集算器

创新大数据计算引擎

复杂计算之多层递归案例

润乾软件出品

项目背景

石油化工行业，经常要构造特殊中间表，以物料、时间、指标名等为参数，查询出对应层级的指标值！

由于前端报表模块已经定型，用户不希望再改变，所以只能按照数据库的思路造出这种中间数据集以适用于报表；但事先不能确定维度及其层次的深度，导致SQL很难写！

为了便于描述
示例做了简化

时间维度第1层
+
物料维度第4层

时间维度第2层
+
物料维度第1层

时间维度第3层
+
物料维度第1层

结果记录

Index	时间	物料	指标	数值
31	20160202	油类	处理量	30000
32	20160202	油类	消耗量	2
33	20170101	油类	处理量	330
34	20170101	油类	消耗量	4
35	20170102	油类	处理量	3000
36	20170102	油类	消耗量	2
37	201602	100号柴油	处理量	10000
38	201602	100号柴油	消耗量	1
39	201602	90号柴油	处理量	20000
40	201602	90号柴油	消耗量	1
41	201602	柴油	处理量	30000
42	201602	柴油	消耗量	2
43	201602	成品油	处理量	30000
44	201602	成品油	消耗量	2
45	201602	油类	处理量	30000
46	201602	油类	消耗量	2
47	2016	100号柴油	处理量	10000
48	2016	100号柴油	消耗量	1
49	2016	90号柴油	处理量	20000
50	2016	90号柴油	消耗量	1

怎么做？

明细记录

Index	时间	物料	处理量	消耗量
1	20170101	92号汽油	100	1
2	20170101	98号汽油	200	1
3	20170102	100号柴油	1000	1
4	20170102	90号柴油	2000	1
5	20160202	100号柴油	10000	1
6	20160202	90号柴油	20000	1
7	20170101	90号柴油	10	1
8	20170101	100号柴油	20	1

时间维度

Index	时间	上级时间
1	20170101	201701
2	20170102	201701
3	20171230	201712
4	20171231	201712
5	201701	2017
6	201712	2017
7	20160202	201602
8	201602	2016
9	2017	(null)
10	2016	(null)

物料维度

Index	物料	上级物料
1	92号汽油	汽油
2	98号汽油	汽油
3	100号柴油	柴油
4	90号柴油	柴油
5	汽油	成品油
6	柴油	成品油
7	成品油	油类
8	油类	(null)

源数据结构

维度表的结构用“编号-上级编号”这样的双字段来表示上下级关系，上级编号为空，则表示本条数据为顶级。需要注意的是：维度层数事先未知，每次计算都可能不同！

时间维度表为3层

Index	时间	上级时间
1	20170101	201701
2	20170102	201701
3	20171230	201712
4	20171231	201712
5	201701	2017
6	201712	2017
7	20160202	201602
8	201602	2016
9	2017	(null)
10	2016	(null)

物料维度表为4层

Index	物料	上级物料
1	92号汽油	汽油
2	98号汽油	汽油
3	100号柴油	柴油
4	90号柴油	柴油
5	汽油	成品油
6	柴油	成品油
7	成品油	油类
8	油类	(null)

维度字段对应各维度表的叶子节点

明细记录

	时间	物料	处理量	消耗量
1	20170101	92号汽油	100	1
2	20170101	98号汽油	200	1
3	20170102	100号柴油	1000	1
4	20170102	90号柴油	2000	1
5	20160202	100号柴油	10000	1
6	20160202	90号柴油	20000	1
7	20170101	90号柴油	10	1
8	20170101	100号柴油	20	1

指标

算法描述

将原始表各维度的各层级进行叉乘分组汇总，对每种组合的处理量和消耗量进行求和汇总，如物料维度为4层、时间维度为3层，则需进行12次分组汇总。示意图如下：

明细记录

Index	时间	物料	处理量	消耗量
1	20170101	92号汽油	100	1
2	20170101	98号汽油	200	1
3	20170102	100号柴油	1000	1
4	20170102	90号柴油	2000	1
5	20160202	100号柴油	10000	1
6	20160202	90号柴油	20000	1
7	20170101	90号柴油	10	1
8	20170101	100号柴油	20	1

时间维度

Index	时间	上级时间
1	20170101	201701
2	20170102	201701
3	20171230	201712
4	20171231	201712
5	201701	2017
6	201712	2017
7	20160202	201602
8	201602	2016
9	2017	(null)
10	2016	(null)

物料维度

Index	物料	上级物料
1	92号汽油	汽油
2	98号汽油	汽油
3	100号柴油	柴油
4	90号柴油	柴油
5	汽油	成品油
6	柴油	成品油
7	成品油	油类
8	油类	(null)

1 时间第1层+物料第1层

Index	时间	物料	处理量	消耗量
1	20160202	100号柴油	10000	1
2	20160202	90号柴油	20000	1
3	20170101	100号柴油	20	1
4	20170101	90号柴油	10	1
5	20170101	92号汽油	100	1
6	20170101	98号汽油	200	1
7	20170102	100号柴油	1000	1
8	20170102	90号柴油	2000	1

2 时间第1层+物料第2层

Index	时间	物料	处理量	消耗量
1	20160202	柴油	30000	2
2	20170101	柴油	30	2
3	20170101	汽油	300	2
4	20170102	柴油	3000	2

... ...

11 时间第3层+物料第3层

Index	时间	物料	处理量	消耗量
1	2016	成品油	30000	2
2	2017	成品油	3330	6

12 时间第3层+物料第4层

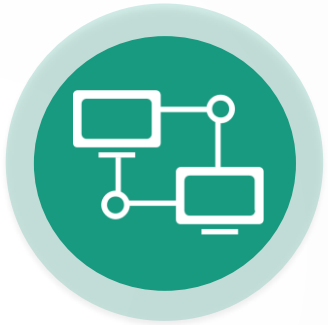
Index	时间	物料	处理量	消耗量
1	2016	油类	30000	2
2	2017	油类	3330	6

原算法存在问题

由于前端报表模型已固定，开发项目组只能通过SQL事先要把各种组合情况计算好（将各维度、各层级按一定规律组合），用上百行代码才实现一种类型的算法！

难以简化

SQL无法用直观的方式表达层级关系，必须用复杂的关联嵌套语句来表达，语句长，要自动拼接生成基本很难。



工作量大

前端报表模型已固定，又不能确定层数，既然难以简化就只能老老实实依次写出叉乘语句，而类似算法有300多个。

维护难

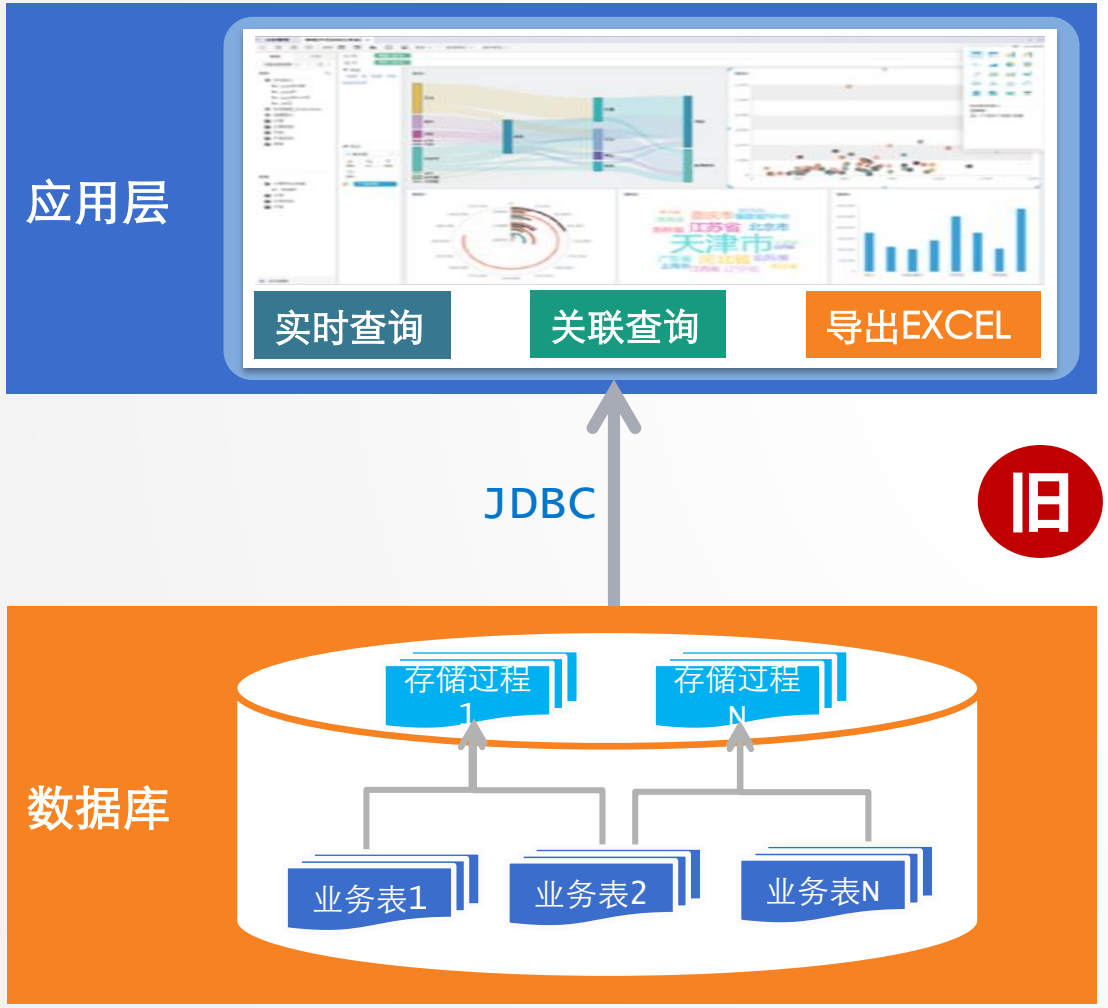
算法最终交给外包项目组去维护，但因代码太长、太复杂，多个维护人员都表示无法顺利解读，给维护带来隐患。

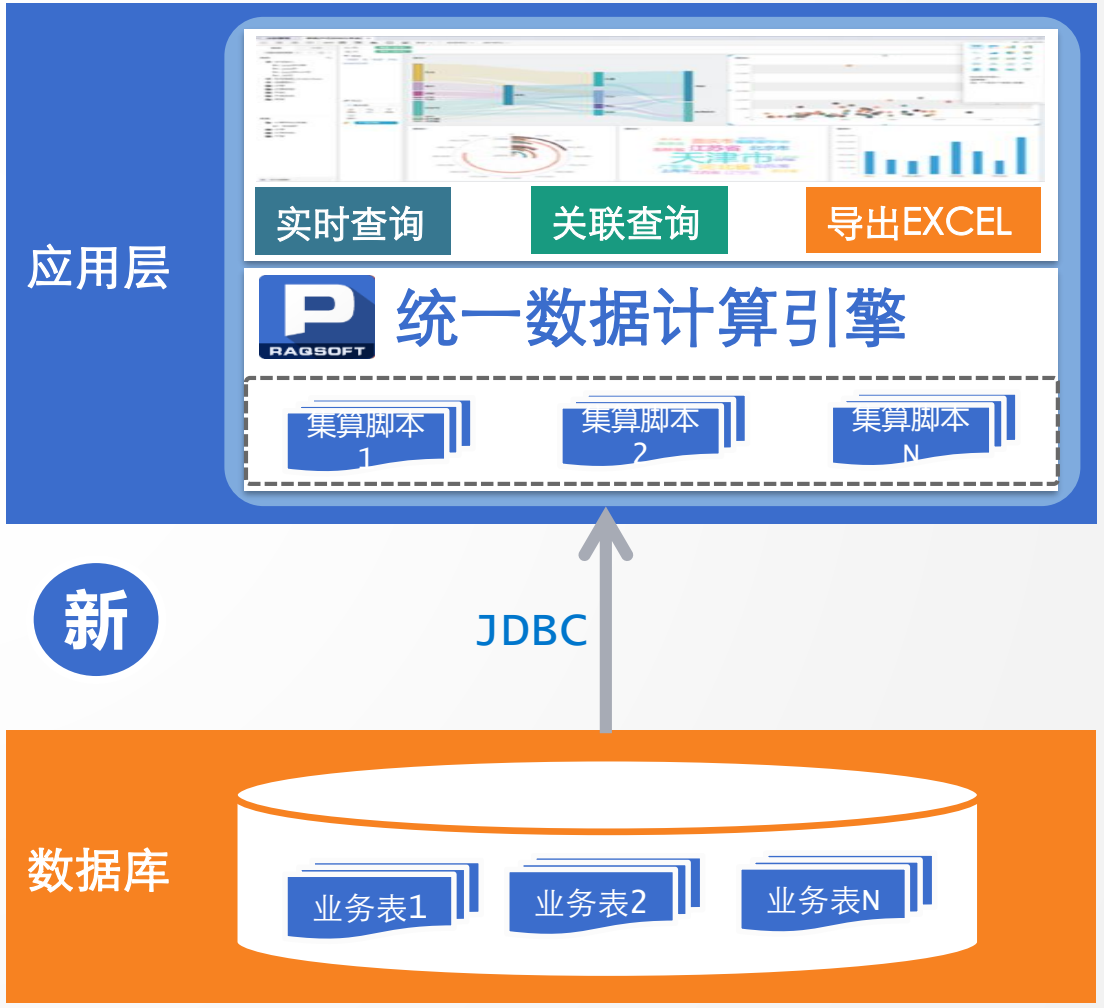
优化难

每种算法都是基于原始表进行计算，若计算下一层可借助上一层结果接着运算，则性能可提高，但SQL却很难优化。

引入数据计算层

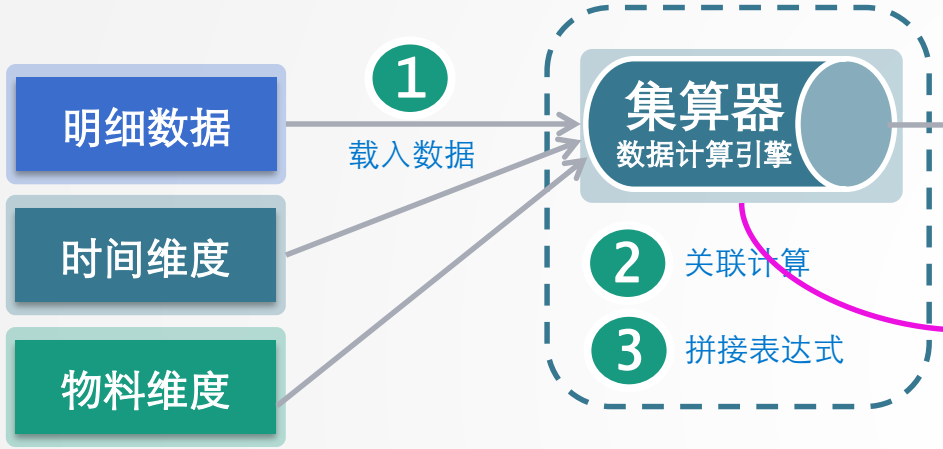
在应用层集成数据计算引擎中间件，使用一致的结构化计算模型，为应用提供统一计算服务！且独立的计算层可极大的降低应用与数据库的耦合，后期维护也完全模块化！





集算器解法流程

集算器实现本算法仅用了12行代码！



Index	时间	物料	处理量	消耗量
16	20160202	油类	30000	2
17	20170101	油类	330	4
18	20170102	油类	3000	2
19	201602	100号柴油	10000	1
20	201602	90号柴油	20000	1
21	201701	100号柴油	1020	2

Index	时间	物料	指标	数值
34	20170101	油类	消耗量	4
35	20170102	油类	处理量	3000
36	20170102	油类	消耗量	2
37	201602	100号柴油	处理量	10000
38	201602	100号柴油	消耗量	1
39	201602	90号柴油	处理量	20000

	A	B	C	D	E	F	G	H	I
1	=原始表=db.query("select 时间,物料,处理量,消耗量 from 指标表")				=时间维度=db.query("select 时间,上级时间 from 时			=物料维度=db.query("select 物料,上级	
2					=时间维度.switch(上级时间,时间维度:时间)			=物料维度=物料维度.switch(上级物料,	
3	=原始表=原始表.switch(时间,时间维度:时间;物料,物料维度:物料)								
4	=时间层级=时间维度.select@1(时间==原始表(1).时间.时间).prior(上级时间)								
5	=物料层级=物料维度.select@1(物料==原始表(1).物料.物料).prior(上级物料)								
6	=结果=原始表.create()								
7	for 时间层级.len()	=时间串="时间"+if(A7==1,"","."+(A7-1).("上级时间").string("."))+".时间:时间"							
8		for 物料层级.len()	=物料串="物料"+if(B8==1,"","."+(B8-1).("上级物料").string("."))+".物料:物料"						
9			=表达式="原始表.groups("+时间串+","+物料串+";sum(处理量):处理量,sum(消耗量):消耗量)"						=output@t(表
10			=eval(表达式)					/汇总	
11			=结果=结果|C10					/追加	
12	=结果=结果.pivot@r(时间,物料;指标,数值)				/转置为结果格式				

数据层级关系详解

3

时间	上级时间
2017	(null)

2

时间	上级时间
201701	2017

1

时间	上级时间
20170101	201701

4

物料	上级物料
油类	(null)

3

物料	上级物料
成品油	油类

2

物料	上级物料
汽油	成品油

1

物料	上级物料
92号汽油	汽油

Index	时间	物料	处理量	消耗量
1	20170101	92号汽油	100	1
2	20170101	98号汽油	200	1
3	20170102	100号柴油	1000	1
4	20170102	90号柴油	2000	1
5	20160202	100号柴油	10000	1
6	20160202	90号柴油	20000	1
7	20170101	90号柴油	10	1
8	20170101	100号柴油	20	1

明细表被外键属性化后关系图

分组、汇总

Index	时间	物料	处理量	消耗量
16	20160202	油类	30000	2
17	20170101	油类	330	4
18	20170102	油类	3000	2
19	201602	100号柴油	10000	1
20	201602	90号柴油	20000	1
21	201701	100号柴油	1020	2
22	201701	90号柴油	2010	2
23	201701	92号汽油	100	1
24	201701	98号汽油	200	1
25	201602	柴油	30000	2
26	201701	柴油	3030	4
27	201701	汽油	300	2
28	201602	成品油	30000	2
29	201701	成品油	3330	6
30	201602	油类	30000	2
31	201701	油类	3330	6
32	2016	100号柴油	10000	1
33	2016	90号柴油	20000	1

转置为结果格式

Index	时间	物料	指标	数值
34	20170101	油类	消耗量	4
35	20170102	油类	处理量	3000
36	20170102	油类	消耗量	2
37	201602	100号柴油	处理量	10000
38	201602	100号柴油	消耗量	1
39	201602	90号柴油	处理量	20000
40	201602	90号柴油	消耗量	1
41	201602	柴油	处理量	30000
42	201602	柴油	消耗量	2
43	201602	成品油	处理量	30000
44	201602	成品油	消耗量	2
45	201602	油类	处理量	30000
46	201602	油类	消耗量	2
47	2016	100号柴油	处理量	10000
48	2016	100号柴油	消耗量	1
49	2016	90号柴油	处理量	20000

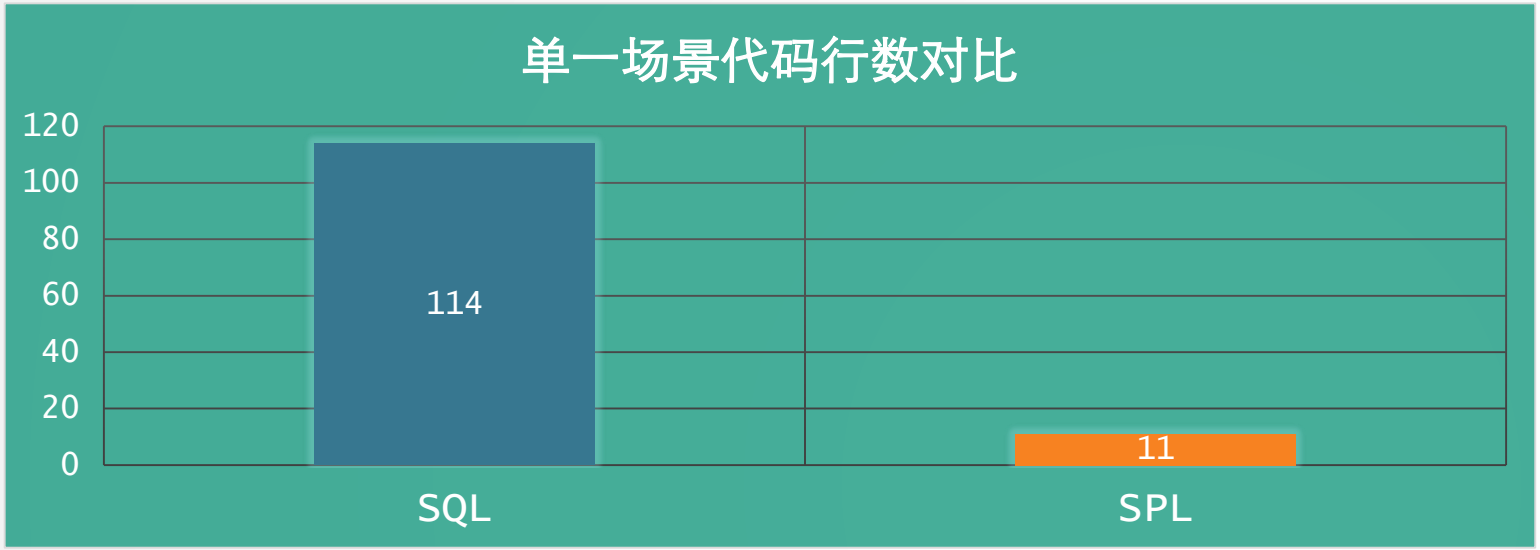
代码相当精简！

统计2017年汽油的指标情况？

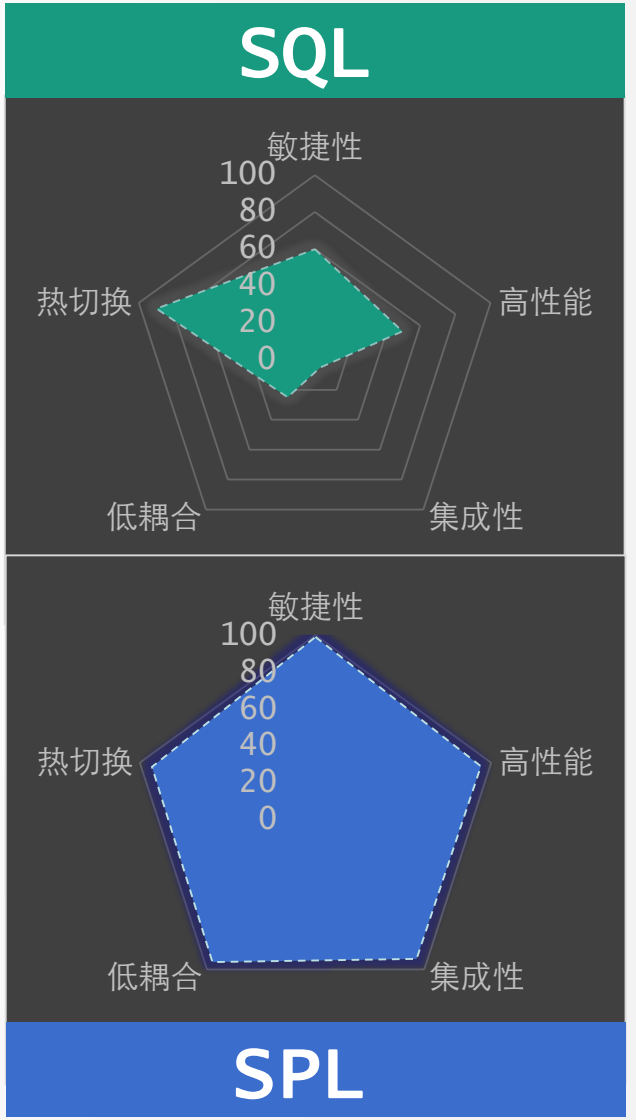
	A
1	原始表.groups(时间.上级时间.上级时间.时间:时间,物料.上级物料.物料:物料;sum(处理量):处理量,sum(消耗量):消耗量)

从上例的代码可以看到，集算器支持对象的方式访问关联表，而不是SQL的嵌套查询；预关联建立后还可复用，相较于SQL的JOIN运算，不仅性能上会更好，且书写更简单！

实测：集算器在各方面的工作量评估



对比指标	SQL	SPL	提升
代码行（单一场景）	114	12	9.5倍
工作量	30人天	5人天	6倍
执行性能	53秒	4秒	平均性能提升13倍
可维护性	无	容易	易管理、热切换
可优化性	较差	容易	质变



为什么集算器能如此提高效率

敏捷语法体系

语法简洁，表达能力强，实现同样算法，只需更少的代码

报表简单开发

开发环境配置简单，让数据计算更有效率

应用部署更易

无框架，标准接口易于实现应用嵌入式集成

优化体系结构

算法外置于应用，文件形式易于管理，计算模块彻底独立化

创新技术
推动应用进步！


各设备故障分析
近7天设备的总故障率
近7天
近1天
近6小时
1.53%
0.95%
2.40%
近7天故障情况
全年故障综合分析
近7天故障次数前5的设备
服务器CPU使用率
今日
昨日
今天硬盘情况