润乾
RAQSOFT

高性能内存数据库

集算器应用场景实施方案

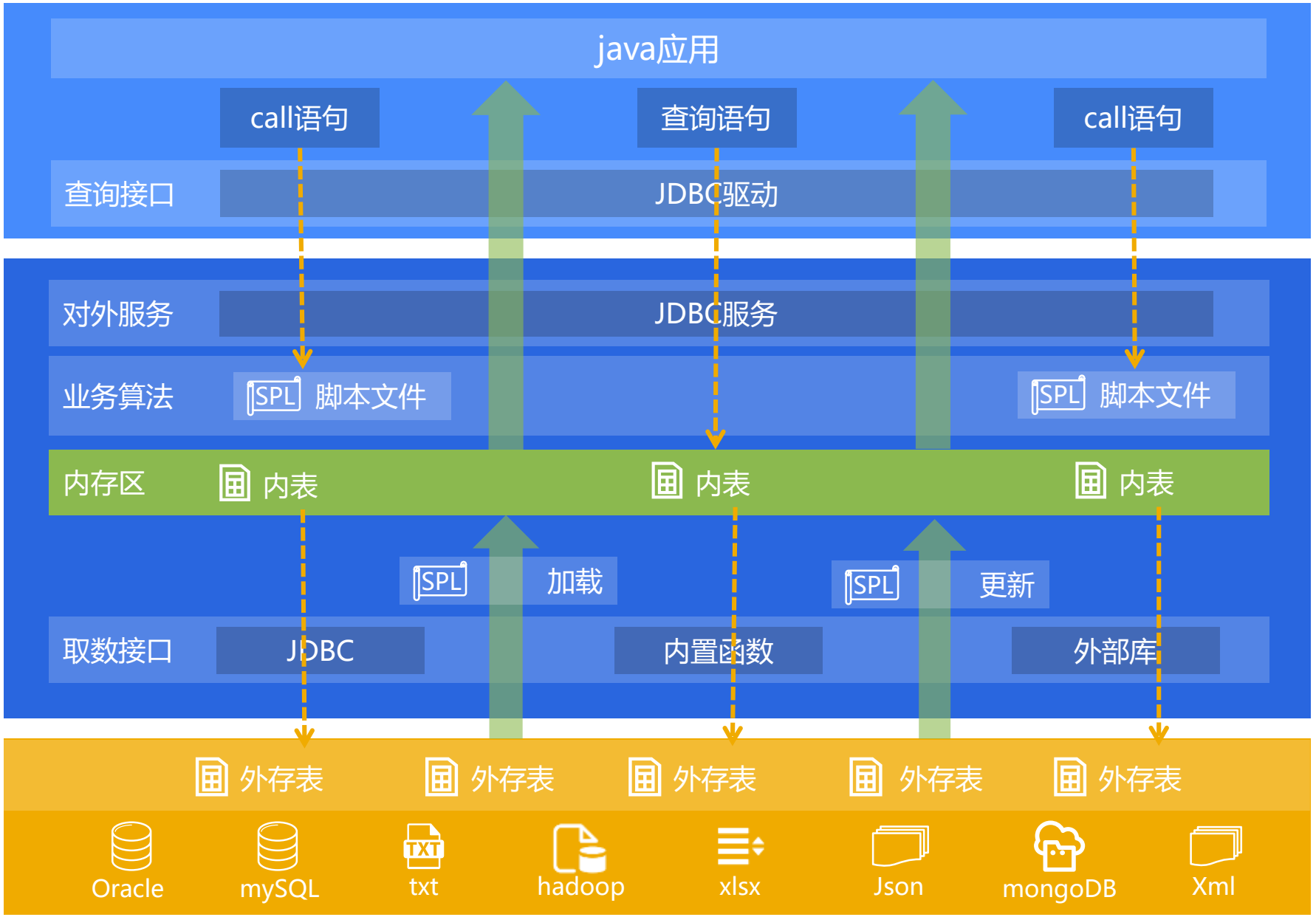
www.raqsoft.com.cn

架构

前端应用

内存数据库

源数据



优势

集算器底层能力	内存数据库特性	优势
指针式复用 指针式引用	支持数据复用 支持预关联	降低内存占用 提高计算性能
内表压缩 序号化、排号键	内存容纳更多数据	降低硬件成本
多源混算	支持冷热数据路由	充分利用硬件性能 降低架构复杂度
支持离散数据集、有序集合、过程计算、高性能算法	简化复杂计算	提高开发效率

目录 CONTENTS

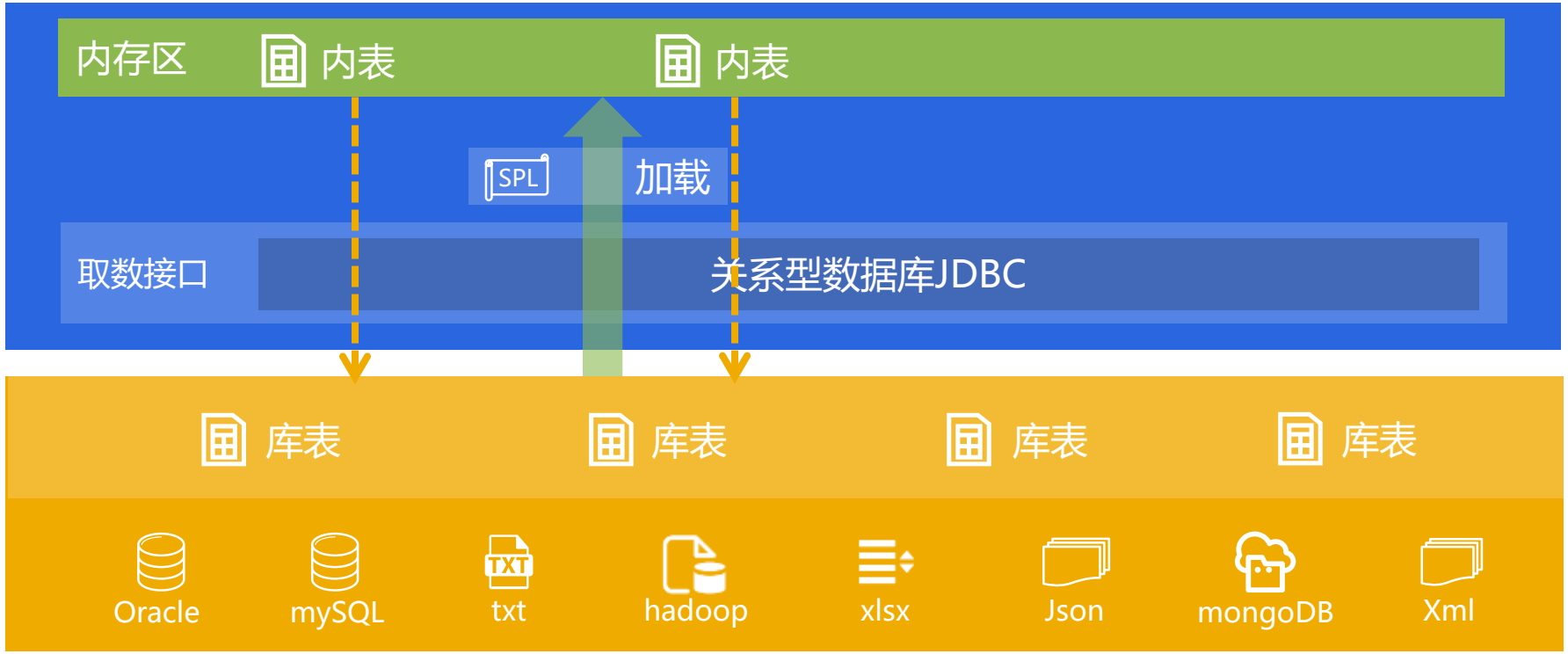
01

数据加载与共享

部署架构

内存数据库通常独立部署，启动时从源数据（常见的比如RDB）取数，并加载到内存中

加载动作通常由一个SPL脚本自动执行



数据加载 1.编写脚本文件

下面是Oracle数据库中的两张表

orders

orderID(key)	client	seller	orderDate	freight
10248	VINET	5	2012-07-04	32.38
10249	TOMSP	6	2012-07-05	11.61
10250	HANAR	4	2012-07-08	65.83
10251	VICTE	3	2012-07-08	41.34
10252	SUPRD	4	2012-07-09	51.3
...	...		...	...

orderDetails

orderID(key/fk)	product(key)	price	amount
10248	17	14	12
10248	42	9	10
10248	72	34	5
10249	14	18	9
10249	51	42	40
...	...	...	...

编写SPL脚本initData.dfx，将orders和orderDetails加载到内存中，数据应按键排序

	A	B	C
1	=connect@l("orcl")		/连接oracle
2	=A1.cursor("select orderid,client,seller,orderdate,freight from order orders by orderid")	=A1.cursor@x("select orderid,product,price,amount from orderdetail orders by orderid,product")	/游标查询数据库
3	=A2.memory(orderid)	=B2.memory(orderid,product)	/根据主键将游标转为内表
4	>env(orders,A3)	>env(orderdetails,B3)	/共享内表，允许其他程序用变量名访问
5	=output("initData.dfx sucessed!")		/额外的控制台信息

数据加载　2.配置加载脚本

在集算器配置文件raqsoftConfig.xml中指定initData.dfx，以便启动时自动执行

```
…
<Init>
　　<dfx>init/initData.dfx</dfx>
</Init>
…
```

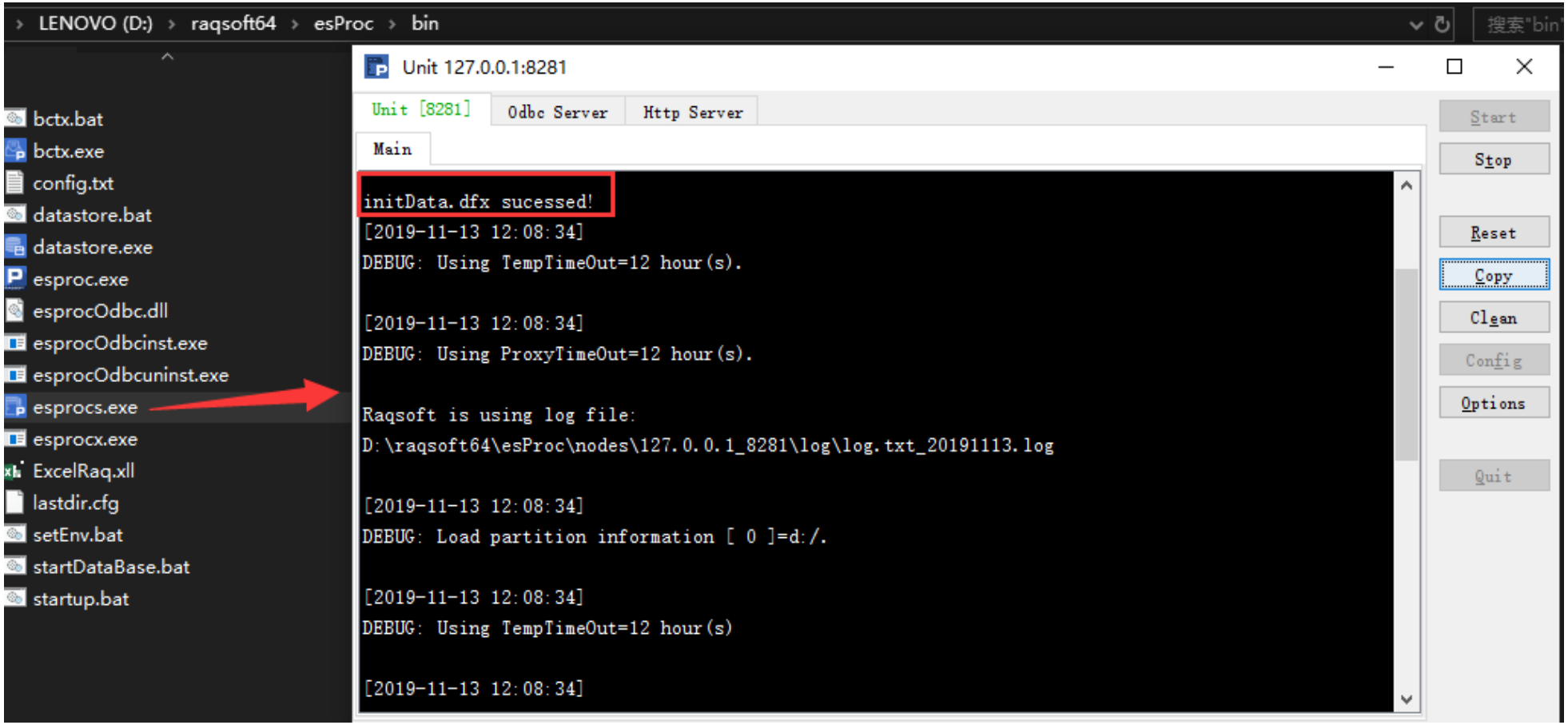
说明1：上述路径相对于mainPath，也可写绝对路径，注意linux和Windows路径的区别

说明2：源数据（关系型数据库）同样要定义在raqsoftConfig.xml，比如例子中用到的Oracle数据库orcl，详见

http://doc.raqsoft.com.cn/esproc/tutorial/pzraqsoftconfig.html

数据加载　3.启动内存数据库，自动执行加载脚本

执行集算器目录下的esprocs.exe(Linux下为esprocs.sh)，点击start按钮以启动SPL Server。



说明：关于SPL Server详细配置（比如地址、端口号），请参考http://doc.raqsoft.com.cn/esproc/tutorial/fuwuqi.html

加载非关系型数据

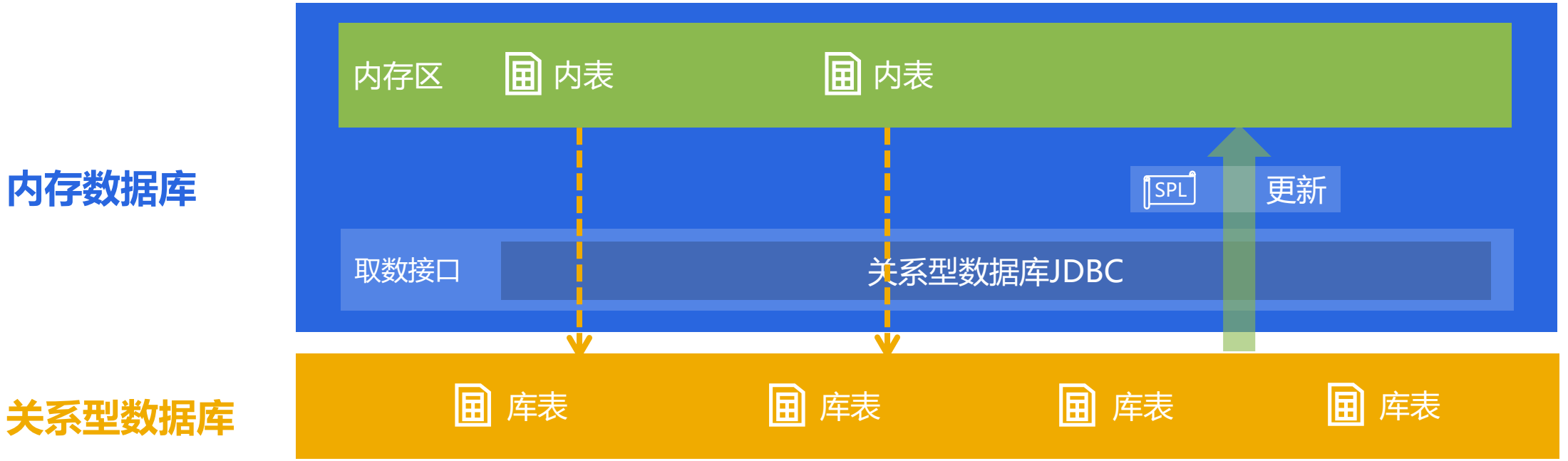
集算器可加载任意数据源，包括但不限于文本文件、集算器简表或组表、hive、mongoDB

例如：从文本文件orders.txt和orderdetails.txt加载数据

	A	B	C
1	=file(“orders.txt”).cursor@t().sortx(orderid)	=file(“orderdetails.txt).cursor@t().sortx(orderid,product)	/用游标读取文本
2	=A2.memory(orderid)	=B2.memory(orderid,product)	/根据主键将游标转为内表
3	>env(orders,A3)	>env(orderdetails,B3)	/共享内表，允许其他程序用变量名访问

数据更新

根据实时性要求，使用SPL脚本定时将库表更新到内存中



数据更新

少量变更：获取数据的变更信息，使用update/delete函数在内表上执行更新操作

获得变更新信息的方法有很多，下面以标记法为例，比如：库表orderChange记录着库表order的变更信息，flag=delete时表示本记录应删除，flag=update表示本记录是新增或修改的记录。则更新内存的脚本如下：

	A	B	C
1	=orderCP=orders.derive()	/用变量名直接引用共享内存，并复制一份	
2	=connect@el("orcl")		/启用数据库事务处理
3	=orderDelete=A2.query("select orderid,client,seller,orderdate,freight from orderChange where flag='delete'")	=orderCP.delete(orderDelete)	/取删除的数据，并变更到内表
4	=orderUpdate=A2.query("select orderid,client,orderdate,freight from orderChange where flag='update'")	=orderCP.update(orderUpdate)	/取更新的数据，并变更到内表
5	=A2.execute@k("delete from orderChange")		/清空标记表
6	if A2.error()==0	=A2.commit()	/事务正常提交
7		=env(orders,orderCP)	/重新共享内表
8	else	=A2.rollback()	/事务失败则回滚
9	=A2.close()		

数据变更

大量变更：变更的数据量较大时，内表需全部重建（即执行加载脚本，替换内表）。

（内表采用紧致存储方式，修改信息会单独存储，计算时再合并。更新数据较少时，修改动作对性能影响不大，如果更新信息过大时则会影响性能，不如全部重建）

重建内表，最简单的办法是重启内存数据库

如果希望热切换变更数据，可以通用集算器客户端，使用命令行执行内存数据库上的加载脚本initData.dfx，具体命令如下：

```
Windows命令行：esprocx.exe –r =callx(\“d:/raqsoft64/temp/init/initData.dfx\";[\"127.0.0.1:8281\"])
Linux命令行：    esprocx.sh –r =callx(“/opt/raqsoft64/temp/init/initData.dfx";["127.0.0.1:8281"])
```

说明：
1.不能利用原内存数据库服务的命令行，否则每次启动命令行都会自动执行initData.dfx，在新进程生成无用的内存数据
2.上述脚本中的”IP:端口”指向内存数据库，请根据实际情况修改
3.自动热切换，即定时执行加载脚本，可使用操作系统自带调度工具，如Windows计划任务、Linux Crontab命令，或第三方可视化工具如opencron。
4.用热切换的方法，短时间内会占用两倍内存，重启内存数据库则无此问题。

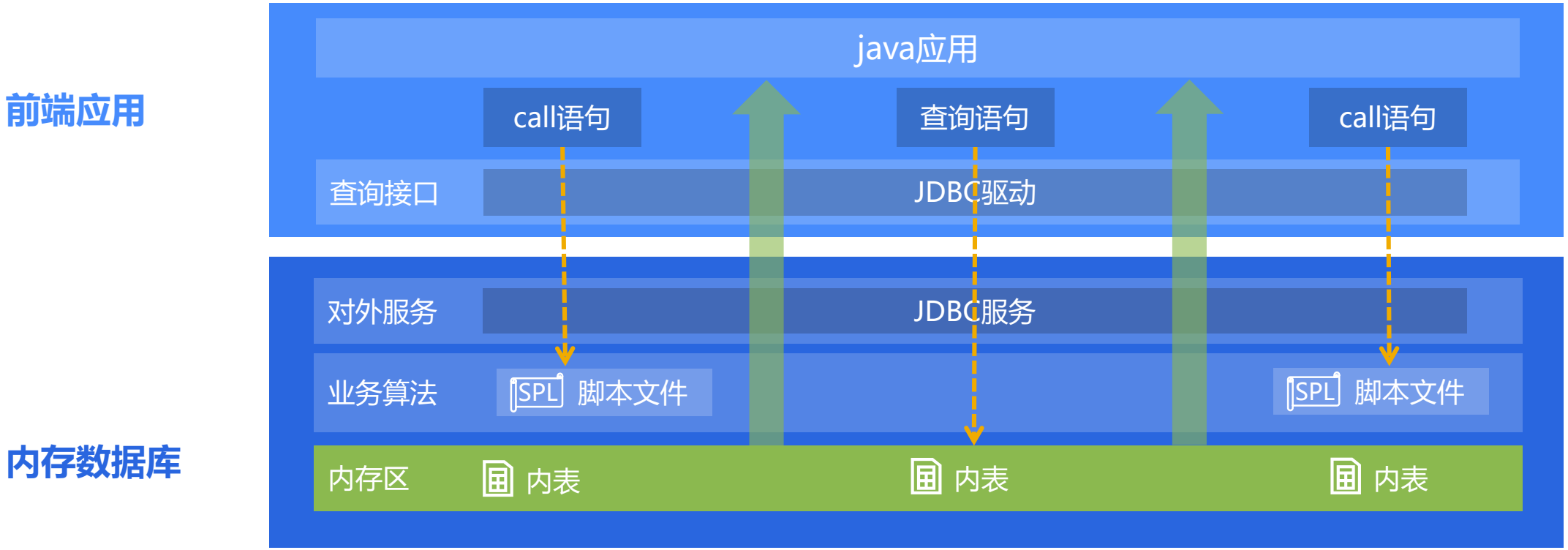
目录 CONTENTS

02

常规运算

前端部署

通常情况下，应在java应用中部署集算器JDBC驱动，在JAVA代码中通过call语句调用内存数据库中的脚本文件，由脚本文件完成内存计算



说明：部署JDBC驱动请参考http://doc.raqsoft.com.cn/esproc/tutorial/jdbcbushu.html

前端部署

1.在内存服务器已启动，且已将订单表加载到内存的前提下，编写用于业务算法的脚本文件

统计某客户（参数名pCustID）每年每月的订单数量，脚本run.dfx如下

	A	B
1	=orders.select@m(client==pCustID)	/用名字引用共享内表，进行多线程查询
2	=A1.groups(year(orderdate):filedyear, month(orderdate):fieldmonth; count(1):quantity)	/分组汇总

也可以合写为一步

	A	B
1	=orders.select@m(client==pCustID).groups(year(orderdate):filedyear, month(orderdate):fieldmonth; count(1):quantity)	

前端部署

2.在JDBC驱动的配置文件raqsoftConfig.xml中指明内存数据库的地址

```
...
<Units>
    <Unit>192.168.0.197:8281</Unit>
</Units>
...
```

说明：详细配置参见http://doc.raqsoft.com.cn/esproc/tutorial/pzraqsoftconfig.html

前端部署

3.在java程序中调用run.dfx，类似调用存储过程

```
…
Class.forName("com.esproc.jdbc.InternalDriver");
con=DriverManager.getConnection("jdbc:esproc:local://onlyServer=true");
PreparedStatement pstmt = con.prepareStatement( “call run(?)” );
pstmt.setObject(1, “AYWYN” ) ;
ResultSet rs=pstmt. executeQuery();
…
```

说明：JAVA报表调用脚本文件时，与调用数据库存储过程类似，详见

http://c.raqsoft.com.cn/article/1560233466960

前端部署

算法比较简单时，也可不写脚本文件，在java程序中用SPL语句直接访问内存

将run.dfx改写为SPL语句

```
…
Class.forName("com.esproc.jdbc.InternalDriver");
con=DriverManager.getConnection("jdbc:esproc:local://onlyServer=true");
PreparedStatement pstmt =
con.prepareStatement( “=orders.select@m(client==\ “AYWYN\").groups(year(orderdate):filedyear,
month(orderdate):fieldmonth;count(1):quantity)” );
ResultSet rs=pstmt. executeQuery();
…
```

前端部署

熟悉SQL语句的程序员，也可以用集算器SQL语句访问内存

将run.dfx改写为SQL语句

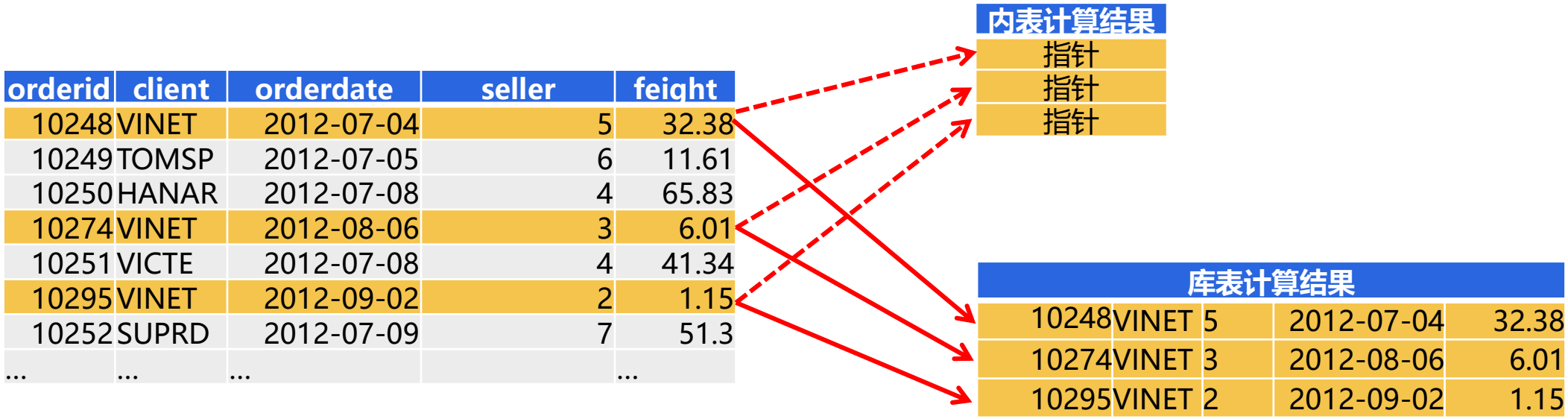
```
…
Class.forName("com.esproc.jdbc.InternalDriver");
con=DriverManager.getConnection("jdbc:esproc:local://onlyServer=true");
PreparedStatement pstmt = con.prepareStatement( “select year(orderdate) AS fieldyear,month(orderdate) AS fieldmonth, count(1) AS quantity from orders where client=? group by year(orderdate), month(orderdate)” );
pstmt.setObject(1, “AYWYN” ) ;
ResultSet rs=pstmt. executeQuery();
…
```

说明：关于集算器SQL语句，可参考http://doc.raqsoft.com.cn/esproc/func/sqljia.html

指针式复用

集算器内存计算时，数据以指针的形式参与计算，多步骤计算和不同的算法都只是复用同一个内表，而不是像关系型数据库表那样每次要复制记录

比如，执行同样的算法select * from orders where client= “VINET” ，内表和库表对内存的利用方式如下：



指针式复用可显著节省内存，而且比记录式复制速度更快，但要注意的是，修改计算结果等同于修改原数据，这是指针的性质决定的。

如果需要修改计算结果，又不想影响原数据（比如再次复用），则应当用derive或new函数复制一份原数据。

计算举例

集算器支持完备的结构化算法数据算法，下面再举几例

高性能键值查询

参数pOrderList代表订单编号列表（比如[1,3,4]），按pOrderList查询内表orders的记录

	A
1	=orders.find@k(pOrderList)

集合运算

找出2018年运货费总额较少（小于200000），而2017年较多（大于200000）的客户名单

	A	B
1	=orders.select(year(orderdate)==2018).groups(client;sum(freight):sAmount).select(sAmount<200000)	/2018运货费较少的客户
2	=orders.select(year(orderdate)==2017.groups(client;sum(freight):sAmount).select(sAmount>200000)	/2017运货费较多的客户
3	=A1.(client) ^ A2.(client)	/客户名单交集

计算举例

组内相对位置

计算每个客户的运费月增长率

	A	B
1	=orders.groups(client,year(orderdate):fieldyear,month(orderdate):fieldmonth;count(1):quantity)	/按客户、年、月分组
2	=A1.new(client,fieldyear,fieldmonth,if(client==client[-1],(quantity-quantity[-1])/quantity[-1]))	/计算运费月增长率

关联计算

统计每年每月的订单数量和订单金额。

	A	B
1	=join(orderdetails:sub,orderid; orders:main,orderid)	/主子表归并关联
2	=A1.groups(year(main.orderdate):fieldyear, month(main.orderdate):fieldmonth; sum(sub.price*sub.amunt):subtotal,count(1):quantity)	/分组汇总

说明：上述关联算法中，由于订单表和订单明细都按orderid有序，因此可以用分段并行提高性能，即将join改为join@m

索引查找

对内表的主键建立索引，可以提高主键的查找性能，频繁查找时尤为明显

数据加载时，对orders表的主键建立索引

	A	B
1	=connect@l("orcl")	
2	=A1.cursor("select orderid,client,seller,orderdate,freight from orders")	/游标查询
3	=A2.memory()	/根据将游标转为内表
4	>A3.keys@i(orderid)	/建立主键，同时创建索引
5	>env(orders,A3)	/共享内表

实现业务计算：按参数订单编号pOrder查询内表orders的记录

	A
1	=orders.find(pOrder)

说明：有索引的内表，不必事先排序

并行计算

订单orders已常驻内存（参考数据加载一节），转成内存游标可进行并行计算

	A	B
1	=orders.cursor@m(;8)	/将内表转为8路游标，即8并行
2	=A1.groups(year(orderdate),month(orderdate),client;sum(feight))	/分组汇总

说明：不指明并行数时，将使用配置文件中预设的并行数

目录 CONTENTS

03

内存压缩

内表压缩

数据量较大时，内存往往放不下，这种情况下可使用压缩内表，以便内存容纳更多的数据

比如，人口表数据量巨大，无法直接放入内存，因此采用压缩内表读入内存

	A	B
1	=connect@l("orcl")	/连接oracle
2	=A1.cursor("select id,fname,lname, gender, birthday,phonenumber,address from population order by id")	/游标查询数据库
3	=A2.memory@z(id)	/创建压缩内表
4	>env(population,A3)	/共享内表

压缩内表在使用时会自动按区块解压缩，其性能比普通内表稍低，但远高于外存计算。

对程序员而言，压缩内表的用法与普通内表完全一样。

实现业务算法：按出生日期查询人口表（SQL语句）

```
select * from population where birthday>=date(‘2007-01-01’) and birthday<date(‘2008-01-01’)
```

数据类型压缩

很多字段的数据类型为字符串，这类字段会占用非常大的内存空间，且计算速度较慢。如果用整数或布尔来代替字符串，将显著节省内存，并显著提高计算性能。

employee表部分数据如下

eid	name	surname	gender	state	birthday	hiredate	dept	salary
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D	7000
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance	11000
3	Rachel	Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales	9000
4	Emily	Smith	F	Texas	1985/3/7	2006/8/15	HR	7000
5	Ashley	Smith	F	Texas	1975/5/13	2004/7/30	R&D	16000

将emplyee常驻内存，并将state、dept转为整数类型，将gender转为布尔类型

	A	B	C
1	=connect@l("demo").query@x("select * from employee order by eid")		/取员工表
2	=A1.(state).id()	=env(indexState,A2)	/获得state列表，并共享
3	=A1.(dept).id()	=env(indexDept,A3)	/获得dept列表，并共享
4	=A1.run(state=indexState.pos@b(state), dept=indexDept.pos@b(dept), gender=if(gender=="M",true,false))		/将员工表的state、dept转为整数，gender转为布尔
5	=A1.cursor().memory(eid)	=env(employee,A5)	/将员工表转为内表，并共享

数据类型压缩

使用类型转换后的数据：对内表emplyee进行条件查询，找出符合条件的记录。参数有pState（比如[“California”，“Colorado”，“Arkansas”]）、pDept（比如“R&D）、pGender（比如”F“）

	A	B
1	=indexState.pos@b(pState)	/pState：从字符串列表转为整数列表
2	=indexDept.pos@b(pDept)	/pDept:从字符串转为整数
3	=if(pGender=="M",true,false)	/PGender：从字符串转为布尔
4	=employee.select(A1.contain(state) && dept==A2 && gender==A3)	/按常规方式查询

说明1：可在参数表单中利用indexState、indexDept生成整数类型的参数pState、pDept，从而避免计算时的类型转换（A1-A3）

说明2：只含日期的time类型、位数较少的bigDecimal也可转为整数，需根据情况灵活处理

目录 CONTENTS

04

预关联

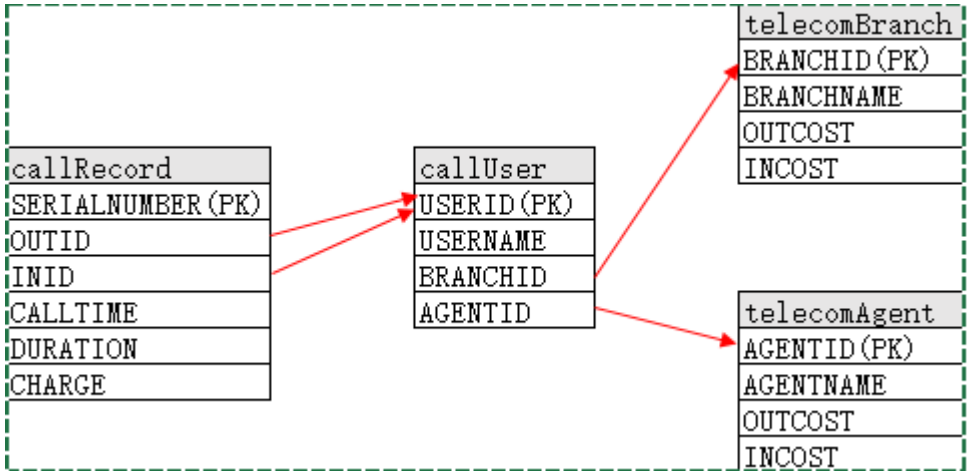
维表预关联

维表关联是个比较耗时的动作，如果加载数据时进行预先关联，计算时就会显著提升性能

传统数据库：用记录式复制做关联，占用大量额外内存

集算器内存数据库：用指针式复用做关联，几乎不占额外内存

比如：通话记录表、电信用户表、电信机构表、电信代理商表关系如下，要求计算通讯总成本，即所有呼出用户和呼入用户分别对应的网点均摊成本、代理商均摊成本之和。



说明：关于预关联对计算性能的影响，这里有一份性能对比说明，参考《http://c.raqsoft.com.cn/article/1574142747764》

维表预关联

在加载脚本中进行预关联

	A	B	C
1	=connect("orcl")		
2	=A1.query(“select * from telecomAgent”).keys(AGENTID)		/取维表telecomAgent
3	=A1.query("select * from telecomBranch").keys(BRANCHID)		/取维表telecomBranch
4	=A1.query("select * from callUser").keys(USERID)		/取维表callUser
5	=A4.switch(AGENTID,A2:AGENTID; BRANCHID,A3:BRANCHID)		/预关联callUser和telecomAgent、telecomBranch
6	=A1.cursor@x("select * from callRecord order by SERIALNUMBER")	=A6.memory(SERIALNUMBER)	/取callRecord，建立内表
7	=B6.switch(OUTID,A5:USERID; INID,A5:USERID)		/预关联callRecord和callUser
8	=env(callRecord,A7)		/共享全局变量callRecord

进行业务计算时，直接使用”.”操作符引用关联字段（该操作符只支持SPL脚本或SPL语句）

```
=callRecord.sum(OUTID.BRANCHID.OUTCOST+INID.BRANCHID.INCOST+OUTID.AGENTID.OUTCOST+INID.AGENTID.INCOST)
```

维表预关联

多主键 绝大多数维表只有单主键，用switch函数可实现预关联，如果偶尔遇到多主键（联合主键）维表，则应当使用join函数

事实表orders和维表product的关联字段是brand、type，部分数据如下（orders中标红色的记录在product中缺失）

orders

orderid	brand(fk)	type(fk)	quantity	orderdate
1	xiaomi	3310	2	2018/01/04
2	xiaomi	9210	1	2018/01/04
3	samsung	9210	4	2018/01/04
4	samsung	9210	5	2018/01/05
5	iphone	6plus	1	2018/01/05

product

brandid(pk)	typeid(pk)	price	origin
xiaomi	3310	199	china
xiaomi	9210	699	china
samsung	9210	599	korea
samsung	9770	899	korea
iphone	xr	1499	usa

在内存加载阶段，对两表进行预关联，同样用指针复用的方式

	A	B
1	=connect@l("orcl")	
2	=A1.query("select * from product order by brandid,typeid").keys(brandid,typeid)	/取维表
3	=A1.cursor@x("select * from orders order by orderid")	/游标取事实表
4	=A3.join@1(brand:type,A2:brandid:typeid,~:product)	/联合主键关联
5	=env(orders,A4.memory(orderid))	/共享内表

维表预关联

多主键

由于product表缺失记录，因此使用join@1进行左关联。关联后orders表新增字段product，该字段类型为指针，直向product表的记录。此时，orders的结构如下图

orderid	brand	type	quantity	orderdate	product
1	xiaomi	3310	2	2018-01-04	[xiaomi,3310]
2	xiaomi	9210	1	2018-01-04	[xiaomi,9210]
3	samsung	9210	4	2018-01-04	[samsung,9...
4	samsung	9210	5	2018-01-05	[samsung,9...
5	iphone	6plus	1	2018-01-05	(null)

brandid	typeid	price	origin
xiaomi	3310	199	china

brandid	typeid	price	origin
xiaomi	9210	699	china

brandid	typeid	price	origin
samsung	9210	599	korea

brandid	typeid	price	origin
samsung	9210	599	korea

业务计算时写法与之前类似，比如：计算各产地销售额

```
=orders.group(product.origin;sum(product.price*quantity))
```

说明1：switch函数也支持@1左关联

主子表预关联

主子表同样可以进行预关联

在加载脚本中，对订单和订单明细进行预关联

	A	B	C
1	=connect@l("orcl")		/连接oracle
2	=A1.cursor("select orderid,client,orderdate,freight from order orders by orderid")	=A1.cursor@x("select orderid,product,price,amount from orderdetail orders by orderid,product")	/游标查询数据库
3	=A2.memory(orderid)	=B2.memory(orderid,product)	/根据主键将游标转为内表
4	>env(orders,A3)	>env(orderdetails,B3)	/共享内表，允许其他程序用变量名访问
5	=join@m(B3:sub,orderid; A3:main,orderid)	>env(relation,A5)	/共享关联关系表

业务计算时进行分组汇总

```
=relation.groups(year(main.orderdate):fieldyear, month(main.orderdate):fieldmonth;
sum(sub.price*sub.amunt):subtotal,count(1):quantity)
```

说明：上述算法也可以使用switch函数预关联，但大主子表用join@m速度快得多，且支持多字段关联

目录 CONTENTS

05

内外存混合

内外存混合

有时数据太大无法放入内存，但又要与内存表共同计算，这种情况下可以利用集算器实现内外存混合计算。

例如：订单明细存储于外存，数据量较大，现在要将订单明细与内存里的订单关联起来，统计出每年每种产品的销售数量。首先在让订单表常驻内存：

	A
1	=connect@l("orcl")
2	=A1.cursor("select orderid,client,orderdate,freight from order orders by orderid")
3	=A2.memory(orderid)
4	>env(orders,A3)

业务计算时，先从组表读取订单明细，再和内存数据做混合计算

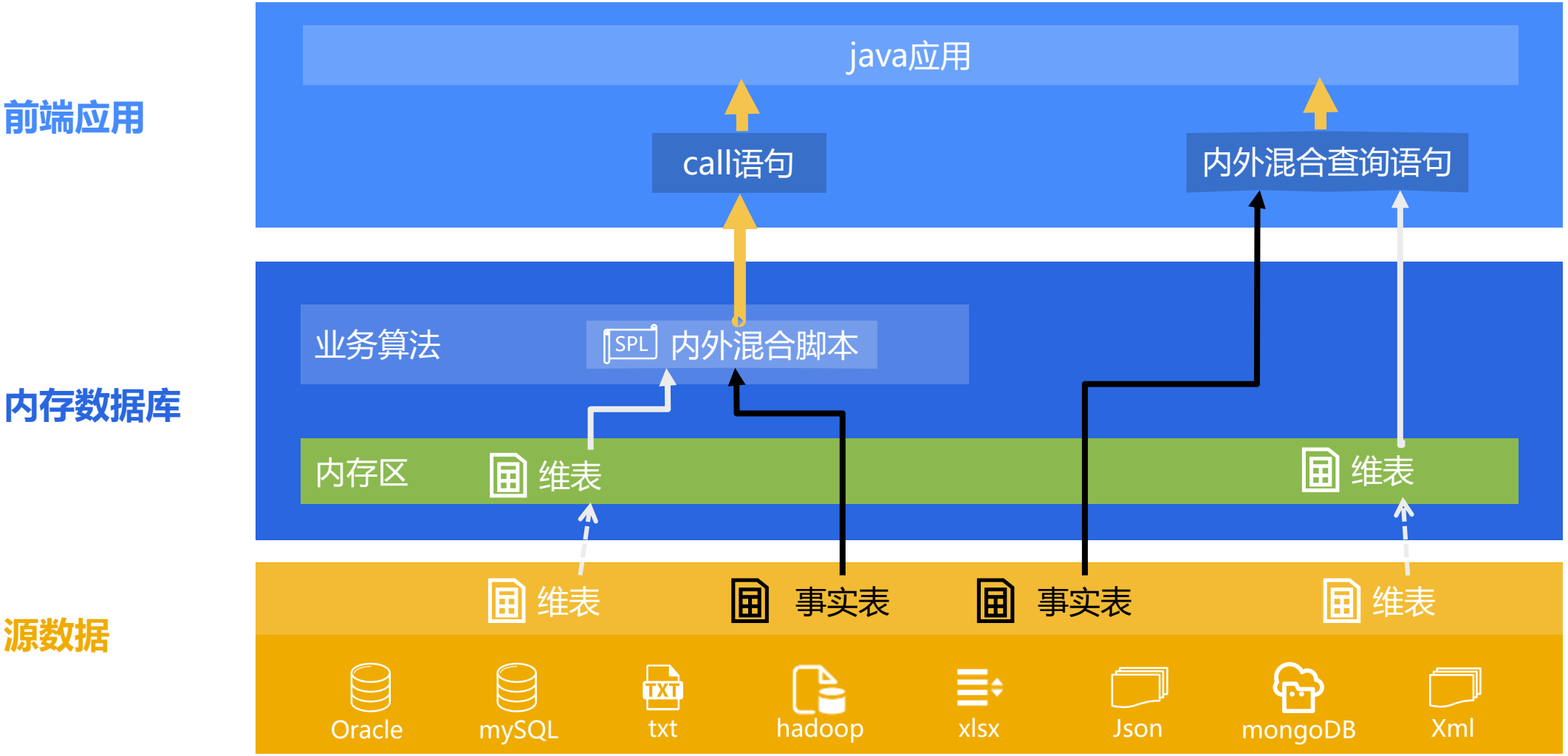
	A	
1	=file(“orderdetail.ctx”).create().cursor(orderid,product,amount)	/游标取订单明细
2	=orders.cursor()	/内表转为游标
3	=joinx(A2:sub,orderid; A3:main,orderid)	/并行归并关联
4	=A4.groups(year(main.orderdate):fieldyear,sub.product:product ;sum(sub.amount):subtotal)	/分组汇总

说明1：订单表较小时，A2A3可改写为=joinx@u(A2:sub,orderid; orders:main,orderid)

说明2： 外存数据也可存储于数据库、文本、mongodb等，但计算性能远不如组表。组表的用法参考 http://doc.raqsoft.com.cn/esproc/tutorial/zubiao.html 。《轻量级高性能文件型数据仓库》 http://c.raqsoft.com.cn/article/1570868099462

内存维表+外存事实表

典型场景：维表通常较小，适合放于内存，事实表通常较大，适合放于外存，可使用集算器完成内外存混合计算。



内存维表+外存事实表

电信用户表、电信机构表、电信代理商是维表，可用下面脚本常驻内存：

	A	B
1	=connect("orcl")	
2	=A1.query(“select * from telecomAgent”).keys(AGENTID)	/取维表telecomAgent
3	=A1.query("select * from telecomBranch").keys(BRANCHID)	/取维表telecomBranch
4	=A1.query@x("select * from callUser").keys(USERID)	/取维表callUser，数据量少时不必转为内表
5	=A4.switch(AGENTID,A2:AGENTID; BRANCHID,A3:BRANCHID)	/预关联callUser和telecomAgent、telecomBranch
6	=env(callUser,A5)	/共享维表

通话记录表是大事实表，以组表格式存储于外存。

计算通讯总成本时，可用如下脚本实现内外存混合计算

	A	B
1	=file(callRecord.ctx).create()	
2	=A1.cursor(OUTID,INID)	/游标形式取事实表
3	=A2.switch(OUTID, callUser:USERID; INID, callUser:USERID)	/外存事实表关联内存维表
4	=A3.sum(OUTID.BRANCHID.OUTCOST+INID.BRANCHID.INCOST+OUTID.AGENTID.OUTCOST+INID.AGENTID.INCOST)	/分组汇总

内存+任意外存

集算器支持多样性数据源，因此内存可以和任意外存混合计算

员工表和绩效表在逻辑上是同维关系，如下

employee

key	empID
	empName
	baseSalary
	post
	dept

performance

key	empID
	bonus

员工表已常驻内存，加载脚本如下：

	A	B
1	=connect（“orcl”）.cursor@x（“select * from employee order by empID”）	/取员工表
2	=env(employee,A1.memory(empID))	/常驻内存

内存+任意外存

假如绩效表存储于数据库，则计算实际工资时，脚本如下：

	A	B
1	=connect@l(“orcl”).query@x(“select * from performance order by empID”)	/数据库查询绩效
2	=A1.keys(empID)	/设置主键
3	=employee.derive().switch(empID,A2:empID)	/复制共享内表，并进行关联计算
4	=A2.new(empName, baseSalay+empID.bonus:realSalary ,post,dept)	/计算实际工资

假如绩效表存储于MongoDB，则计算实际工资时，脚本如下：

	A	B
1	=mongo_open("mongodb://192.168.1.7:27017/mydb")	/打开mongoDB
2	=mongo_shell@x(A1,” performance.find()").sort(empID)	/取数，并设置主键
3	=employee.derive().switch(empID,A2:empID)	/复制共享内表，并进行关联计算
4	=A2.new(empName, baseSalay+empID.bonus:realSalary ,post,dept)	/计算实际工资

说明1：数据库或mongodb等外存格式比组表慢得多，甚至不如文本文件。

说明2：复制内表的目的是保持原数据不变。

目录 CONTENTS

06

序号化与排号键

序号化

用switch函数实现外存事实表和内存维表的关联计算时，用字段值关联虽然已经很快了，但本质上仍然是hash关联。

如果维表的主键是序号，则应当用序号做关联，以便进一步提高性能

比如：orders表是外存事实表，employee是维表，已常驻内存，可以通过orders的seller字段与emplyee的eid字段建立关联。其中emplyee的eid字段是以1起始的序号，部分数据如下：

eid	name	surname	gender	state	birthday	hiredate	dept	salary
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D	7000
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance	11000
3	Rachel	Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales	9000
4	Emily	Smith	F	Texas	1985/3/7	2006/8/15	HR	7000
5	Ashley	Smith	F	Texas	1975/5/13	2004/7/30	R&D	16000

实现业务算法：用序号关联orders和employee，统计各部门的订单数量

	A	B
1	=connect@l（“orcl”）.cursor@x（“select * from orders”）	/游标取外存表
2	=A1.switch(seller, employee:#)	/内外存序号关联，#即序号
3	=A2.groups(sellerid.dept;count(1):quantity)	/分组汇总

序号化

如果关联字段不是序号，可以使之序号化，从而提高计算性能

比如：orders表是外存事实表，customer是小维表，可以通过orders的client字段与customer表的clientid字段建立关联。

其中关联字段是1位字母+4位数字的形式，customer的部分数据如下：

clientid	companyname	city	province	country
a0001	automatic line	shenyang	liaoning	CHINA
a0002	auto north co.ltd	dalian	shandong	CHINA
a0003	anistion product	Bibb	Alabama	USA
a0004	around out machine	Kauai	Hawaii	USA
b0001	berklin register	seo-gu	daego	KOREA

第一步：事实表关联字段序号化。将事实表的关联字段替换为序号。

	A	B
1	=connect@l("demo")	
2	=A1.query(“select *,0 as newid from customer order by clientid")	/取维表
3	=A2.run(#:newid)	/新增序号字段newid
4	=A1.cursor@x("select * from sales")	/游标取事实表
5	=A4.switch(client,A3:clientid)	/switch关联
6	=A5.new(orderid,client.newid:client,seller,orderdate,freight)	/用序号代替client字段
7	=file("orders.btx").export@z(A6)	/持久化存储

说明：序号化的重点是替换事实表的关联字段，替换后的结果应当存储起来，可以存储为任意格式，比如库表或文本都可以，但采用简表或组表的性能要好得多。

序号化

第二步：维表常驻内存。

	A	B
1	=connect@l("demo")	
2	=A1.cursor@x("select * from customer order by clientid")	/取维表
3	=env(customer,A2.memory())	/常驻内存

第三步：业务算法，用序号关联外存事实表和内存维表，完成分组汇总计算

	A	B
1	=file（“orders.btx”).cursor@b()	/游标取外存表
2	=A1.switch(client,customer:#)	/内外存关联
3	=A2.groups(client.country,client. province,client.city; count(1))	/分组汇总

排号键

序号化的性能最好，但需要事实表和维表先关联再转换，过程比较繁琐。

如果原键值接近序号，可以转换成排号键，避免复杂的转换过程，同样获得较高关联性能

比如：事实表orders的外键client为1位字母+4位数字的形式。

第一步，将client转换为排号键，并持久化

	A	B
1	=connect@l("demo")	
2	=A1.cursor@x("select * from sales")	/游标取事实表
3	=A2.run(client=k(asc(mid(clientid,1,1)),int(mid(client,2,2)),int(mid(client,4,2))))	/将clientid转为排号键
4	=file("orders.btx").export@z(A1)	/持久化存储

说明：排号键相当于多层的序号，计算时性能较高

排号键

第二步，维表customer常驻内存时，也需要将主键转为排号键

	A	B
1	=connect@l("demo")	
2	=A1.cursor@x("select * from customer order by clientid")	/取维表
3	=A2.run(client=k(asc(mid(clientid,1,1)),int(mid(clientid,2,2)),int(mid(clientid,4,2))))	/将client转为排号键
4	=env(customer,A2.memory().keys@s(client))	/建立主键索引，并常驻内存

第三步：业务算法，用排号键关联外存事实表和内存维表，完成分组汇总计算

	A	B
1	=file(“orders.btx”).cursor@b()	/游标取外存表
2	=A1.switch(client,customer:clientid)	/内外存关联
3	=A2.groups(client.country,client. province,client.city; count(1))	/分组汇总

说明：用keys@s()为排号键建立索引后，同样可用于高速查找。与keys@i()建立的哈希索引相比，排号键索引占用空间更小，性能更高

排号键

业务计算时，由前端应用完成（少量）排号键的转换，查询频繁时可降低服务器压力。

第一步，维表customer常驻内存，将主键转为排号键。内存服务器地址为192.168.1.10:8281

	A	B
1	=connect@l("demo")	
2	=A1.cursor@x("select * from customer order by clientid")	/取维表
3	=A2.run(client=k(asc(clientid,1),int(mid(clientid,2,2)),int(mid(clientid,4,2))))	/将client转为排号键
4	=env(customer,A2.memory().keys@s(client))	/常驻内存

第二步，前端应用查询query.dfx。pClient是前端输入的客户编号，在前端应用将pClient转为排号键，再将查询请求发给内存数据库，返回查询结果

	A	B	C
1	=k(asc(mid(pClient,1,1)),int(mid(pClient,2,2)),int(mid(pClient,4,2)))		/pClient转换为键值
2	fork A1;["192.168.1.10:8281"]	=customer.find(A2)	/在内存服务器上查找

如果查询过程较复杂或有重用的必要，则应当分离成内存数据库上的脚本文件，如下compute.dfx

	A	B
1	=customer.find(pCode)	/按排号键查询

此时，前端应用须用callx函数调用内存服务器的脚本文件，query.dfx如下

	A	B
1	=k(asc(mid(pClient,1,1)),int(mid(pClient,2,2)),int(mid(pClient,4,2)))	/将pClient转为排号键
2	=callx(“/raqsoft64/dfx/compute.dfx”,A1;[“192.168.1.10:8281"])	/调用服务器脚本

排号键构成举例

技巧：排号键特别适合转换结构复杂的字段

身份证号18位，比如10032219801213023X。具体格式如下

位数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
规则	行政区划						生日								流水号			校验
细则	省		地区		县		年				月		日		流水号			校验
范例	1	0	0	3	2	2	1	9	8	0	1	2	1	3	0	2	3	x

说明：
省、地区、县：各自取值范围为1-99。
生日：取值为“1900-01-01”至“2020-01-01” 。
流水：取值为1-999。
校验：根据前17位计算出的冗余校验位，取值为0至10，其中10用x来表示

将身份证转化为排号键，关键代码如下：

```
=k(int(mid(cardNo,1,2)),
int(mid(cardNo,3,2)),
int(mid(cardNo,5,2)),
int(mid(cardNo,7,4))-1900,
int(mid(cardNo,11,2)),
int(mid(cardNo,13,2)),
int(mid(cardNo,15,2)),
int(mid(cardNo,17,1))*11+if((c=mid(cardNo,18,1))=="X",10,int(c)))
```

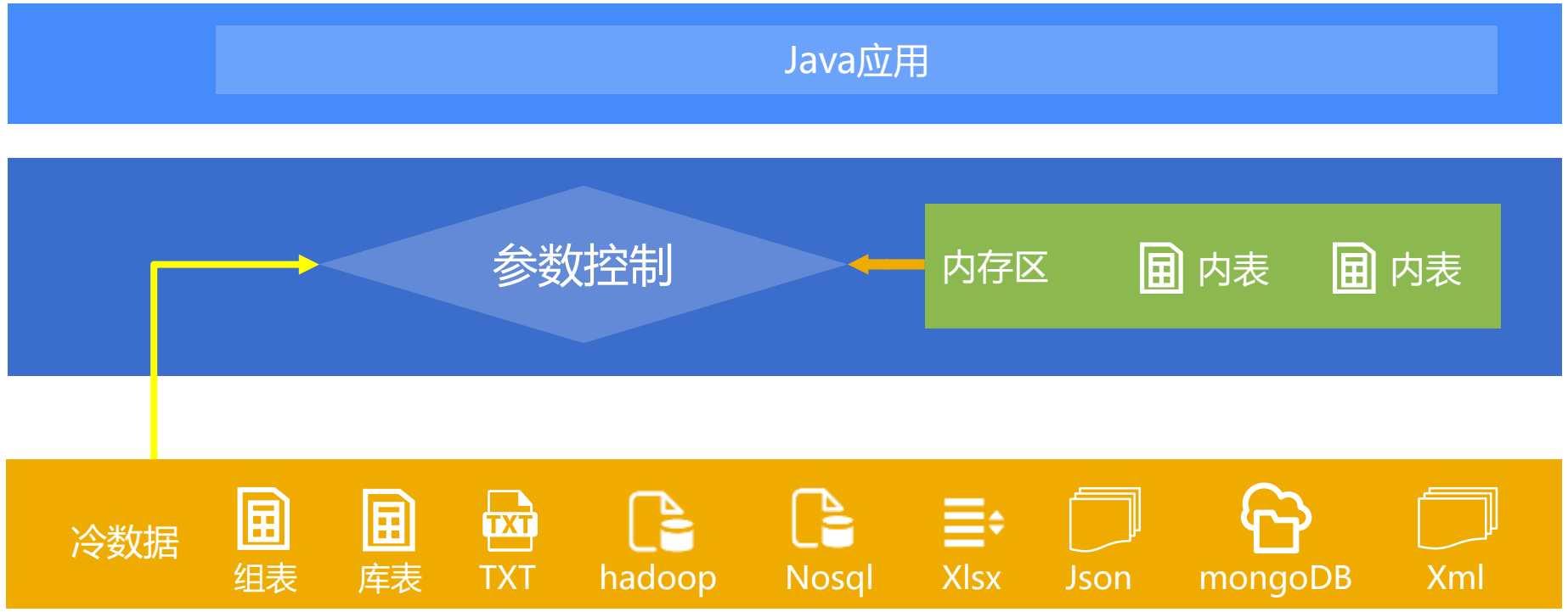
目录 CONTENTS

07

冷热路由

冷热路由

频繁访问的热数据可常驻内存，偶尔访问的冷数据可存储于远端，计算时通过判断参数区间，来决定返回冷数据还是热数据。



冷热路由

例子：以2019-01-01为分界线，分界线之前的冷数据从数仓取，分界线之后的热数据从内存取。
加载内存热数据initData.dfx

	A	B
1	=date("2019-01-01")	/设定冷热数据分界线
2	=env(hotcoldLine,A1)	/内存常驻冷热数据分界线
3	=connect@l(“orcl").cursor@x("select * from order where orderdate>=? order by orderdate",hotcoldLine)	/取热数据
4	=A3.memory(orderid)	/转为内表
5	=env(ordershot,A4)	/共享热数据，名为ordershot

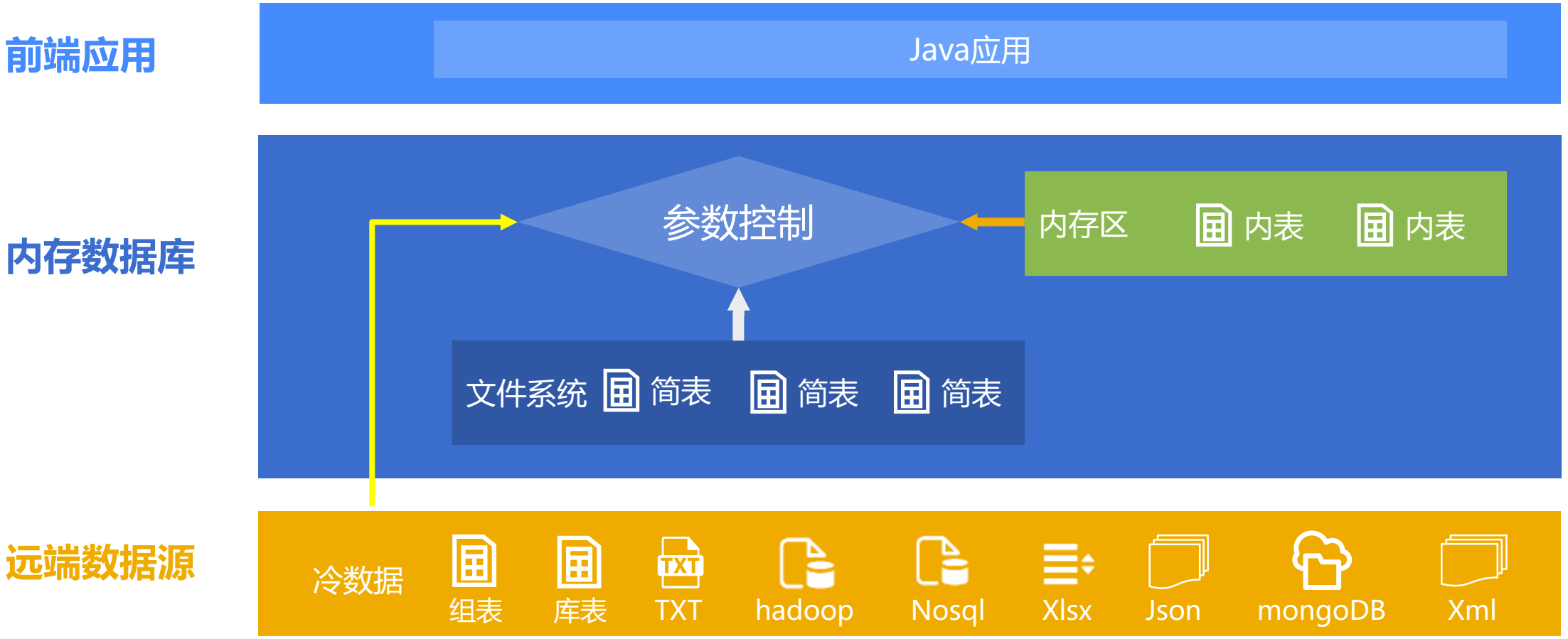
实现冷热路由router.dfx ：外部参数pBeginDate、pEndDate为取数范围，根据参数判断路由，分别从内存或数据库取数并合并，最后执行分组汇总算法

	A	B
1	=connect@l(“orcl").query@x("select * from order where orderdate>=? and orderdate<? order by orderdate",min(pBeginDate,hotcoldLine),min(pEndDate,hotcoldLine))	/取有序的冷数据，有可能为空
2	=ordershot.select@b(orderdate>=max(pBeginDate,hotcoldLine) && orderdate<max(pEndDate,hotcoldLine))	/取有序的热数据，有可能为空
3	=A1|A2	/合并冷热数据
4	=A12.groups@o(year(orderdate),month(orderdate);sum(amount):sAmount)	/有序分组汇总

说明：冷数据应按orderDate字段建数据库索引，以便为空时快速返回。

温度分层

为了充分利用硬件性能，数据可按温度分三层，高频热数据用全内存存放，中频温数据用本地文件存放，低频冷数据用数据库或数仓存放



温度分层

例子：以时间为分界线，当月（2019年11月）为内存热数据，当年不含当月（2019年）为简表温数据，往年（2019年以前）为数仓冷数据。

首先，用脚本warm.dfx生成温数据，可以选择在每月月初定时执行。

	A	B
1	=warmLine=pdate@y(now())	/当年温数据起点2019-01-01
2	=hotLine=pdate@m(now())	/当月热数据起点2019-11-01
3	=connect@l("demo").cursor@x("select * from sales where orderdate>=? and orderdate<? order by orderdate",warmLine,hotLine)	/游标取温数据
4	=file("orderwarm.btx").export@z(A3)	/生成简表

关于简表的详细用法，包括定时执行，可参考《敏捷数据计算中间件》应用数据缓存一节

http://c.raqsoft.com.cn/article/1573639294979

温度分层

内存数据库启动时，使用initMemory.dfx自动加载内存热数据

	A	B
1	=env(warmLine,pdate@y(now()))	/内存常驻温数据起点
2	=env(hotLine,pdate@m(now()))	/内存常驻热数据起点
3	=connect@l("demo").cursor@x("select * from sales where orderdate>=? order by orderdate",hotLine)	/取热数据
4	=A3.memory(orderid)	/转为内表
5	=env(orderhot,A4)	/共享热数据，名为ordershot

使用router.dfx进行业务计算。根据外部参数pBeginDate、pEndDate在温度分层数据中查找数据，合并结果，最后实现分组汇总算法。

	A	B
1	=connect@l("demo").cursor@x("select * from sales where orderdate>=? and orderdate<? order by orderdate",min(pBeginDate,warmLine),min(pEndDate,warmLine))	/从数仓取冷数据
2	=file("orderwarm.btx").cursor@b().select@b(orderdate>=max(pBeginDate,warmLine) && orderdate<min(pEndDate,hotLine))	/从简表取温数据
3	=ordershot.cursor().select@b(orderdate>=max(pBeginDate,hotLine) && orderdate<max(pEndDate,hotLine))	/从内存取热数据
4	=[A1,A2,A3].conjx()	/合并
5	=A4.groups@o(year(orderdate),month(orderdate);sum(amount):sAmount)	/高性能分组汇总

润乾 RAQSOFT

想要了解更多 请联系我们

技术内容请移步 乾学院

http://c.raqsoft.com.cn

优惠价购买请加入 好多乾

http://sys.misdiy.com/hdq.html

www.raqsoft.com.cn