Gdevops

全球敏捷运维峰会

新技术提升数据库开发效率

演讲人：韦伟

一道面试题

某支股票最长连续涨了多少交易日

```sql
select max(连续日数)
from (select count(*) 连续日数
        from (select sum(涨跌标志) over(order by 交易日) 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 涨跌标志
                      from 股价表) )
        group by 不涨日数)
```

SQL

更复杂的SQL

```
PROCEDURE SaleGoodsDayTpRt(in_store IN VARCHAR2,
                           in_billno IN VARCHAR2,
                           in_billid IN VARCHAR2,
                           in_rq IN VARCHAR2,
                           in_sj IN VARCHAR2)
  IS
  lstore      ba_area%ROWTYPE;
  ltime       dim_times%ROWTYPE;
  lsale       SALEGOODSDAYTPRT%ROWTYPE;
  lhead       tmp_sellhead%ROWTYPE;
  ldet        tmp_selldetail%ROWTYPE;
  lpay        tmp_sellpay%ROWTYPE;
  isj         number;
  newstore    char(4);

BEGIN
  DELETE FROM tmp_selldetail;
  DELETE FROM tmp_sellpay;
  DELETE FROM tmp_sellhead;
  DELETE FROM tmp_STRYHD;

  INSERT INTO tmp_sellhead
    SELECT * FROM Iv_Ord_Head_Zjl
     WHERE STR = in_store AND RQ  = in_rq  AND DH  = in_billno
       AND DJLX = in_billid AND SJ   = in_sj;
IF SQL%ROWCOUNT = 0 OR SQL%ROWCOUNT IS NULL THEN
    RETURN;
  END IF;
  INSERT INTO tmp_selldetail
    SELECT * FROM IV_ORD_DETAIL_zjl
     WHERE STR = in_store AND RQ  = in_rq  AND DH  = in_billno
       AND DJLX = in_billid AND SJ   = in_sj
       AND sn = 10;
  IF SQL%ROWCOUNT = 0 OR SQL%ROWCOUNT IS NULL THEN
    raise_Application_Error(ErrCode,'销售明细不存在!');
  END IF;
  INSERT INTO tmp_sellpay
    SELECT * FROM IV_ORD_PAY_zjl
     WHERE STR = in_store AND RQ  = in_rq  AND DH  = in_billno
       AND DJLX = in_billid AND SJ   = in_sj;
  IF SQL%ROWCOUNT = 0 OR SQL%ROWCOUNT IS NULL THEN
    raise_Application_Error(ErrCode,'销售付款不存在!');
  END IF;
  INSERT INTO tmp_stryhd
    SELECT * FROM YH_STRYHD_rpt
     WHERE STR = in_store AND RQ  = in_rq
     AND DH  = in_billno ;
```

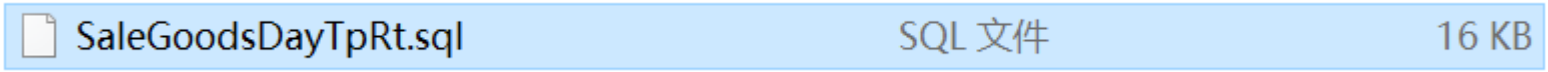
更复杂的SQL

```
SELECT * INTO lhead FROM tmp_sellhead
     WHERE STR = in_store AND RQ  = in_rq  AND DH  = in_billno
       AND DJLX = in_billid AND SJ   = in_sj;
    SELECT NVL(SUM(xsje),0) INTO ldet.xsje FROM tmp_selldetail
     WHERE STR = in_store AND RQ  = in_rq  AND DH  = in_billno
       AND DJLX = in_billid AND SJ   = in_sj;
    SELECT NVL(SUM(payje),0) INTO lpay.payje FROM tmp_sellpay
     WHERE STR = in_store AND RQ  = in_rq  AND DH  = in_billno
       AND DJLX = in_billid AND SJ   = in_sj;
    IF ABS(lhead.xsje - ldet.xsje)>0.01 OR
       ABS(lhead.xsje - lpay.payje)>0.01  THEN
       raise_Application_Error(ErrCode,'销售单不平衡!');
    END IF;

    if ldet.xsje < 0 then
       select nvl(max(str),in_store) into newstore from sa_set_strrq
        where befstr = in_store and dtype in ('KDSJ','BDSJ');
    else
        newstore := in_store;
    end if;
```

```
BEGIN
       SELECT * INTO lstore FROM ba_area
        WHERE str = newstore;
    EXCEPTION WHEN OTHERS THEN
      raise_application_error(errcode ,in_store||'门店未定义!');
    END;
    lsale.store      := lstore.str;
    lsale.storecls   := lstore.strtype ;
    lsale.saleorg    := lstore.orgid ;
    lsale.saleorgcls := '0';
    lsale.filiale    := lstore.fgsid ;
    lsale.filialecls := lstore.fgstype ;
    lsale.region1    := lstore.areaid ;
    lsale.region1cls := lstore.artype ;
    lsale.region2    := lstore.dqid  ;
    lsale.region2cls := '0';
    lsale.region3    := '0';
    lsale.region3cls := '0';

          ......
```



SQL的困难在哪里？

- 为查询设计的语法，并不适合计算
- 不能够分步骤计算
- 缺乏调试手段
- 性能不可控

有没有好的解决方案？

集算器

专门设计的代码语法体系

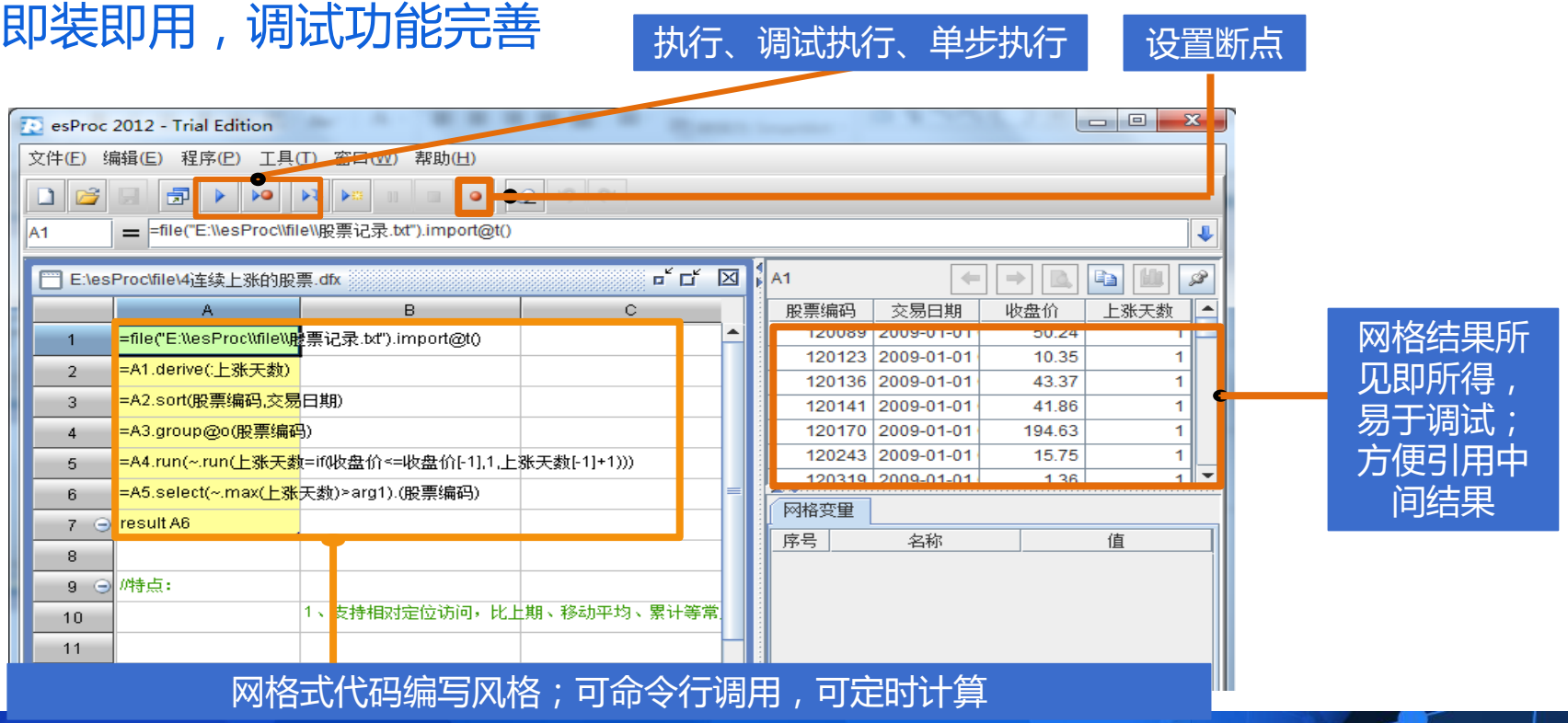
- 特别适合复杂过程运算

	A	B	C	D	E	F
1	=esProc.query("SELECT 订单ID AS 合同,订购日期 AS 日期,客户 ,订单金额 AS 金额,员工ID AS 销售 FROM 销售记录表 WHERE year(订购日期)=? OR year(订购日					
2	=esProc.query("select * from 员工表")					
3	>A1.run(销售=A2.select@1(编号:A1.销售))		/字段值 是记录			
4	=A1.group(销售)					
5	=create(销售,今年销售额,去年销售额,增长率,客户数,大客户数,大客户占比)					
6	for A4	=A6(1).销售.姓名				
7		=A6.select(year(日期)==年份).sum(金额)		/今年销售额		
8		=A6.select(year(日期)==年份-1).sum(金额)		/去年销售额		
9		=B8/B7-1	/增长率			
10		=A6.group(客户).(~.sum(金额))				
11		=B10.count()				
12		=A6.count(~.销售==A6(1).销售)				
13		=B12/B11				
14		>A5.insert(0,B6,B7,B8,B9,B11,B12,B13)				
15	result A5					

天然分步、层次清晰、直接引用单元格名**无需定义变量**

集算器的开发环境

- 即装即用，调试功能完善

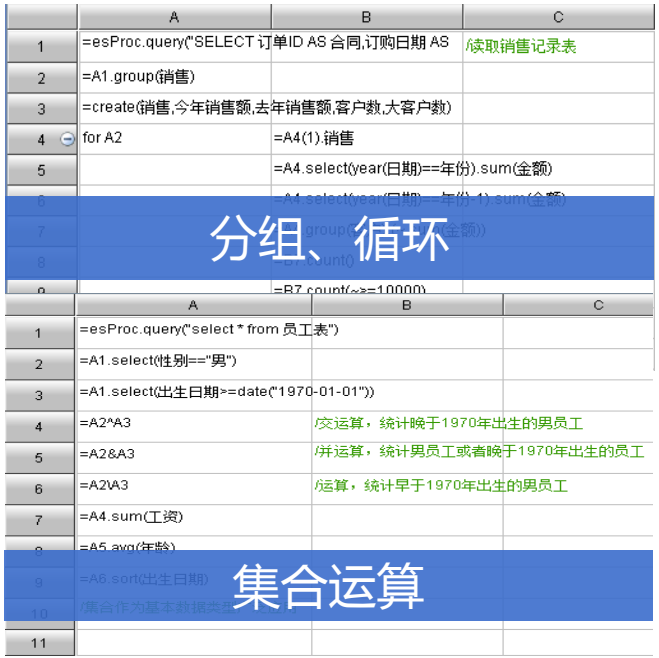


丰富的外部数据接口

- RDB：Oracle,DB2,MS SQL,MySQL,PG,....
- MongoDB，REDIS，...
- Hadoop：HDFS，HIVE，HBASE
- TXT/CSV，JSON/XML，EXCEL
- HTTP、ALI-OTS
- ...
- 内置接口，即装即用

丰富的运算类库

- 专门针对结构化数据表设计



集算器的解决办法

某支股票最长连续涨了多少交易日

```
1  select max(连续日数)
2  from (select count(*) 连续日数
3        from (select sum(涨跌标志) over(order by 交易日) 不涨日数
4              from (select 交易日,
5                    case when 收盘价>lag(收盘价) over(order by 交易日)
6                    then 0 else 1 end 涨跌标志
7              from 股价表) )
8        group by 不涨日数)
```

SQL

	A
1	=股价表.sort(交易日)
2	=0
3	=A1.max(A2=if(收盘价>收盘价[-1],A2+1,0))

集算脚本

思考：按照自然思维怎么做？

语法体系更容易描述人的思路

数据模型不限制高效算法实现

分组子集（一）

计算任务：找出总分500分以上的学生各科成绩记录

	A
1	=成绩表.group(学生).select(~.sum(成绩)>=500).conj()

有离散性后才能还原分组运算的本来面目

SQL没有游离记录构成的显式集合，无法保持分组结果，强迫聚合后要两次扫描并连接

```
1 WITH T AS
2 (SELECT 学生 FROM 成绩表 GROUP BY 学生 HAVING SUM(成绩)>500)
3 SELECT TT.* FROM T LEFT JOIN 成绩表 TT on T.学生=TT.学生
```

非常规聚合（一）

计算任务：列出用户首次登录的记录

	A
1	=登录表.group(uid).(~.minp(logtime))

聚合运算不一定总是SUM/COUNT这些，还可以理解为取出某个成员

有离散性时可以简单针对分组子集实施这种聚合

```
1  SELECT * FROM
2        (SELECT RANK() OVER(PARTITION BY uid ORDER BY logtime) rk, T.* FROM 登录表 T) TT
3  WHERE TT.rk=1;
```

非常规聚合（二）

计算任务：列出每个用户最近一次登录间隔

	A	
1	=登录表.groups(uid;top(2,-logtime))	最后2个登录记录
2	=A1.new(uid,#2(1).logtime-#2(2).logtime:间隔)	计算间隔

聚合函数返回值不一定是单值，也可以返回一个集合

彻底的集合化后很容易针对分组子集实施返回集合的聚合运算

```
1  WITH TT AS
2          (SELECT RANK() OVER(PARTITION BY uid ORDER BY logtime DESC) rk, T.*  FROM 登录表 T)
3  SELECT uid,(SELECT TT.logtime FROM TT where TT.uid=TTT.uid and TT.rk=1)
4          -(SELET TT.logtim FROM TT WHERE TT.uid=TTT.uid and TT.rk=2) 间隔
5  FROM 登录表 TTT GROUP BY uid
```

主子表

计算任务：由订单明细计算金额

	A	
1	=订单表.derive(订单明细.select(编号==订单表.编号):明细)	建立子表集合字段
2	=A1.new(编号,客户,明细.sum(单价*数量):金额)	计算订单金额

字段取值也可以是个集合，从而轻松描述主子表，适应于多层结构数据

SQL没有显式集合数据，没有离散性也不能引用记录，要JOIN后再GROUP

```
1  SELECT订单表.编号,订单表.客户,SUM(订单明细.价格)
2          FROM 订单表
3          LEFT JOIN 订单明细 ON 订单表.编码=订单明细.编号
4          GROUP BY 订单表.编号,订单表.客户
```

有序计算

关系代数延用了数学上的无序集合概念

早期SQL要生成序号后JOIN才能实现有限的有序计算

计算股票的涨幅

```
1 WITH T AS
2         (SELECT rownum 序号,交易日,收盘价
3         FROM ( SELECT * FROM 股票 ORDER BY 交易日) )
4 SELECT T1.交易日,T1.收盘价-T2.收盘价
5 FROM T T1 JOIN T T2 ON T1.序号=T2.序号+1
```

SQL2003标准增加了窗口函数方便生成序号并引用相邻成员

计算股票的涨幅（用窗口函数）

```
1 SELECT  交易日,收盘价-LAG(收盘价) OVER (ORDER BY 交易日) FROM 股票
```

跨行引用（一）

计算任务：选出比上月销售量和销售额均高出10%的产品销售记录

	A	
1	=销售表.sort(产品,月份)	
2	=A1.select(if(产品==产品[-1],销售量/销售量[-1])>1.1 && 销售额/销售额[-1])>1.1))	

集合有序后可直接提供跨行引用的方案

SQL窗口函数必须使用子查询才能实现跨行计算，多个跨行项要分别使用窗口函数

```
1  WITH T AS
2          (SELECT 销售量/LAG(销售量) OVER(PARTITION BY 产品 ORDER BY 月份) r1
3          (SELECT 销售额/LAG(销售额) OVER(PARTITION BY 产品 ORDER BY 月份) r2,
4          A.*, FROM 销售表 A)
5  SELECT * FROM T WHERE r1>1.1 AND r2>1.1
```

跨行引用（二）

计算任务：计算每月前后各一个月的销售额移动平均值

	A
1	=销售表.sort(月份).derive(销售额{-1,1}.avg()):移动平均)

有序集合上容易提供跨行集合引用方案

SQL窗口函数只有简单的跨行引用，涉及集合要用成员去拼

```
1  WITH B AS
2          (SELECT LAG(销售额) OVER (ORDER BY 月份) f1, LEAD(销售额) OVER (ORDER BY 月份) f2,   A.* FROM 销售表 A)
3  SELECT 月份,销售额,
4          (NVL(f1,0)+NVL(f2,0)+销售额)/(DECODE(f1,NULLl,0,1)+DECODE(f2,NULL,0,1)+1) 移动平均
5  FROM B
```

有序分组（一）

计算任务：一批婴儿中连续出生的同性别人数超过5人的有多少组

	A
1	=婴儿表.sort(生日).group@o(性别).count(~.len()>=5)

分组可能和次序相关，不只是等值分组

有序集合上可以定义与次序有关的分组，考察值变化时即产生新组

```
SELECT COUNT(*) FROM
        (SELECT 改变次数 FROM
                (SELECT SUM(改变标志) OVER ( ORDER BY 生日) 改变次数 FROM
                (SELECT CASE WHEN 性别=LAG(性别) OVER (ORDER BY 生日) THEN 0 ELSE 1 END 改变标志  FROM 婴儿表) )
        GROUP BY 改变次数 HAVING COUNT(*)>=5)
```

位置利用（一）

计算任务：计算一组商品价格的中位数

	A
1	=价格表.sort(价格).([(价格表.len()+1)\2,价格表.len()\2+1]).avg()

有序集合可以直接用序号访问成员

SQL的无序集合需要人为制造出序号，不分步运算加剧了困难

```
1  WITH N AS (SELECT COUNT(1) FROM 价格表)
2  SELECT AVERGE(价格) FROM
3          (SELECT 价格,ROW_NUMBER() OVER (ORDER BY 价格) r FROM 价格表 ) T
4          WHERE r=TRUNC((N+1)/2) OR r=TRUNC(N/2)+1)
```

位置利用（二）

计算任务：某股票股价最高的那三天的平均涨幅

	A
1	=股票.sort(交易日)
2	=A1.calc(A1.ptop(3,-收盘价),收盘价-收盘价[-1]).avg()

有序集合可以提供丰富的按位置访问机制

无序集合不能利用位置访问相邻成员，计算量增大，描述复杂度提高

```
1  SELECT AVG(涨幅) FROM
2          ( SELECT 交易日, 收盘价-LAG(收盘价) OVER ( ORDER BY 交易日) 涨幅 FROM 股价表
3                  WHERE 交易日 IN
4                          (SELECT 交易日 FROM
5                                  (SELECT 交易日, ROW_NUMBER() OVER(ORDER BY 收盘价 DESC) 排名 FROM 股票)
6          WHERE 排名<=3 )
```

结构化数据计算总结

离散性与集合化的有效结合

集合化是批量计算的基本能力
离散计算也不可或缺
离散性支持更彻底的集合化
离散性产生有序集合运算

离散集合模型 = 集合运算 + 游离成员

=> 彻底集合化/有序集合

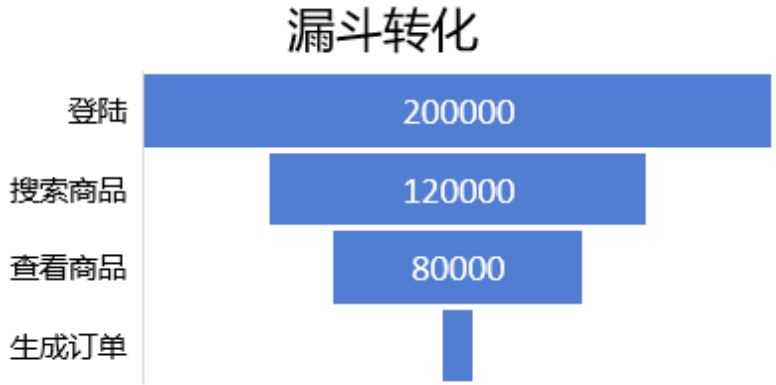
集算器对应用开发的提升

- 更少的代码，更短的开发周期
- 更好的性能保证
- 更稳定的应用
- 高度的可集成性
- 更好的代码可维护性

电商漏斗分析

漏斗分析是帮助运营人员分析一个多步骤过程中每一步的转化与流失情况

如：依次有序触发 “登陆”→“搜索商品”→“查看商品”→“生成订单 “ 事件的人群的转化流失情况



问题条件

日期过滤　如：仅针对2017-02-01至2017-02-28之间的事件

时间窗口　如：目标事件得在7天之内发生的

目标事件的有序性　如：先浏览商品后放入购物车，反之则不算

事件属性过滤　如：brand==“Apple”，其他品牌的不予统计

原始数据格式

用户ID：字符串类型

时间戳：毫秒级别，Long类型

事件ID：Int类型，包含10001到10010十个事件

事件名称：字符串类型，包含启动、登陆、搜索商品等十个事件

事件属性：Json串格式

日期：字符串类型

问题难点

时间窗口

如：2017-01-01 12:00 至2017-01-08 12:00之间也是7天

目标事件的顺序不确定

由输入参数决定目标事件的顺序，可能是”浏览商品->放入购物车”，也可能是”放入购物车->浏览商品”

事件属性不确定

浏览商品事件有brand属性，登录事件可能没有任何属性。

属性以json串的形式存在

没有事件码表和用户码表

聚合算法解说 针对单个用户

T =时间窗口 k=目标事件个数 C=long(date(2030,12,31))缺省的初始时间戳

初值有三个：M=最大事件序号，初值为0
A=k+1个最大事件序号，初值为0
S=k+1个初始时间戳，初值为C

假设k=3,T=3600000

M
3

A
3
1
0
0

S
1483600000000
1483602000000
1924876800000
1924876800000

事件ID	时间戳
1	1483200000000
事件ID	时间戳
2	1483201000000
事件ID	时间戳
1	1483600000000
事件ID	时间戳
2	1483601000000
事件ID	时间戳
1	1483602000000
事件ID	时间戳
3	1483602500000

实际代码 聚合算法

	A	B	C	D	E
7	=[0]*dNum	=to(dNum).([0]*I)	=to(dNum).([B1]*I)	=sdID-1	=now()
8	for A3,	for A8	if (p=B8.用户ID-D7,A7(p))==k	next	
9			>n=B8.事件ID, t=O+86400000*B8.日期+B8.时间戳, A=B7(p), S=C7(p)	for t-S(1)>T	>A.shift(),S.shift()
10			>i=A.pselect(n>~)	if n!=A(i)+1	next
11			>if((A(i)=n)==1,S(i)=t)	>if(i==1&&A7(p)<n, A7(p)=n)	>if(i>1&&A(i)==A(j=i-1)), (A.shift(j),S.shift(j)))

集算器其版本及功能

	关闭连接官网	64位	IDE	CMD	HTTP	JDBC	ODBC	常规数据源	强文本数据源	游标计算	外部数据源接口	并行计算	数据压缩存储	分段并行计算	高性能存储格式	索引	SERVER	节点负载均衡与容错	文件分区与冗余式容错	内存分区与备份式容错
个人版	★	★	★	★				★	★	★	★	64	★	★						
应用版	★	★	★	★	★	★	★	★	★	★	★	64	★	★	★	★				
服务器版	★	★						★	★	★	★	64	★	★	★	★	★	★	★	★
	直接运行无须连接官网	可以在64位环境下运行	IDE方式独立运行	命令行方式运行，可被调度	本地HTTP计算和访问	嵌入式JDBC计算	ODBC驱动	关系数据库、普通文本/规整Excel、动态数据源	多层json与XML计算	游标运算运行超出内存的大数据遍历	hadoop,nosql数据源Java数据接口，webservice访问	多线程计算	二进制压缩格式数据效率更高	外置数据支持分段并行处理	列式存储，组合表，序号层次键	数据索引	服务器方式独立运行	智能向节点分配任务，避开失效节点	源数据分区存储在节点上，自定义冗余规则	数据拆分加载到节点内存，出错时自动启动备份机补充

Gdevops

全球敏捷运维峰会

THANK YOU!