指标B对应的科目为[2702,153102,12310105], 那就代表对前4位是2702、前6位是153102、前8位是12310105的所有科目值进行累计，用SQL写出来就是：select sum(金额) from T1 where concat(年,月)<=? and (left(科目,4)="2702" or left(科目,6)="153102" or left(科目,8)="12310105")

**每个指标对应的科目数量不定，可多达10个以上，各个科目的层次也不尽相同。报表工具原则上并不难实现这类格式复杂、指标参数任意组合的需求，只是原始数据量大，查询就会非常慢，用户体验差。**

# 多次遍历方案

友乾营

**最常见的开发思路，对报表的每个指标都写一句完整的SQL来计算，有100个指标，就写100个SQL。**

**有些商用报表工具提供了函数，以润乾报表V5为例，可以直接在单元格中执行SQL (比如query/call等)，单元格的表达式大概会是这样：**

**单元格1**

=query("select sum(金额) from T1 where concat(年,月)<=? and (left(科目,4)= ‘1001' or left(科目,4)= ‘1002’ ” , concat(year,month))

指标1

**...**

=query("select sum(金额) from T1 where concat(年,月)<=? and (left(科目,4)='2702' or left(科目,6)='153102' or left(科目,8)='12310105')", concat(year,month))

指标2

**单元格N**

=query("select sum(金额) from T1 where concat(年,月)<=? and (left(科目,4)='1121' or left(科目,8)='12310106' or left(科目,8)='12310206' or left(科目,8)='12310306' or left(科目,8)='12310406' or left(科目,8)='12310506' or left(科目,8)='12310606' or left(科目,8)='12310706' or left(科目,8)='12310806' or left(科目,8)='12319006' or left(科目,8)='12319106')", concat(year,month))

指标N

# 多次遍历方案

友乾营

**如果是非多源报表工具，以BIRT为例，则可以借助外部程序数据源来实现，比如可以直接用集算器编写以下SPL脚本：**


| | A |
|---|---|
| 1 | =connect("demo") |
| 2 | =A1.query@1("select sum(金额) from T1 where concat(年,月)<=? and (left(科目,4)='1001' or left(科目,4)='1002')",concat(year,month)) |
| 3 | =A1.query@1("select sum(金额) from T1 where concat(年,月)<=? and (left(科目,4)='2702' or left(科目,6)='153102' or left(科目,8)='12310105')",concat(year,month)) |
| ... | ... |
| 102 | >A1.close() |
| 103 | return [A2:A101] |

**[缺陷]无论直接在格中使用SQL还是在程序数据源中计算，实际上每计算一个指标就得遍历一次源数据；而每个指标还对应多个需要AND的条件，这些都会严重降低性能！当大数据量的情况下时，查询几乎就不能使用了！**

# 一次遍历方案

友乾营

“多次遍历方案”的问题在于，无论如何，对源数据遍历100次实在是太低效了，那么，我们有没有办法只做一次遍历就把所有指标都计算出来呢?

这种一次遍历的思路确实是可以的，我们只需要把SQL中的WHERE条件拼到SELECT中就行了，比如前面说到的多个指标，可以写成如下这样：

```
SELECT
    SUM(CASE WHEN (LEFT(科目,4)='1001' OR LEFT(科目,4)='1002’ )
      THEN 金额 ELSE 0 END) 指标A,
    SUM(CASE WHEN (LEFT(科目,4)="2702" OR LEFT(科目,6)="153102" OR
      LEFT(科目,8)="12310105") THEN 金额 ELSE 0 END) 指标B,
    SUM(CASE WHEN (LEFT(科目,4)="1121" OR LEFT(科目,8)="12310106" OR
      LEFT(科目,8)="12310206" OR LEFT(科目,8)="12310306" OR
      LEFT(科目,8)="12310406" OR LEFT(科目,8)="12310506" OR
      LEFT(科目,8)="12310606" OR LEFT(科目,8)="12310706" OR
      LEFT(科目,8)="12310806" OR LEFT(科目,8)="12319006" OR
      LEFT(科目,8)="12319106") THEN 金额 ELSE 0 END) 指标C,
    ...
FROM T1
WHERE CONCAT(年,月)<=?
```

SQL太长

可读性差

# 一次遍历方案—优化

友乾营

**真要用上面那个SQL的思路来处理100个指标，可见这个SQL会有多长，维护难度会有多大。为此，我们可以利用集算器的游标来实现这个逻辑，适当降低维护难度！**

**另外，我们注意到其中处理的都是不再变化的历史数据，那么我们就可以把数据先搬出数据库存成文件，然后用文件作为数据源，从而加快IO访问。**

1、把数据搬出数据库保存成文件

| | A |
|---|---|
| 1 | =connect("demo") |
| 2 | =A1.cursor("select 科目,年,月,金额 from T1") |
| 3 | =file("总账凭证-pre.btx") |
| 4 | >A3.export@b(A2) |
| 5 | >A1.close() |

2、遍历一次源数据，计算100个指标！

| | A | B | C |
|---|---|---|---|
| 1 | =file("总账凭证-pre.btx") | | /指标参数列 |
| 2 | =A1.cursor@b() | | |
| 3 | =A2.select(concat(年,月)<=concat(year,month)) | | |
| 4 | for A3 | =@+A4.select(C4.contain(科目\1000000)).sum(金额) | [1001,1002] |
| 5 | | =@+A4.select(C5.contain(科目\1000000)\|\|C5.contain(科目\10000)\|\|C5.contain(科目\100)).sum(金额) | [2702,153102,12310105] |
| ... | ... | ... | ... |
| 104 | return [B4:B103] | | |

**[优点]显然，这种计算逻辑和参数分离的写法，可读性强，极大地提高可维护性！**

# 预先汇总方案—数据预处理思路

友乾营

**如果能够把数据事先按科目汇总，那么我们就可以不必重复累加科目相等的记录了，而且存储量也会变少，IO也会更快。**

## 步骤一：分组汇总、并计算累计值

**明细记录**

**按照科目、年、月分组，统计本科目下当月的金额汇总值！**

**最终是要计算指标为“期末余额”，也就是截止某个日期的金额累计值！**

| 科目 | 年 | 月 | 金额 | … |
|---|---|---|---|---|
| 1234567890 | 2017 | 1 | 100 | … |
| 1234567890 | 2017 | 1 | 120 | … |
| 1234567890 | 2017 | 2 | 200 | … |
| 1234908765 | 2017 | 1 | 130 | … |
| 1234569087 | 2017 | 1 | 160 | … |
| 1234567892 | 2017 | 1 | 110 | … |
| … | … | … | … | … |

| 科目 | 年 | 月 | 金额 | … |
|---|---|---|---|---|
| 1234567890 | 2017 | 1 | 220 | … |
| 1234567890 | 2017 | 2 | 200 | … |
| 1234908765 | 2017 | 1 | 130 | … |
| 1234569087 | 2017 | 1 | 160 | … |
| 1234567892 | 2017 | 1 | 110 | … |
| … | … | … | … | … |

| 科目 | 年 | 月 | 累计金额 | … |
|---|---|---|---|---|
| 1234567890 | 2017 | 1 | 220 | … |
| 1234567890 | 2017 | 2 | 420 | … |
| 1234908765 | 2017 | 1 | 130 | … |
| 1234569087 | 2017 | 1 | 160 | … |
| 1234567892 | 2017 | 1 | 110 | … |
| … | … | … | … | … |

# 预先汇总方案—数据预处理思路

友乾营

**每个计算指标都是按照科目截取前4位、前6位、前8位等作为参数集合，那么在构造不同层次的科目号时，也需要和这种规则匹配，从而计算出不同层次的聚合值。**

## 步骤二：构造多层科目汇总值

**对于科目是1234567890，那么就需要新增科目1234/123456/12345678对应的汇总金额；其中科目1234会把所有1234开头的科目的金额值进行累计汇总，依次类推。**

**结果数据可直接被查询，每个指标的计算条件只要相等比较就可以，而不再需要截取、计算前几位。**

| 科目 | 年 | 月 | 累计金额 | … |
|---|---|---|---|---|
| 1234567890 | 2017 | 1 | 220 | … |
| 1234567890 | 2017 | 2 | 420 | … |
| 1234908765 | 2017 | 1 | 130 | … |
| 1234569087 | 2017 | 1 | 160 | … |
| 1234567892 | 2017 | 1 | 110 | … |
| … | … | … | … | … |

截取前8位

| 新增科目 | 年 | 月 | 累计金额 | … |
|---|---|---|---|---|
| 12345678 | 2017 | 1 | 330 | … |
| 12345678 | 2017 | 2 | 420 | … |
| … | … | … | … | … |

截取前6位

| 新增科目 | 年 | 月 | 累计金额 | … |
|---|---|---|---|---|
| 123456 | 2017 | 1 | 490 | … |
| 123456 | 2017 | 2 | 420 | … |
| … | … | … | … | … |

截取前4位

| 新增科目 | 年 | 月 | 累计金额 | … |
|---|---|---|---|---|
| 1234 | 2017 | 1 | 620 | … |
| 1234 | 2017 | 2 | 420 | … |
| … | … | … | … | … |

组合

| 新增科目 | 年 | 月 | 金额 | … |
|---|---|---|---|---|
| 12345678 | 2017 | 1 | 330 | … |
| 12345678 | 2017 | 2 | 420 | … |
| 123456 | 2017 | 1 | 490 | … |
| 123456 | 2017 | 2 | 420 | … |
| 1234 | 2017 | 1 | 620 | … |
| 1234 | 2017 | 2 | 420 | … |
| … | … | … | … | … |

**查询参数：[1234,1567]**

# 预先汇总方案—数据预处理思路—示例

友乾营

**步骤一：分组汇总、并计算累计值**

**集算器提供了跨行引用的语法，可以用A[-1]代表上一行的A，这样就可以计算：累计值=上一行的累计值+当前行值。**

| Index | 科目 | 年 | 月 | 金额 |
|---|---|---|---|---|
| 1 | 1207077013 | 2017 | 1 | 7374288.42 |
| 2 | 1207077013 | 2017 | 3 | 8.62996792... |
| 3 | 1207077013 | 2017 | 4 | 716500.0 |
| 4 | 1207077013 | 2017 | 5 | 927436.0 |
| 5 | 1207077013 | 2017 | 6 | 1787522.0 |
| 6 | 1207077013 | 2017 | 8 | 10000.0 |
| 7 | 1207077013 | 2017 | 9 | 2958654.42 |
| 8 | 1207077013 | 2017 | 10 | 9862702.28... |
| 9 | 1207077013 | 2017 | 11 | 1.55290101... |

| Index | 科目 | 年 | 月 | 金额 |
|---|---|---|---|---|
| 1 | 1207077013 | 2017 | 1 | 7374288.42 |
| 2 | 1207077013 | 2017 | 3 | 9.36739676... |
| 3 | 1207077013 | 2017 | 4 | 9.43904676... |
| 4 | 1207077013 | 2017 | 5 | 9.53179036... |
| 5 | 1207077013 | 2017 | 6 | 9.71054256... |
| 6 | 1207077013 | 2017 | 8 | 9.71154256... |
| 7 | 1207077013 | 2017 | 9 | 1.00074080... |
| 8 | 1207077013 | 2017 | 10 | 1.09936782... |
| 9 | 1207077013 | 2017 | 11 | 1.25465792... |

| | A | B |
|---|---|---|
| 1 | =file("总账凭证-pre.btx") | |
| 2 | =file("总账凭证-mid.btx") | |
| 3 | =A1.cursor@b() | /根据文件创建游标返回 |
| 4 | =A3.groupx(科目,年,月;sum(金额):金额 ) | |
| 5 | for A4;科目 | =A5.run(金额=金额[-1]+金额) |
| 6 | | >A2.export@ab(B5,#1:科目,#2:年,#3:月,#4:累计金额) |

# 预先汇总方案—数据预处理思路—示例

友乾营

## 步骤二：构造多层科目汇总值

全内存计算

| | A |
|---|---|
| 1 | =file("总账凭证-mid.btx") |
| 2 | =file("总账凭证-later.btx") |
| 3 | =A1.import@b() |
| 4 | =A3.groups((科目\100):科目,年,月;sum(累计金额):累计金额汇总) |
| 5 | =A4.groups((科目\100):科目,年,月;sum(累计金额汇总):累计金额汇总) |
| 6 | =A5.groups((科目\100):科目,年,月;sum(累计金额汇总):累计金额汇总) |
| 7 | =[A6,A5,A4].conj() |
| 8 | >A2.export@z(A7) |

| Index | 科目 | 年 | 月 | 累计金额汇总 |
|---|---|---|---|---|
| 1 | 12070714 | 2017 | 10 | 1230028.46 |
| 2 | 12070728 | 2017 | 4 | 4531.27 |
| 3 | 12070728 | 2017 | 6 | -25468.73 |
| 4 | 12070728 | 2017 | 8 | -24468.73 |
| 5 | 12070735 | 2017 | 9 | 19316.24 |
| 6 | 12070756 | 2017 | 3 | 10444.44 |

| Index | 科目 | 年 | 月 | 累计金额汇总 |
|---|---|---|---|---|
| 1 | 120707 | 2017 | 1 | 3.49353208... |
| 2 | 120707 | 2017 | 2 | 2.74671589... |
| 3 | 120707 | 2017 | 3 | 6.93724938... |
| 4 | 120707 | 2017 | 4 | 6.93740141... |
| 5 | 120707 | 2017 | 5 | 7.01545969... |
| 6 | 120707 | 2017 | 6 | 7.02892303... |

| Index | 科目 | 年 | 月 | 累计金额汇总 |
|---|---|---|---|---|
| 1 | 1207 | 2017 | 1 | 3.92445479... |
| 2 | 1207 | 2017 | 2 | 2.74671589... |
| 3 | 1207 | 2017 | 3 | 7.44437481... |
| 4 | 1207 | 2017 | 4 | 6.93981641... |
| 5 | 1207 | 2017 | 5 | 7.52753734... |
| 6 | 1207 | 2017 | 6 | 7.02907303... |

外存速度慢，尽量减少硬盘访问，争取遍历一次数据而获得多个结果。所以，我们采用游标+管道的机制。

当计算A6时，会同时计算A4、A5

外存计算（游标+管道）

| | A |
|---|---|
| 1 | =file("总账凭证-mid.btx") |
| 2 | =file("总账凭证-later.btx") |
| 3 | =A1.cursor@b() |
| 4 | =channel(A3).groupx((科目\100):科目,年,月;sum(累计金额):累计金额汇总) |
| 5 | =channel(A3).groupx((科目\10000):科目,年,月;sum(累计金额):累计金额汇总) |
| 6 | =A3.groupx((科目\1000000):科目,年,月;sum(累计金额):累计金额汇总) |
| 7 | =[A6,A5.result(),A4.result()].conjx() |
| 8 | >A2.export@z(A7) |

# 预先汇总方案—优化"一次遍历"查询—示例

友乾营

**经过上面两步数据预处理，结果数据可以直接作为报表的数据源，每个指标的计算条件只要相等比较就可以，而不再需要截取、计算前几位了。**

| | A | B | C |
|---|---|---|---|
| 1 | =file("总账凭证-later.btx") | | /指标参数列 |
| 2 | =A1.cursor@b() | | |
| 3 | =A2.select(concat(年,月)<=concat(year,month)) | | |
| 4 | for A3 | =@+A4.select(C4.contain(科目)).sum(累计金额汇总) | [1001,1002] |
| 5 | | =@+A4.select(C5.contain(科目)).sum(累计金额汇总) | [2702,153102,12310105,1122,12310101,12310401,12319001,12310201,12310301,12310501,12310601,12310701,12310801,12319101] |
| ... | | ... | ... |
| 104 | return [B4:B103] | | |

SPL

**按照预先汇总的思路，事先根据数据特征对数据进行预处理，可以让总的数据量变小，同时减少遍历量，从而避免前述方案中总是从最底层再去累加的模式。**

**经过实测：从报表取数到报表展现整个环节比“常规”方案足足提高了6-8倍左右！**

# 有序计算方案—数据预处理思路

友乾营

**需求特征：每次只需要取出总数据量的很小一部分(100个指标涉及的所有科目号大概几百个，即在几百万记录中取几百条)，这时我们通常能想到的是：如果能利用数据有序直接进行有序查找(若源数据有序，可以快速定位到这几百条记录，不需要遍历几百万记录甚至更多的数据)，将能够获得更好的查询效率。**

## 步骤一：多字段合并唯一主键、排序

**明细记录**

| 科目 | 年 | 月 | 金额 | ... |
|---|---|---|---|---|
| 1234567890 | 2017 | 1 | 100 | ... |
| 1234567890 | 2017 | 1 | 120 | ... |
| ... | ... | ... | ... | ... |

**用年和月两列字段动态计算一个变量值，称为"月号"，按照科目、月号分组，统计本科目下月号的累计金额！**

| 科目 | 月号 | 金额 | ... |
|---|---|---|---|
| 1234567890 | 37 | 220 | ... |
| ... | ... | ... | ... |

**月号计算规则：假设原始数据是从2014年开始的，所谓"月号"就是每条记录的时间是从初始年份1月开始的第几个月。公式：月号=(当前年-初始年)*12+当前月**

**(2017-2014)*12+1**

**月号和科目合并成唯一主键key，排序后进行存储，有序查询时就可以快速定位值。**

| 新增科目 | 月号 | 金额 | ... |
|---|---|---|---|
| 12345678 | 37 | 330 | ... |
| 123456 | 37 | 490 | ... |
| 1234 | 37 | 620 | ... |
| ... | ... | ... | ... |

| 新增科目 | 月号(转换后) |
|---|---|
| 12345678 | 370000000000 |
| 123456 | 370000000000 |
| 1234 | 370000000000 |
| ... | ... |

**月号为2位（假设数据记录跨度不超过99个月），科目为固定的10位，这样为了保证合并成主键后的唯一性，需要定义新主键的总长度为12位。**
**新主键的构造规则：key(12位)=月号(2位)*10000000000+科目号(最长10位)**

| key | 金额 | ... |
|---|---|---|
| 370000001234 | 620 | ... |
| 370000123456 | 490 | ... |
| 370012345678 | 330 | ... |
| ... | ... | ... |

# 有序计算方案—有序查询思路

友乾营

根据查询年、月、初始年，构造月号；接着与科目号构造key → 把查询指标的所有科目号合并，然后统一排序生成序号 → 通过序号在有序结果集中找到对应的金额 → 再利用位置序号把金额倒回到每个指标中，每个指标下对多个科目号的金额汇总，即指标汇总值


查询参数：年=2017、月=1、初始年=2014

月号=(当前年-初始年)*12+当前月

key=月号*10000000000+科目号

**指标参数A**

| 科目号 |
|---|
| 1478 |
| 18983412 |
| 202476 |

**指标参数B**

| 科目号 |
|---|
| 1234 |
| 156832 |

**转化后的指标参数A**

| key | 序号 |
|---|---|
| 370000001478 | 2 |
| 370018983412 | 4 |
| 370000202476 | 5 |

**转化后的指标参数B**

| key | 序号 |
|---|---|
| 370000001234 | 1 |
| 370000156832 | 3 |

**预处理后的有序结果集**

| 序号 | key | 金额 |
|---|---|---|
| … | … | … |
| 1 | 370000001234 | 110 |
| 2 | 370000001478 | 140 |
| 3 | 370000156832 | 150 |
| 4 | 370018983412 | 180 |
| 5 | 370000202476 | 200 |
| … | … | … |
| … | … | … |

**指标A的结果集**

| 序号 | 金额 |
|---|---|
| 2 | 140 |
| 4 | 180 |
| 5 | 200 |

**指标B的结果集**

| 序号 | 金额 |
|---|---|
| 1 | 110 |
| 3 | 150 |

# 有序计算方案—数据预处理思路—示例

友乾营

## 步骤一：合并主键、排序

| Index | 科目 | 月号 | 累计金额 |
|---|---|---|---|
| 1 | 1207071413 | 46 | 1230028.46 |
| 2 | 1207072813 | 40 | 4531.27 |
| 3 | 1207072813 | 42 | -25468.73 |
| 4 | 1207072813 | 44 | -24468.73 |
| 5 | 1207073513 | 45 | 19316.24 |
| 6 | 1207075613 | 39 | 10444.44 |
| 7 | 1207075613 | 40 | 156663.33 |
| 8 | 1207075613 | 41 | 630424.33 |

| | A | B |
|---|---|---|
| 1 | =file("总账凭证-pre.btx") | |
| 2 | =file("总账凭证-mid.btx") | |
| 3 | =A1.cursor@b() | >A3.run(((年-inityear)*12+月):月) |
| 4 | =A3.groupx(科目,月:月号;sum(金额):金额 ) | |
| 5 | for A4;科目 | =A5.run(金额=金额[-1]+金额) |
| 6 | | >A2.export@ab(B5,#1:科目,#2:月号,#3:累计金额) |

## 步骤二：月号和科目合并成主键key，然后进行排序

| | A |
|---|---|
| 1 | =file("总账凭证-mid.btx") |
| 2 | =file("总账凭证-later.btx") |
| 3 | =A1.cursor@b() |
| 4 | =channel(A3).groupx((科目\100):科目,月号;sum(累计金额):累计金额汇总) |
| 5 | =channel(A3).groupx((科目\10000):科目,月号;sum(累计金额):累计金额汇总) |
| 6 | =A3.groupx((科目\1000000):科目,月号;sum(累计金额):累计金额汇总) |
| 7 | =[A6,A5.result(),A4.result()].conjx() |
| 8 | =A7.new(#2*10000000000+#1:key,#3:累计金额汇总).sortx(key) |
| 9 | >A2.export@z(A8) |

| | A |
|---|---|
| … | … |
| 4 | =channel(A3).groupx(月号,(科目\100):科目;sum(累计金额):累 |
| 5 | =channel(A3).groupx(月号,(科目\10000):科目;sum(累计金 |
| 6 | =A3.groupx(月号,(科目\1000000):科目;sum(累计金额):累计金 |
| 7 | =[A6,A5.result(),A4.result()].mergex(月号,科目) |
| 8 | =A7.new(#1*10000000000+#2:key,#3:累计金额汇总) |
| … | … |

另一种写法

| Index | key | 累计金额汇总 |
|---|---|---|
| 1 | 370000001207 | 3.924454793E7 |
| 2 | 370000001214 | -1.217372485E7 |
| 3 | 370000001228 | 97692.31 |
| 4 | 370000001242 | 150424.0 |
| 5 | 370000001249 | -6391365.3 |

# 有序计算方案—有序查询思路—示例

友乾营

**把100个计算指标的科目号整理好，然后执行一次iselect()函数，把所有指标汇总结果都查找出来。**

**利用pos()的函数技巧，根据每个计算指标中多个科目号与月号构造的主键在结果集中的找到坐标位置(与key列字段比对)，返回位置序号,接着根据位置序号在结果集中找到的累计金额汇总字段进行求和，求和结果再按位置序号倒回到每个指标中，即每个指标的汇总值计算完成。**

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | /参数变量 | =now() | =((yyyy-inityear)*12+mm)*10000000000 | | |
| 2 | [1001,1002,1012] | [2001] | [1101] | [1121,12310106,12310206,12310306,12310706,12319006,12319106,…] | [2101] |
| … | … | … | … | … | … |
| 21 | [221102] | [1221,12310102,12310202,12310302,12310402,12310802,12319102,…] | [2221] | [1321,1401,1402,1403,1404,1405,1406,1407,1408,1412,1461,1471,…] | [1403,147101,1471050100,1408,…] |
| 22 | =[A2:E21] | =A22.(~.(~==C1+~)) | =A22.union().sort() | | |
| 23 | =file("总账凭证-later.btx") | | | | |
| 24 | =A23.iselect@b(C22,key) | =A24.fetch() | =B24.(key) | =B24.(汇总金额) | |
| 25 | =A22.(~.(C24.pos@b(~))) | /按照参数条件找到结果集中的坐标位置 | | | |
| 26 | =A25.(~.sum(D24(~))) | /获取指标结果后再汇总 | | | |
| 27 | return A26 | | =interval@ms(B1,now()) | | |

一次遍历
搞定难题

**实测结果：报表从取数到展现整个环节大概需要1-2秒，其中指标计算部分用时不到1秒。**

# 序号定位的查询流程

友乾营

**为了描述清楚序号定位与查找的过程，这里以单一指标参数A[1001,1002,1321]为例来说明获取结果流程：**

| Index | Member |
|---|---|
| 1 | [1001,1012,1321] |
| 2 | [2001] |
| 8 | [1126] |
| 9 | [2202] |
| 10 | [1123,12310103,123... |

Input argument

| Title | Value |
|---|---|
| yyy | 2017 |
| mm | 1 |
| inityear | 2014 |

OK Cancel

1、与外部参数，构造月号；接着与科目号构造key，形成指标的参数组集合

| Index | Member |
|---|---|
| 1 | [370000001001,370000001012,370000001321] |
| 2 | [370000002001] |
| 3 | [370000001101] |
| 4 | [370000001121,370012310106,37001231020... |
| 5 | [370000002101] |
| 6 | [370000001122,370012310101,37001231040... |
| 7 | [370000002201] |
| 8 | [370000001126] |
| 9 | [370000002202] |
| 10 | [370000001123,370012310103,37001231020... |

2、对指标参数组集合合并、排序,且能看到参数A在整个集合中的位置序号

| Index | Member |
|---|---|
| 1 | 370000001001 |
| 2 | 370000001012 |
| 3 | 370000001101 |
| 4 | 370000001121 |
| 5 | 370000001122 |
| 6 | 370000001123 |
| 7 | 370000001126 |
| 8 | 370000001131 |
| 9 | 370000001132 |
| 10 | 370000001221 |
| 11 | 370000001303 |
| 12 | 370000001304 |
| 13 | 370000001321 |
| 14 | 370000001401 |

3、参数组集合与文件中有序的key字段进行比对，返回结果集

| Index | key | 汇总金额 |
|---|---|---|
| 1 | 370000001001 | 935194.54 |
| 2 | 370000001012 | -9.71479655E7 |
| 3 | 370000001101 | -9820384.2999999... |
| 4 | 370000001121 | -1.058846174E8 |
| 5 | 370000001122 | -5.9411944700000... |
| 6 | 370000001123 | 3120910.6 |
| 7 | 370000001126 | 917550.89 |
| 8 | 370000001131 | 1.911443268E7 |
| 9 | 370000001132 | 2.37146520600000... |
| 10 | 370000001221 | -5.080509844E7 |
| 11 | 370000001303 | 2.82264175400000... |
| 12 | 370000001304 | -1.129630158E7 |
| 13 | 370000001321 | 8260104.83 |
| 14 | 370000001401 | -586254.69000000... |

4、指标参数在结果集中找到坐标，返回位置序号

| Index | Member |
|---|---|
| 1 | [1,2,13] |
| 2 | [43] |
| 3 | [3] |
| 4 | [4,,, ...] |
| 5 | [44] |
| 6 | [5,,, ...] |
| 7 | [45] |
| 8 | [7] |
| 9 | [46] |
| 10 | [6,,, ...] |

5、根据位置序号在结果集中找到的汇总金额字段进行求和

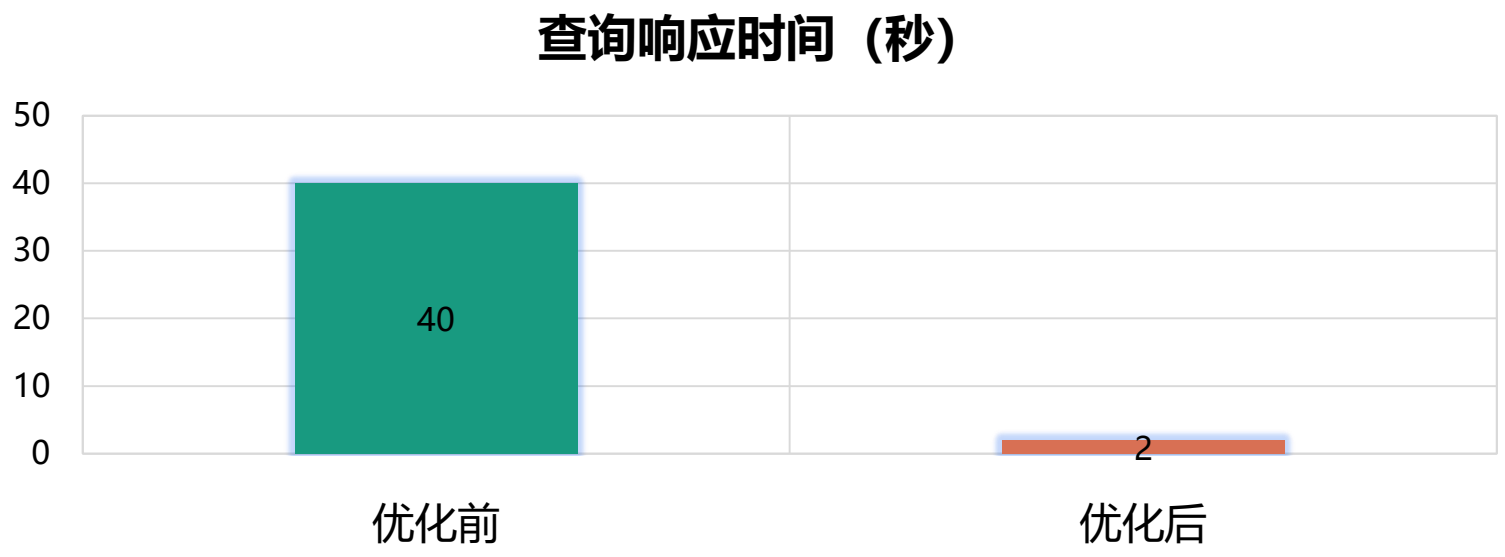
| Index | Member |
|---|---|
| 1 | -8.795266613E7 |
| 2 | -7383536.450000001 |
| 3 | -9820384.299999997 |
| 4 | -1.058846174E8 |
| 5 | -1.8202222122000003E8 |
| 6 | -5.941194470000001E7 |
| 7 | -5.927386471000002E7 |
| 8 | 917550.89 |
| 9 | 2.0029960970000003E7 |
| 10 | 3120910.6 |

6、求和结果再按位置序号倒回到每个指标中

# 实测:性能优化前后的对比

友乾营

**实际业务中，我们针对客户提供的生产环境进行了POC测试（原数据表大概6000万明细记录），在同等硬件环境下，优化后的方案比原方案在性能上有质的提升，结论如下：**




计算性能获得

**20倍提升**

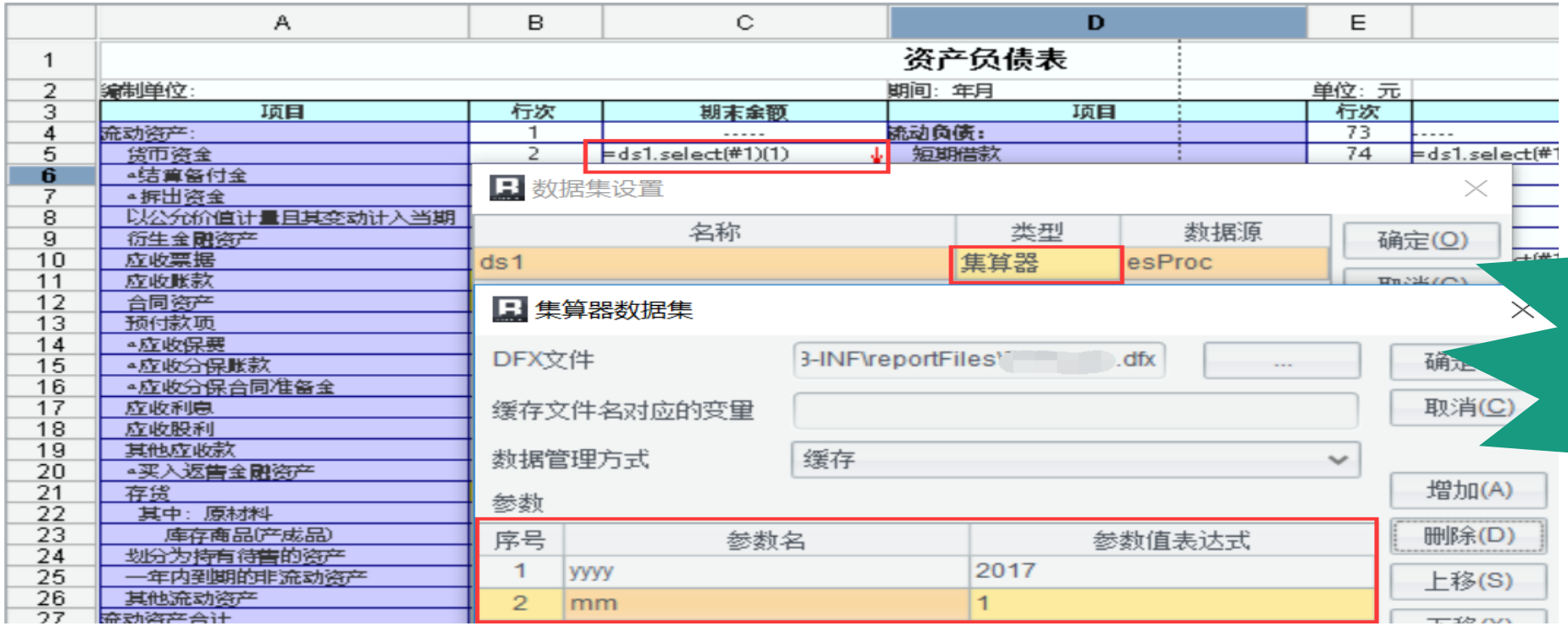
# 作为报表数据源

友乾营

**对于报表的制作过程来说，并不需要做什么改变，只需要把数据源切换到集算器即可。**

**1、导入excel表样，创建数据集类型为集算器，选取已经做好的SPL脚本；同时设置相应的查询参数。**

**2、在报表的每个单元格里分别按顺序取值，即可得到每个指标汇总结果；比如单元格C5的表达式写法：=ds1.select(#1)(1)，单元格C6的写法：=ds1.select(#1)(2)，...，报表单元格表达式从上往下，依次类推。样例如下图：**



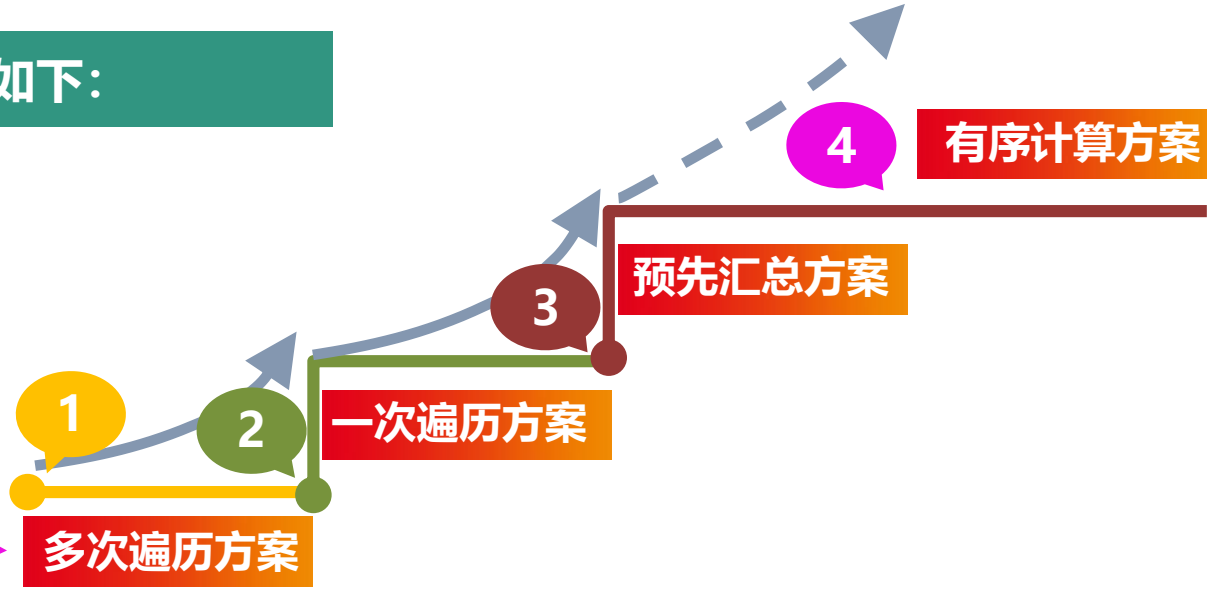

以润乾报表为例！

# 性能优化总结

友乾营

在实际的报表开发过程中，当我们遇到问题，往往并不能一开始就想到最优的解决办法。我们可以试着先用最简单、最容易的办法实现，然后再一步步进行优化；对比每种方案的存在的缺陷及改进后所带来的性能提升，从而最终满足业务需求。

上述中我们就采用了这种方式，逐步优化的步骤如下：




整个过程中，我们用到的集算器相关技术包括：游标、管道、遍历复用、数据外置、分组子集、跨行组计算、有序计算/查询、二分法查找、位置序号等。

了解了这些概念并熟练掌握集算器相关的函数后，我们就可以写出高效的代码，快速实现报表数据集的准备工作！


# 润乾线上直销系统

玩转好多乾

http://www.raqsoft.com.cn/wx/hdq-strategy.html