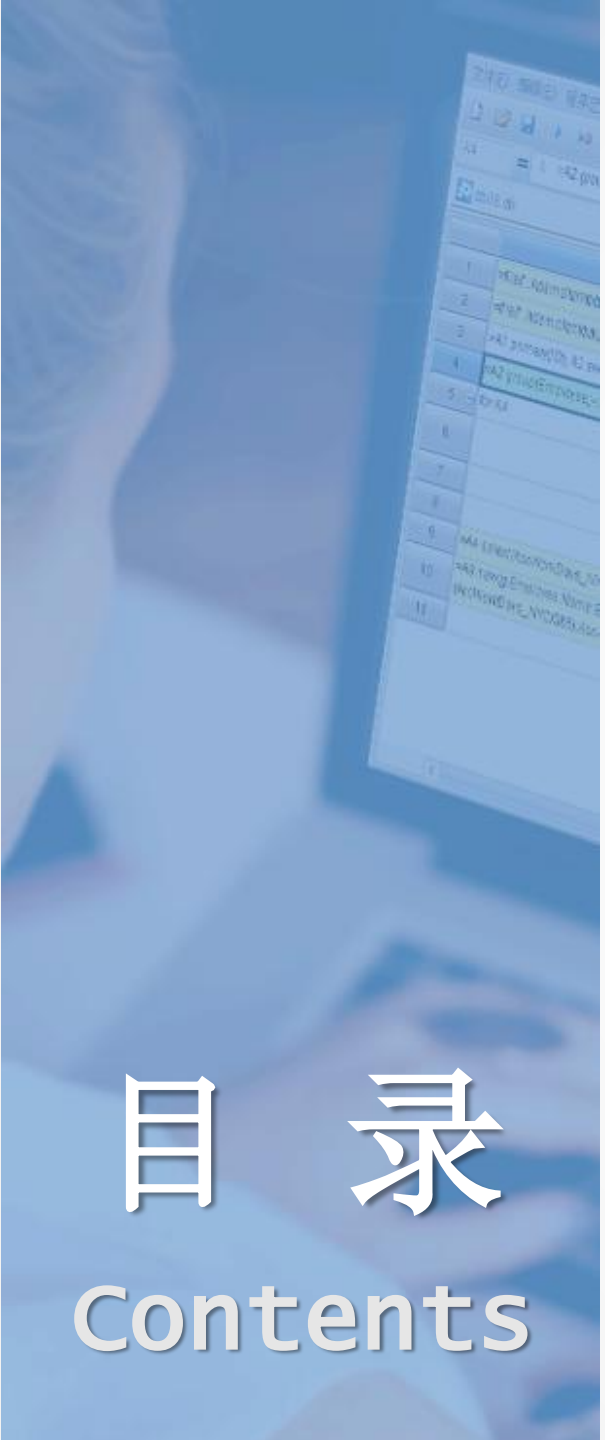
集算器

创新大数据计算引擎

Java计算辅助神器

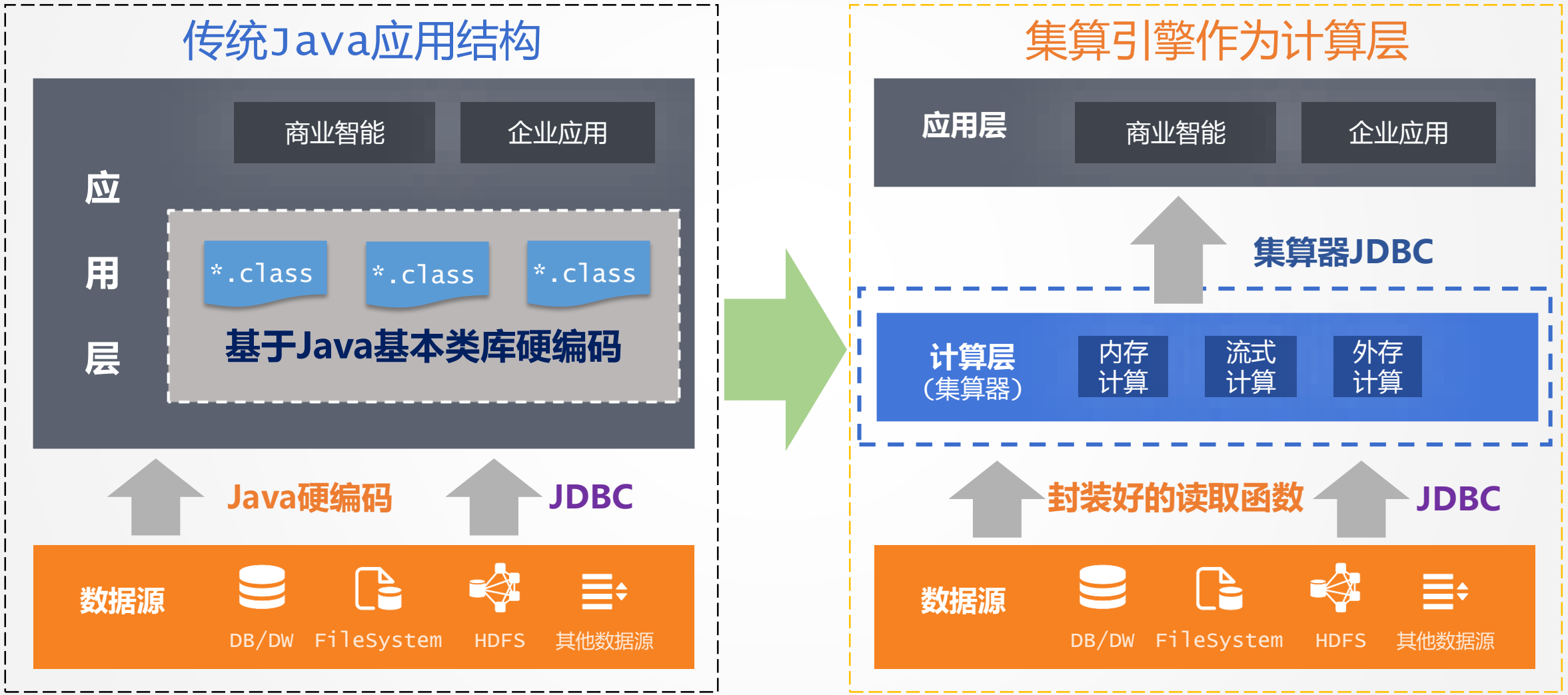
润乾软件出品



目 录

Contents

应用结构


传统Java应用结构
应用层
商业智能
企业应用
*.class
*.class
*.class
基于Java基本类库硬编码
Java硬编码
JDBC
数据源
DB/DW
FileSystem
HDFS
其他数据源
集算引擎作为计算层
应用层
商业智能
企业应用
集算器JDBC
计算层
(集算器)
内存计算
流式计算
外存计算
封装好的读取函数
JDBC
数据源
DB/DW
FileSystem
HDFS
其他数据源

减轻代码量

提供丰富的结构化数据计算函数，代替通过Java基本类库硬编码

举例：从文件中读取销售记录，计算出每个销售员的销售额

```
Comparator<salesRecord> comparator = new Comparator<salesRecord>() {
      public int compare(salesRecord s1, salesRecord s2) {
            if (!s1.salesman.equals(s2.salesman)) {
                  return s1.salesman.compareTo(s2.salesman);
            } else {
                  return s1.ID.compareTo(s2.ID);
            }
      }
};
Collections.sort(sales, comparator);
ArrayList<resultRecord> result = new ArrayList<resultRecord>();
salesRecord standard = sales.get(0);
float sumValue = standard.value;
for(int i = 1;i < sales.size(); i++){
      salesRecord rd = sales.get(i);
      if(rd.salesman.equals(standard.salesman)){
            sumValue = sumValue+rd.value;
      }else{
            result.add(new resultRecord(standard.salesman,sumValue));
            standard = rd;
            sumValue = standard.value;
      }
}
result.add(new resultRecord(standard.salesman,sumValue));
return result;
```

Java硬编码

	A
1	=file("salesRecord.txt").import@t().groups(salesman; sum(value))

```
…
Class.forName("com.esproc.jdbc.InternalDriver");
con= DriverManager.getConnection("jdbc:esproc:local://");
//调用存储过程，其中createTable1是dfx的文件名
st =con.prepareCall("call salesvalue()");
//执行存储过程
st.execute();
//获取结果集
ResultSet rs = st.getResultSet();
…
```

结构化数据计算函数使用与JDBC调用

可以看到，硬编码实现计算存在很多缺点：基础类库缺乏结构化算法，开发效率低，开发周期长，代码冗长难以维护，耦合太深难以复用。

多样性数据源

统一的数据结构使得计算代码具有一致性

举例：从txt、excel、数据库中分别读取2013、2014、2015年的销售记录，按客户分组求总金额

	A	B
1	=file("2013.txt").import@t()	/读取txt
2	=file("2014.xlsx").xlsimport@t()	/读取excel
3	=demo.query("select * from 销售记录表")	/查询数据库
4	=[A1:A3].conj().groups(客户;sum(订单金额):总金额)	/三者的聚合运算

客户	总金额
ALFKI	538.7
ANATR	2587.95...
ANTON	7515.34...
AROUT	13315.0
BERGS	26968.1...
BLAUS	3239.8
BLONP	19088.0
BOLID	6383.8

订单ID	客户	订购日期	员工ID	ID	订单金额
10248	VINET	2013-07-04	5	1	2440.0
10249	TOMSP	2013-07-05	6	2	1863.4
10250	HANAR	2013-07-08	4	3	1813.0
10251	VICTE	2013-07-08	3	4	670.8
10252	SUPRD	2013-07-09	4	5	3730.0

A1中的txt数据

订单ID	客户	订购日期	员工ID	ID	订单金额
10400	EASTC	2014-01-01	1	153	3063
10401	RATTC	2014-01-01	1	154	3868.6
10402	ERNSH	2014-01-02	8	155	2713.5
10403	ERNSH	2014-01-03	4	156	1005.9
10404	MAGAA	2014-01-03	2	157	1675

A2中的excel数据

订单ID	客户	订购日期	员工ID	ID	订单金额
10808	OLDWO	2015-01-01	2	561	1660.0
10809	WELLI	2015-01-01	7	562	140.0
10810	LAUGB	2015-01-01	2	563	187.0
10811	LINOD	2015-01-02	8	564	852.0
10812	REGGC	2015-01-02	5	565	1852.0

A3中的数据库数据

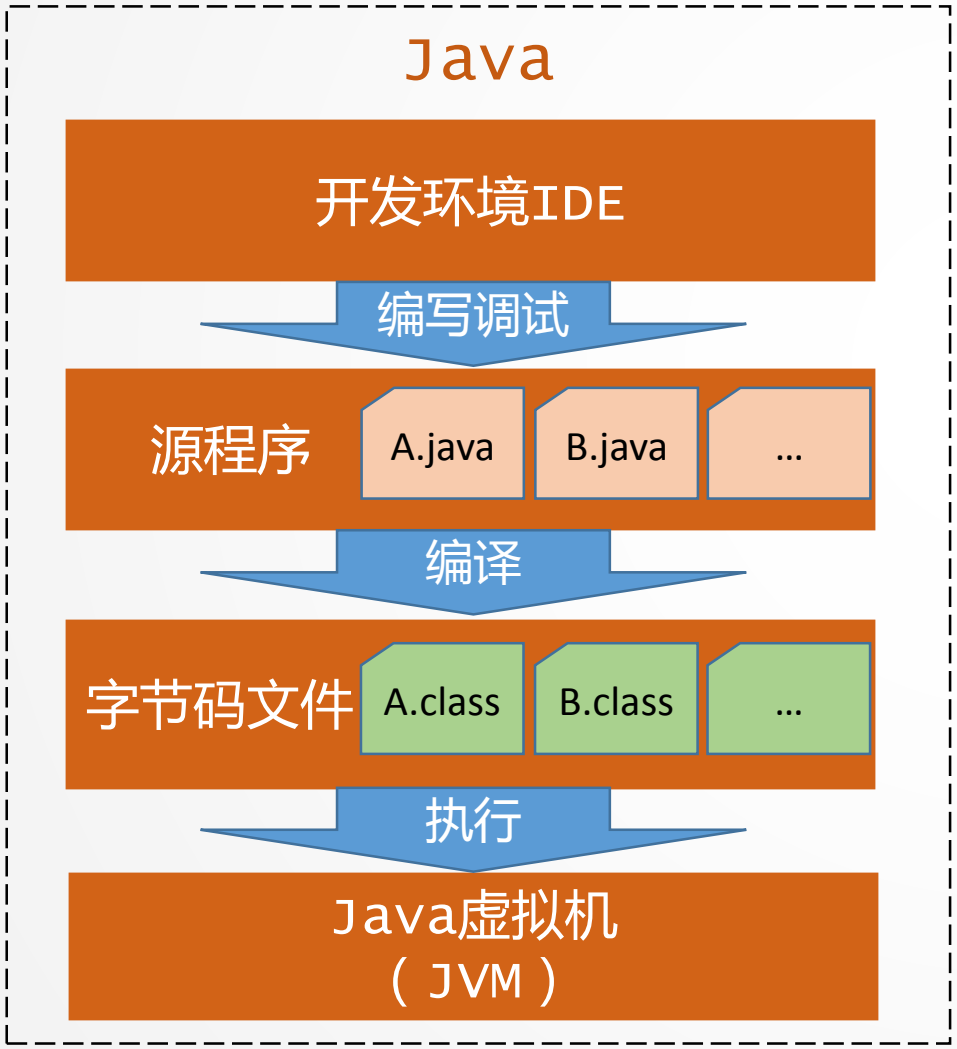
1	订单ID	客户	订购日期	员工ID	ID	订单金额
2	10248	VINET	2013-07-04	5	1	2440.0
3	10249	TOMSP	2013-07-05	6	2	1863.4
4	10250	HANAR	2013-07-08	4	3	1813.0
5	10251	VICTE	2013-07-08	3	4	670.8
6	10252	SUPRD	2013-07-09	4	5	3730.0
7	10253	HANAR	2013-07-10	3	6	1444.8
8	10254	CHOPS	2013-07-11	5	7	625.2
9	10255	RICSU	2013-07-12	9	8	2490.5
10	10256	WELLI	2013-07-15	3	9	517.8

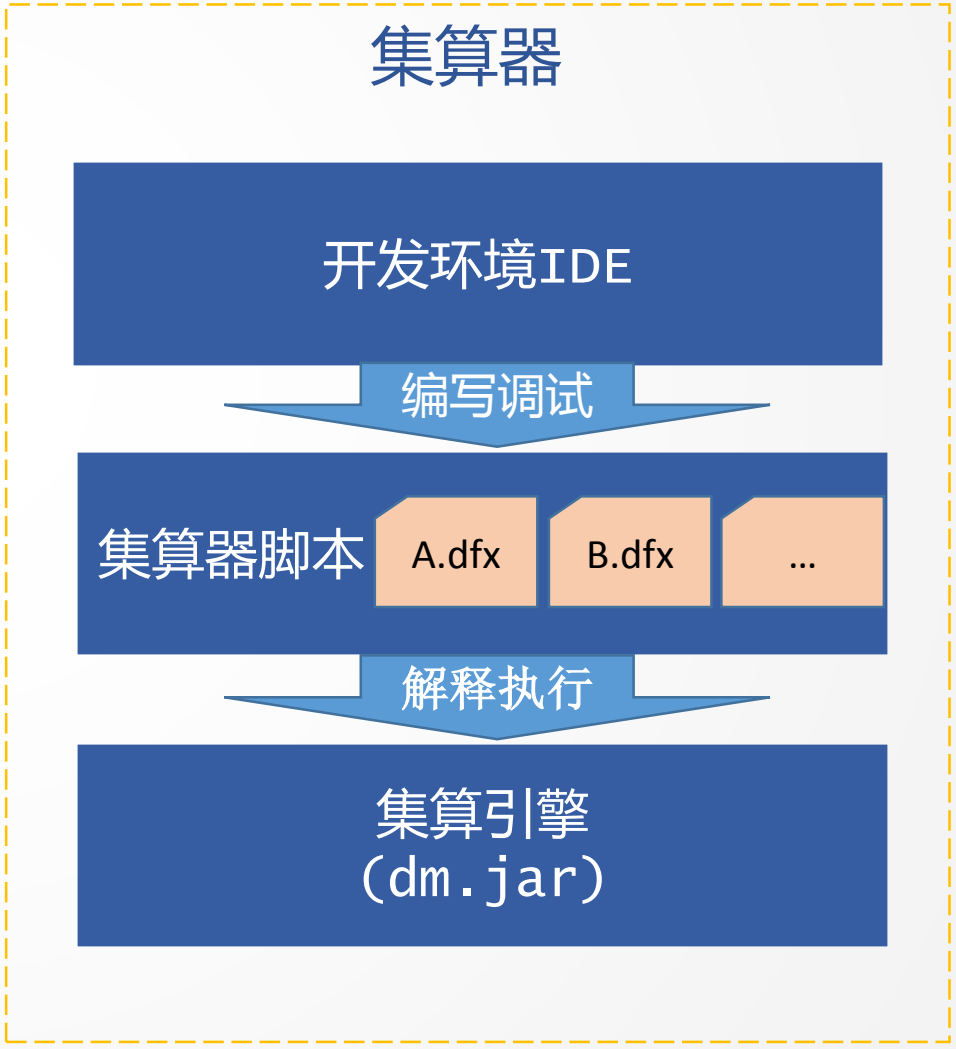
1	订单ID	客户	订购日期	员工ID	ID	订单金额
2	10400	EASTC	1/1/2014	1	153	3063
3	10401	RATTC	1/1/2014	1	154	3868.6
4	10402	ERNSH	1/2/2014	8	155	2713.5
5	10403	ERNSH	1/3/2014	4	156	1005.9
6	10404	MAGAA	1/3/2014	2	157	1675
7	10405	LINOD	1/6/2014	1	158	400
8	10406	QUEEN	1/7/2014	7	159	2018.2
9	10407	OTTIK	1/7/2014	2	160	1194
10	10408	FOLIG	1/8/2014	8	161	1622.4

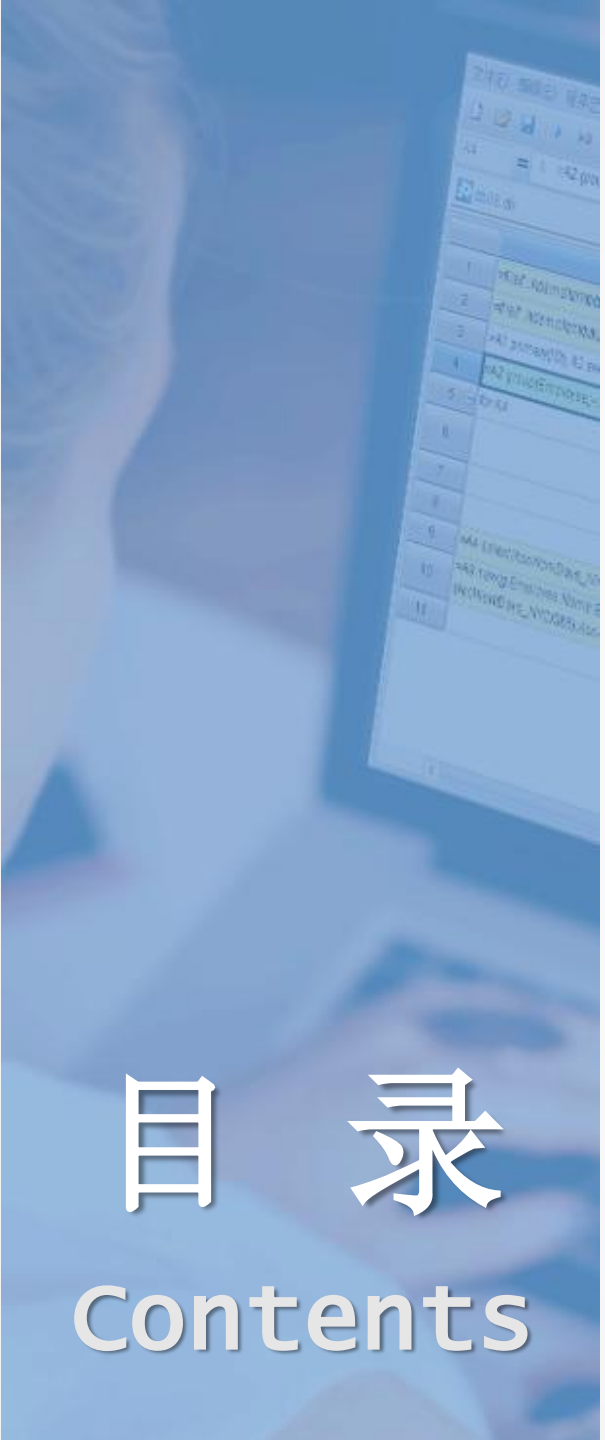
*	订单ID	客户	订购日期	员工ID	ID	订单金额
1	10808	OLDWO	2015-01-01	2	561	1660.0
2	10809	WELLI	2015-01-01	7	562	140.0
3	10810	LAUGB	2015-01-01	2	563	187.0
4	10811	LINOD	2015-01-02	8	564	852.0
5	10812	REGGC	2015-01-02	5	565	1852.0
6	10813	RICAR	2015-01-05	1	566	648.0
7	10814	VICTE	2015-01-05	3	567	2070.0
8	10815	SAVEA	2015-01-05	2	568	40.0
9	10816	GREAL	2015-01-06	4	569	8891.0
10	10817	KOENE	2015-01-06	3	570	11490.7

热切换

解释执行的集算器脚本可实现热切换







目 录

Contents

常规计算

专业的结构化计算层可以大幅度地简化代码编写，使文件具有计算能力

举例：1、排序：从文件中读取销售记录，按订购年份升序、订单金额降序排序；2、过滤：查找订购年份等于2014年且订单金额小于5000的记录；3、分组：、按客户、订购年份分组，统计订单总金额和订单数

	A	B
1	=file("销售记录表.txt").import@t()	/读取txt
2	=A1.sort(year(订购日期),订单金额:-1)	/1、排序
3	=A1.select(year(订购日期)==2014 && 订单金额<5000)	/2、过滤
4	=A1.groups(客户,year(订购日期):年份;sum(订单金额):总金额,count(~):订单数)	/3、分组

	订单ID	客户	订购日期	员工ID	ID	订单金额
2	10248	VINET	2013-07-04	5	1	2440.0
3	10249	TOMSP	2013-07-05	6	2	1863.4
4	10250	HANAR	2013-07-08	4	3	1813.0
5	10251	VICTE	2013-07-08	3	4	670.8
6	10252	SUPRD	2013-07-09	4	5	3730.0
7	10253	HANAR	2013-07-10	3	6	1444.8
8	10254	CHOPS	2013-07-11	5	7	625.2
9	10255	RICSU	2013-07-12	9	8	2490.5
10	10256	WELLI	2013-07-15	3	9	517.8

A1中部分销售记录数据

订单ID	客户	订购日期	员工ID	ID	订单金额
10281	ROMEY	2013-08-14	4	34	86.5
10391	DRACD	2013-12-23	3	144	86.4
10271	SPLIR	2013-08-01	6	24	48.0
10424	MEREP	2014-01-23	7	177	11493.2
10417	SIMOB	2014-01-16	4	170	11283.2
10440	SAVEA	2014-02-10	4	193	10793.1

A2中排序结果

订单ID	客户	订购日期	员工ID	ID	订单金额
10400	EASTC	2014-01-01	1	153	3063.0
10401	RATTC	2014-01-01	1	154	3868.6
10402	ERNSH	2014-01-02	8	155	2713.5
10403	ERNSH	2014-01-03	4	156	1005.9
10404	MAGAA	2014-01-03	2	157	1675.0

A3中过滤结果

客户	年份	总金额	订单数
ALFKI	2015	538.7	1
ANATR	2013	88.8	1
ANATR	2014	1129.75	3
ANATR	2015	1369.4	2
ANTON	2013	403.2	1
ANTON	2014	6452.15	5
ANTON	2015	660.0	1

A4中分组结果

使用SQL

对文件也能够类似数据库表一样使用SQL查询

分组举例：对电子产品订单信息（Order_Electronics.txt），按订购日期分组，统计订购数量总和大于110000的每日订单数和总订购数量

	A
1	$(esProcOdbc) select Date, count(ID) Count, sum(Quantity) Sum from Order_Electronics.txt group by Date having sum(Quantity)>110000 order by Sum

Index	Date	Count	Sum
1	2013-12-25	139	110002
2	2013-07-22	136	110149
3	2013-04-19	138	110417
4	2013-07-11	136	110655
5	2013-02-10	139	110717
6	2013-04-10	137	110811
7	2013-08-29	138	110856
8	2013-07-26	139	111163
9	2013-01-05	139	111454
10	2013-04-29	136	111526

在分组后，用having从句，从结果中选出sum(Quantity)大于110,000的数据

关联举例：对州数据文件（states.txt）与城市数据文件（cities.txt），按州编号（StateId）关联。

	A
1	$(esProcOdbc) select C.NAME,C.POPULATION, S.StateId, S.Abbr from states.txt S join cities.txt C on S.StateId=C.STATEID

Index	C.NAME	C.POPULATION	S.StateId	S.Abbr
1	Argentina	435245.0	5	CA
2	Auckland	402777.0	5	CA
3	Anaheim	334425.0	5	CA
4	Bakersfield	308392.0	5	CA
5	Chula Vista	212756.0	5	CA
6	Chicago	2886251.0	13	IL
7	Buffalo	276059.0	32	NY
8	Austin	671873.0	43	TX
9	Arlington	349944.0	43	TX
10	Corpus Christi	285267.0	43	TX

用join语句将两个外部表关联，列出能找到对应州简称的城市信息

大文本

考虑到内存和性能，大文本应当分批读入、批量计算

排序举例：对大文本文件（销售记录表.txt），按订单金额升序排序，取前10条记录

	A	B
1	=file("销售记录表.txt").cursor@t()	/游标方式打开文件
2	=A1.sortx(订单金额)	/订单金额升序排序
3	=A2.fetch(10)	/取前十条

订单ID	客户	订购日期	员工ID	ID	订单金额
10782	CACTU	2014-12-17	9	535	12.5
10807	FRANS	2014-12-31	4	560	18.4
10586	REGGC	2014-07-02	9	339	28.0
10767	SUPRD	2014-12-05	4	520	28.0
10898	OCEAN	2015-02-20	4	651	30.0

过滤举例：对大文本文件（销售记录表.txt），查找订购年份等于2014年且订单金额小于50的记录

	A	B
1	=file("销售记录表.txt").cursor@t()	/游标方式打开文件
2	=A1.select(year(订购日期)==2014 && 订单金额<50)	/按条件过滤游标
3	=A2.fetch()	/读取游标数据

订单ID	客户	订购日期	员工ID	ID	订单金额
10422	FRANS	2014-01-22	2	175	49.8
10586	REGGC	2014-07-02	9	339	28.0
10674	ISLAT	2014-09-18	4	427	45.0
10767	SUPRD	2014-12-05	4	520	28.0
10782	CACTU	2014-12-17	9	535	12.5
10807	FRANS	2014-12-31	4	560	18.4

分组举例：对大文本文件（销售记录表.txt），按客户、订购年份分组，统计订单总金额和订单数

	A	B
1	=file("销售记录表.txt").cursor@t()	/游标方式打开文件
2	=A1.groups(客户,year(订购日期):年份;sum(订单金额):总金额,count(~):订单数)	/按客户、订购年份分组，统计订单总金额和订单数

客户	年份	总金额	订单数
ALFKI	2015	538.7	1
ANATR	2013	88.8	1
ANATR	2014	1129.75	3
ANATR	2015	1369.4	2
ANTON	2013	403.2	1

分段并行

并行查询可提高性能

并行过滤举例：对大文本文件（销售记录表.txt），查找订购年份等于2014年且订单金额小于50的记录

	A	B
1	4	
2	fork to(A1)	=file("销售记录表.txt").cursor@t(;A2:A1)
3		=B2.select(订单金额>40 && 订单金额<50).fetch()
4	=A2.conj()	

Member
[[10271,SPLIR,2013-08-01, ...],[10422,FRANS,2014-01-22, ...]]
(null)
[[10674,ISLAT,2014-09-18, ...]]
[[10900,WELLI,2015-02-20, ...],[11051,LAMAI,2015-04-27, ...],[11057,NORTS,201...

A2中为四个线程各自返回的结果

订单ID	客户	订购日期	员工ID	ID	订单金额
10271	SPLIR	2013-08-01	6	24	48.0
10422	FRANS	2014-01-22	2	175	49.8
10674	ISLAT	2014-09-18	4	427	45.0
10900	WELLI	2015-02-20	1	653	45.0
11051	LAMAI	2015-04-27	7	804	45.0
11057	NORTS	2015-04-29	3	810	45.0

A4中将四个线程的结果合并

并行分组举例：对大文本文件（销售记录表.txt），按客户、订购年份分组，统计订单总金额和订单数

	A	B
1	4	
2	fork to(A1)	=file("销售记录表.txt").cursor@t(;A2:A1)
3		=B2.groups(客户,year(订购日期):年份;sum(订单金额):总金额,count(~):订单数)
4	=A2.conj()	
5	=A4.groups(客户,年份;sum(总金额):总金额,sum(订单数):订单数)	

客户	年份	总金额	订单数
WHITC	2013	3532.0	2
WOLZA	2013	459.0	1
ANATR	2014	479.75	1
ANTON	2014	5119.75	3
AROUT	2014	2142.9	1
BERGS	2014	6930.59...	4

A4合并各线程的结果，但并不是最终的分组结果

客户	年份	总金额	订单数
ALFKI	2015	538.7	1
ANATR	2013	88.8	1
ANATR	2014	1129.75	3
ANATR	2015	1369.4	2
ANTON	2013	403.2	1
ANTON	2014	6452.15	5

A5对合并后的分组结果再次分组聚合才是正确结果

Excel

直接支持Excel文件解析，简化开发易于维护

举例：对订单金额大于2000的记录，按员工部门分组求总金额

	A	B
1	=file("销售记录表.xlsx").importxls@t()	/解析Excel
2	=file("人员表.xlsx").importxls@t(编号,部门).keys(编号)	/解析Excel，设主键
3	=A1.select(订单金额>2000)	/过滤
4	>A3.switch(员工ID,A2)	/外键关联
5	=A3.groups(员工ID.部门:部门;sum(订单金额):总金额)	/外键表字段分组
6	=file("result.xlsx").exportxls@t(A5)	/结果导出Excel

	A	B	C	D	E	F
1	订单ID	客户	订购日期	员工ID	ID	订单金额
2	10400	EASTC	1/1/2014	1	153	3063
3	10401	RATTC	1/1/2014	1	154	3868.6
4	10402	ERNSH	1/2/2014	8	155	2713.5

订单ID	客户	订购日期	员工ID	ID	订单金额
10400	EASTC	2014-01-01	1	153	3063
10401	RATTC	2014-01-01	1	154	3868.6
10402	ERNSH	2014-01-02	8	155	2713.5

A1:解析Excel文件"销售记录表"

订单ID	客户	订购日期	员工ID	ID	订单金额
10400	EASTC	2014-01-01	1	153	3063
10401	RATTC	2014-01-01	1	154	3868.6
10402	ERNSH	2014-01-02	8	155	2713.5

编号	部门
8	销售部

A3、A4:过滤主数据并关联外键表

	A	B	C	D	E	F	G	H	I
1	编号	部门	姓名	性别	出生日期	入职日期	籍贯	工资	应发工资
2	1	综合部	肖梅	1	1968-12-08	1992-05-01	沈阳	8000	8500
3	2	研发部	李四	1	1962-02-19	1992-08-14	大连	9000	9500
4	3	销售部	李芳	0	1973-08-30	1992-04-01	营口	10000	10500

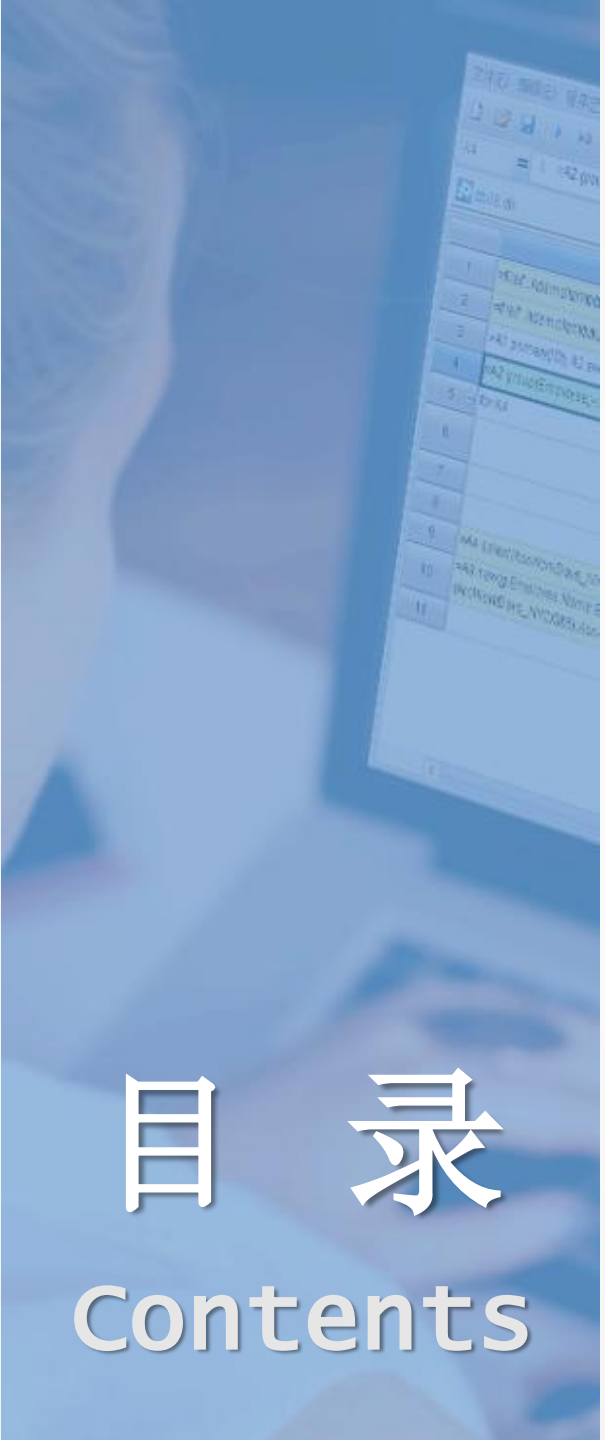
编号	部门
1	综合部
2	研发部
3	销售部

A2:解析Excel文件“人员表“,设编号为主键
这里仅读取用到的编号和部门两个字段

部门	总金额
研发部	132256.39
综合部	156257.72999999998
销售部	185208.33999999997

	A	B
1	部门	总金额
2	研发部	132256.39
3	综合部	156257.73
4	销售部	185208.34

A5、A6：按外键表的部门分组求和并导出



目 录

Contents

并行取数

数据库JDBC性能较差，可借助集算器中多线程并行解决该问题

举例：根据表T的数据字段part将数据分为四个部分，每个线程读取一部分数据

	A	B	C
1	fork 4	=connect(db)	/分4线程，要分别建立连接
2		=B1.query@x("select * from T where part=? ",A1)	/分别取每一段
3	=A1.conj()		/合并结果

困难计算

支持批量结构化运算和有序集合，实现跨行计算较为轻松

举例：某支股票最长连续涨了多少交易日

```
SELECT MAX(连续日数)
FROM (
        SELECT COUNT(*) AS 连续日数
        FROM (
                SELECT SUM(涨跌标志) OVER (ORDER BY 交易日) AS 不涨日数
                FROM (
                        SELECT 交易日
                                , CASE
                                        WHEN 收盘价 > LAG(收盘价) OVER (ORDER BY 交易日) THEN 0
                                        ELSE 1
                                END AS 涨跌标志
                        FROM 股价表
                )
        )
        GROUP BY 不涨日数
)
```

	A	B
1	=demo.query(“select * from 股价表”).sort(交易日)	
2	=0	/临时变量记录连续涨了几天
3	=A1.max(A2=if(收盘价>收盘价[-1],A2+1,0))	/比较前日决定A2是否继续加

批量查找

子查询的结果作为参数再次查询，反复多次查询影响效率

举例：找出销售额占到一半的前n个客户，并按销售额从大到小排序

```
WITH A AS (
        SELECT 客户, 销售额, row_number() OVER (ORDER BY 销售额) AS 排名
        FROM 客户销售表
        )
SELECT 客户, 销售额
FROM (
        SELECT 客户, 销售额, SUM(销售额) OVER (ORDER BY 排名) AS 累计额
        FROM A
)
WHERE 累计额 > (
        SELECT SUM(销售额) / 2
        FROM 客户销售表
)
ORDER BY 销售额 DESC
```

	A	B
1	=demo.query("select * from 客户销售表").sort(销售额:-1)	
2	=A1.cumulate(销售额)	/计算累计序列
3	=A2.m(-1)/2	/最后的累计值即是总和
4	=A2.pselect(~>=A3)	/超过一半的位置
5	=A1(to(A4))	

数据库切换

各数据库有着不同的函数写法，数据库切换后需要重写SQL

举例：将标准SQL中”当年中第几星期”对各类数据库进行SQL翻译

标准函数	含义	oracle	sql server	db2	mysql	teradata	hsql	PostgresSQL
WEEKOFYEAR(d)	当年中第几星期	TO_NUMBER(TO_CHAR(d,'WW'))	DATEPART(WW,d)	WEEK(d)	WEEK(d)	TD_WEEK_OF_YEAR(d)	WEEK(d)	EXTRACT(WEEK FROM d)

	A
1	SELECT ID,WEEKOFYEAR(DATES),CUSTOMER,AREA FROM CLUE
2	=A1.sqltranslate("ORACLE")
3	=A1.sqltranslate("SQLSVR")
4	=A1.sqltranslate("MYSQL")

Value
SELECT ID,TO_NUMBER(TO_CHAR(DATES,'WW')),CUSTOMER,AREA FROM CLUE

A2为order对应的SQL语法

Value
SELECT ID,DATEPART(WW,DATES),CUSTOMER,AREA FROM CLUE

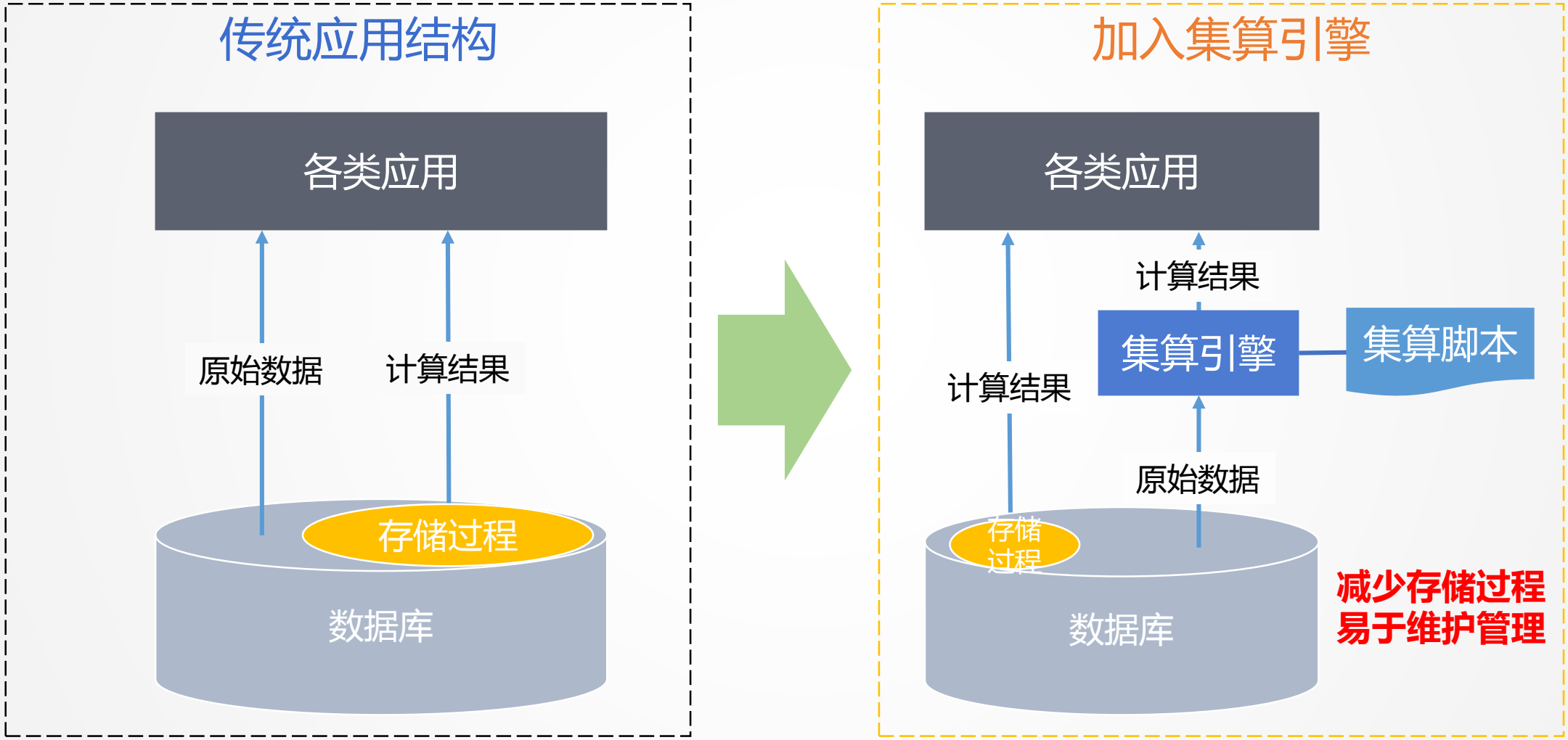
A3为sql server对应的SQL语法

Value
SELECT ID,WEEK(DATES),CUSTOMER,AREA FROM CLUE

A4为mysql对应的SQL语法

各类数据库均有自己的**字符串函数,数值函数,时间函数,转换函数,任意函数。**
对这些不同标准进行统一处理，将为开发维护带来极大的便利。

库外存储过程--结构图


传统应用结构
各类应用
原始数据
计算结果
存储过程
数据库
加入集算引擎
各类应用
计算结果
集算引擎
集算脚本
计算结果
原始数据
存储过程
数据库
减少存储过程
易于维护管理

库外存储过程

数据库中不便修改存储过程时，可借助集算器实现库外计算模块

举例：某开发团队的计算目标是：指定一年中的某个时间段，计算该时间段内每个州的销量均在前二十名的优势产品有哪些？统计出这些优势产品每个月的销量，以及每个月比上个月的销量增长率。

	A	B	C
1	=ora.query("select State,Product,Amount,Time from sales where to_char(Time,'YYYYMMDD')>=? and to_char(Time,'YYYYMMDD')<=? ";start,end)		/根据时间参数从sales table取数据
2	=A1.group(State)	=A2.(~.rank(Amount).pselect@a(~<=20))	/A2：把A1中的数据按state进行分组。B2：在A2的每个组（州）里，求出销量前20的产品的记录序号。
3	=A2.(~(B2(#)).(Product))		/用序号从A2取出每个州对应的产品
4	=A3.isect()		/求A3中各组产品的交集。这里的计算结果就是优势产品
5	=A1.select(A4.pos(Product)>0)		/过滤出优势产品的销量记录
6	=A5.groups(Product,month(Time):Month;sum(Amount):MonthAmount)		/根据A5，汇总出每种产品每个月的销量
7	=A6.derive(Rate)		/为A6增加一个字段Rate，以便将来存储增长率
8	=A7.group(Product)		/将A7按产品重新分组
9	=A8.(~.run(Rate=MonthAmount[-1]/MonthAmount-1))		/计算每种产品每个月比上个月的增长率
10	=A9.union()		/将A9中的分组数据合并。这就是最终计算目标
11	return A10		/返回结果A10

多数据库混算

多数据库之前的数据难以实现数据关联

举例：列出雇员所属部门为市场部和财务部的订单明细（雇员表与订单表分别在两个不同的数据库中）

	A
1	=mysql.query("select OID,EID,AMOUNT from orders")
2	=pg.query("select EID,NAME,DEPT from employee")
3	=A1.switch(EID,A2:EID)
4	=A3.select(["Marketing","Finance"].pos(EID.DEPT))
5	=A4.new(OID,EID.EID:EID,EID.NAME:NAME,EID.DEPT:DEPT,AMOUNT)

OID	EID	AMOUNT
10248	5	428.0
10249	6	1842.0
10250	4	1523.4999898076...
10251	3	624.94999976083...

A1为MySQL中数据

EID	NAME	DEPT
1	Rebecca	R&D
2	Ashley	Finance
3	Rachel	Sales
4	Emily	HR

A2为PG中数据

OID	EID	AMOUNT
10248	[5,Ashley,R&D]	428.0
10249	[6,Matthew,Sales]	1842.0
10250	[4,Emily,HR]	1523.49...
10251	[3,Rachel,Sales]	624.949...

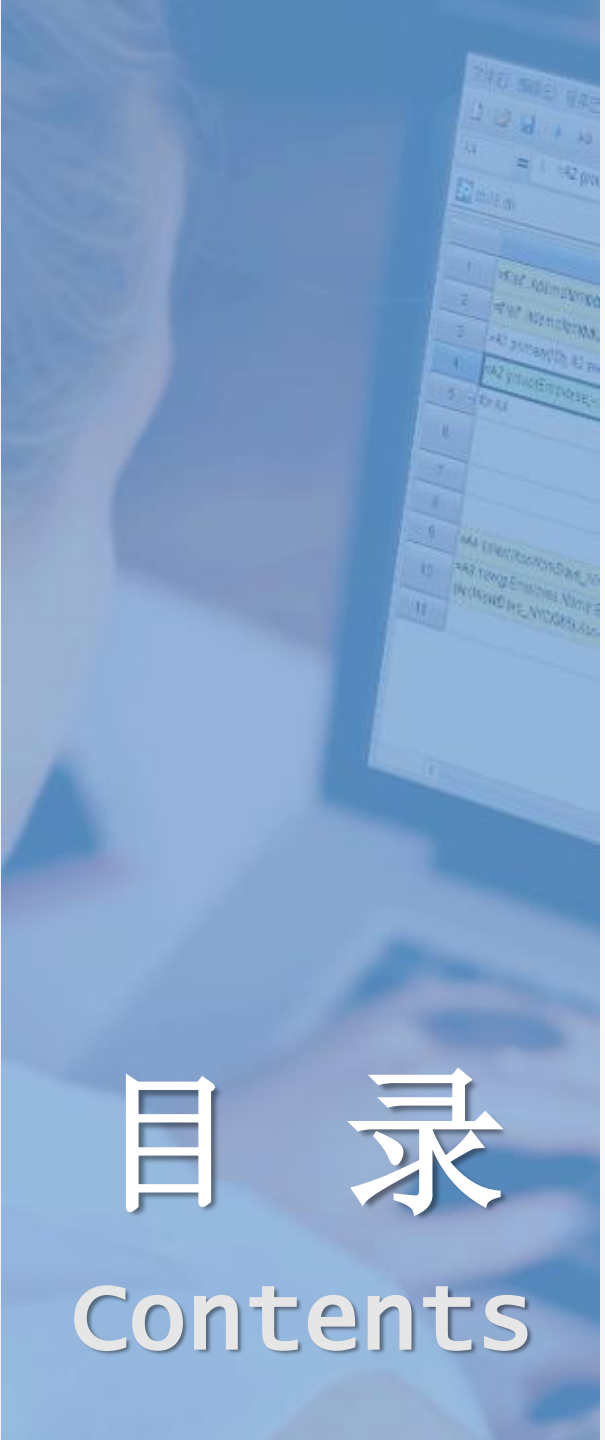
A3将两部分数据
按雇员ID关联

OID	EID	AMOU...
10262	[8,Megan,Marketing]	583.19...
10265	[2,Ashley,Finance]	1170.0
10268	[8,Megan,Marketing]	1098.0
10276	[8,Megan,Marketing]	400.0

A4过滤出雇员所属部门
为市场部和财务部订单

OID	EID	NAME	DEPT	AMOUNT
10262	8	Megan	Marketing	583.19...
10265	2	Ashley	Finance	1170.0
10268	8	Megan	Marketing	1098.0
10276	8	Megan	Marketing	400.0

A5中取出报表需要展现
的数据字段



目 录

Contents

NoSQL/Hadoop

JDBC性能较差或数据库本身不支持JDBC时，可以使用外部库功能。

1、部署外部库：

从网上下载以下两个文件（请根据实际所用版本下载对应的jar文件），放到产品外部库文件夹中；Mongo外部库文件路径为：安装目录\esProc\extlib\MongodbCli；其中润乾核心jar为mongoCli.jar。

bson-3.6.3.jar

mongo-java-driver-3.6.3.jar

2、访问MongoDB数据库：

可使用的外部库函数有mongo_open()、mongo_shell()、mongo_close()。

	A	B
1	=mongo_open("mongodb://localhost:27017/mydb")	/连接mongo server的mydb库
2	=mongo_shell(A1,"emp.find()").fetch()	/查询mydb库中的emp集合中的记录
3	=mongo_close(A1)	/关闭数据库连接

外部库还可以分别实现访问阿里云、elasticsearch、hive、spark、hbase、redis、cassandra、informix数据库、读取报表文件、连接hdfs文件系统、多维数据库、webservice、ftp、sap、kafka系统

json解析与生成

直接支持复杂多层json解析与生成，提高Java对json的计算能力

举例：对销售数据文件sales.json（不同结构的多层JSON），查询2017和2018美国和加拿大的销售数据

json(x)
当x是json格式串时，将x解析成序表返回；当x是记录或序列时，解析成json格式串返回

	A	B
1	=json(file("sales.json").read())	=A1.select([2017,2018].contain(YEAR))
2	=B1.news(MONTHLY_SALES;B1.YEAR:YEAR,B1.MONTH:MONTH,#1,#2)	=A2.select(["USA","Canada"].contain(COUNTRY))
3	=B2.news(NATIONAL_MONTHLY_SALES;B2.YEAR:YEAR,B2.COUNTRY:COUNTRY,ORDERNUMBER,QUANTITYORDERED,CUSTOMERNAME,PHONE,ADDRESSLINE1,ADDRESSLINE2)	

[{"YEAR":2016,"MONTH":1,"MONTHLY_SALES":[{"COUNTRY":"Germany","NATIONAL_MONTHLY_SALES":[{"ORDERNUMBER":10101,"QUANTITYORDERED":25,"PRICEEACH":100,"ORDERLINENUMBER":4,"SALES":3782,"ORDERDATE":"1/9/2016 0:00","STATUS":"Shipped","QTR_ID":1,"PRODUCTLINE":"Vintage Cars","MSRP":127,"PRODUCTCODE":"S18_2325","CUSTOMERNAME":"Blauer See Auto, Co.","PHONE":"+49 69 66 90 2555","ADDRESSLINE1":"Lyonerstr. 34","ADDRESSLINE2":"","CITY":"Frankfurt","STATE":"","POSTALCODE":60528,"TERRITORY":"EMEA","CONTACTLASTNAME":"Keitel","CONTACTFIRSTNAME":"Roland","DEALSIZE":"Medium"},{"ORDERNUMBER":10101,"QUANTITYORDERED":26,"PRICEEACH":100,"ORDERLINENUMBER":1,"SALES":3773.38,"ORDERDATE":"1/9/2016 0:00","STATUS":"Shipped","QTR_ID":1,"PRODUCTLINE":"Vintage Cars","MSRP":168,"PRODUCTCODE":"S18_2795","CUSTOMERNAME":"Blauer See Auto, Co.","PHONE":"+49 69 66 90 2555","ADDRESSLINE1":"Lyonerstr. 34","ADDRESSLINE2":"","CITY":"Frankfurt","STATE":"","POSTALCODE":60528,"TERRITORY":"EMEA",

YEAR	MONTH	MONTHLY_SALES
2017	1	[[France,[10211,41,100, ...
2017	2	[[Australia,[10223,37,10...
2017	3	[[France,[10227,25,100, ...
2017	4	[[Canada,[10235,24,76....
2017	5	[[Finland,[10247,44,100,...
2017	6	[[Canada,[10261,27,100...

COUNTRY	NATIONAL_MONTHLY_SALES
France	[[10211,41,100, ...],[10211,41,1...
Japan	[[10210,23,100, ...],[10210,34,1...
Spain	[[10212,39,100, ...],[10212,33,1...
UK	[[10213,38,94.79, ...],[10213,25...
USA	[[10215,35,100, ...],[10209,39,1...

B1过滤A1解析json后年份为2017、2018的销售记录

YEAR	MONTH	COUNTRY	NATIONAL_MONTHLY_...
2017	1	France	[[10211,41,100, ...],[102...
2017	1	Japan	[[10210,23,100, ...],[102...
2017	1	Spain	[[10212,39,100, ...],[102...
2017	1	UK	[[10213,38,94.79, ...],[1...
2017	1	USA	[[10215,35,100, ...],[102...

A2将B1过滤后的结果进一步展开，形成：年、月、国家、[明细]的二维表

YEAR	MONTH	COUN...	NATIONAL...
2017	1	USA	[[10215,35,...
2017	2	USA	[[10222,49,...
2017	3	USA	[[10228,29,...
2017	4	Canada	[[10235,24,...
2017	4	USA	[[10237,23,...

过滤B2结果中国家为USA、Canada的记录

YEAR	COUNTRY	ORDERNU...	QUANTITY...	CUSTOME...	PHONE	ADDRESS...	ADDRESS...
2017	Canada	10235	34	Royal Can...	(604) 555-...	23 Tsawas...	
2017	USA	10237	23	Vitachrome...	2125551500	2678 King...	Suite 101
2017	USA	10236	22	Motor Mint ...	2155559857	11328 Dou...	
2017	USA	10237	39	Vitachrome...	2125551500	2678 King...	Suite 101
2017	USA	10237	32	Vitachrome...	2125551500	2678 King...	Suite 101

最终将满足要求的结果展开成二维表

xml解析与生成

直接支持xml解析与生成，提高Java对xml的计算能力

举例：将订单记录（多层序表）转为多层xml

	A	B
1	=xml@s(orders_detail)	/将序表解析为多层的xml返回

订单ID	客户	订购日期	员工ID	ID	订单金额
10248	VINET	2013-07-04	5	1	2440.0
10249	TOMSP	2013-07-05	6	2	1863.4
10250	HANAR	2013-07-08	4	3	1813.0
10251	VICTE	2013-07-08	3	4	670.8
10252	SUPRD	2013-07-09	4	5	3730.0

orders_detail

编号	部门	姓名	性别	出生日期	入职日期	籍贯	工资	应发工资
5	研发部	赵军	1	1965-03-04	1993-10-17	沈阳	4000	4500

```
<?xml version="1.0" encoding="utf-8"?>
<xml>
  <row>
    <订单ID>10248</订单ID>
    <客户>"VINET"</客户>
    <订购日期>2013-07-04</订购日期>
    <员工ID>
      <row>
        <编号>5</编号>
        <部门>"研发部"</部门>
        <姓名>"赵军"</姓名>
        <性别>1</性别>
        <出生日期>1965-03-04</出生日期>
        <入职日期>1993-10-17</入职日期>
        <籍贯>"沈阳"</籍贯>
        <工资>4000</工资>
        <应发工资>4500</应发工资>
      </row>
    </员工ID>
    <ID>1</ID>
    <订单金额>2440.0</订单金额>
  </row>
  <row>
    <订单ID>10249</订单ID>
    <客户>"TOMSP"</客户>
```

orders_detail_xml

举例：将订单记录（多层xml）按指定层的内容返回序表

	A	B
1	=xml(orders_detail_xml, "xml/row/员工ID/row")	/从多层订单记录中去员工信息列表

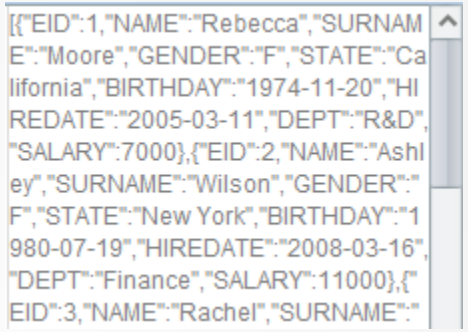
编号	部门	姓名	性别	出生日期	入职日期	籍贯	工资	应发工资
5	研发部	赵军	1	1965-03-04	1993-10-17	沈阳	4000	4500
6	销售部	孙林	1	1967-07-02	1993-10-17	大连	5500	6000
4	综合部	郑建杰	1	1968-09-19	1993-05-03	本溪	6000	6500
3	销售部	李芳	0	1973-08-30	1992-04-01	营口	10000	10500
4	综合部	郑建杰	1	1968-09-19	1993-05-03	本溪	6000	6500

xml(x,s)，其中**s**表示要取出的层标识，多层用/分隔，空表示从根开始取

http访问与发布

将url的返回结果封装成文件流返回

举例：解析某servlet返回的json格式的员工数据，按所在州统计人数，将结果数据作为http服务发布

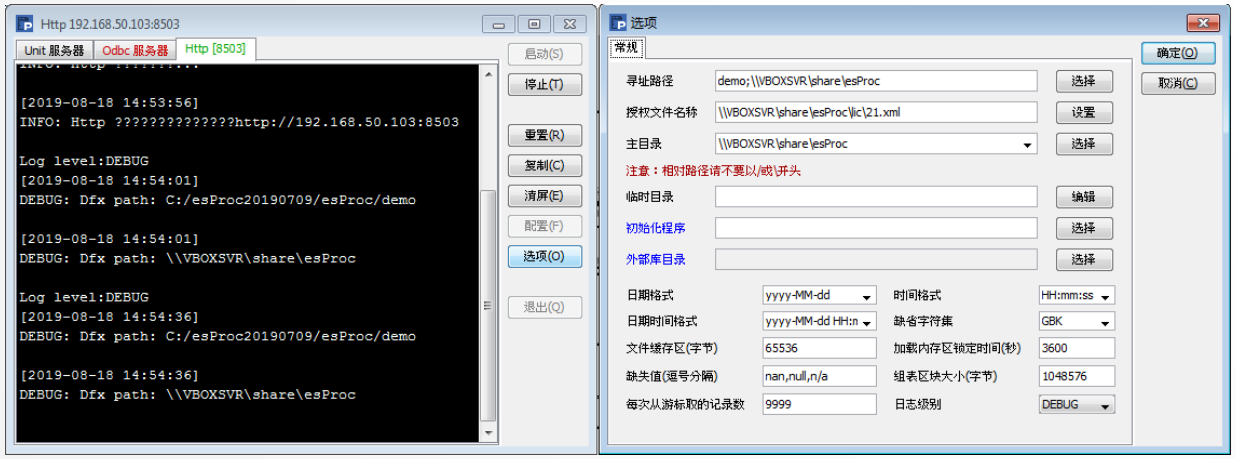


Json格式的员工信息

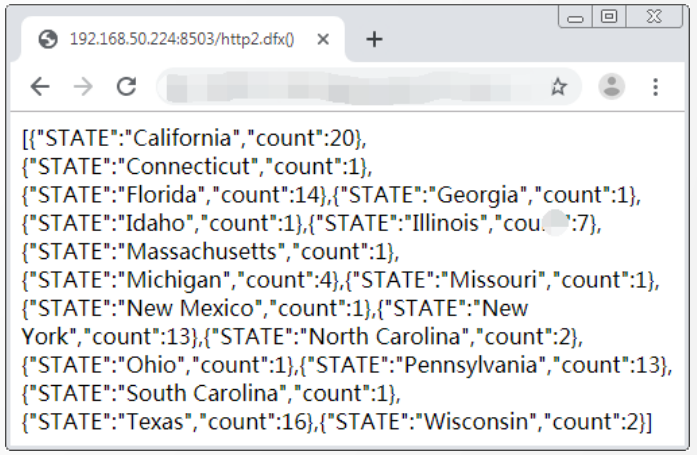
	A	B
1	=httpfile("http://192.168.0.131:6080/myweb/servlet/testServlet?table=employee&type=json").read()	/访问并读取servlet返回的文件流
2	=json(A1).groups(STATE;count(~):count)	/解析json并按州统计人数
3	return json(A2)	/返回json格式的统计结果

STATE	count
California	20
Connecticut	1
Florida	14
Georgia	1

解析json后分组计数



http服务的发布与配置界面



将结果转为json后发布http服务

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
1.53%
近1天
0.95%
近6小时
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