Gdevops

全球敏捷运维峰会

新技术提升数据库应用品质

演讲人：蒋步星

目录
contents

库外计算引擎

集算器计算模型

关联查询问题

库外计算引擎

数据库的封闭性

约束规则确保数据合法性

不能计算库外非标准数据

跨库计算困难

集算器：独立计算引擎

计算无处不在

开放性

可集成性，轻量级

多样性数据

多样性数据普遍存在

txt,csv,xlsx

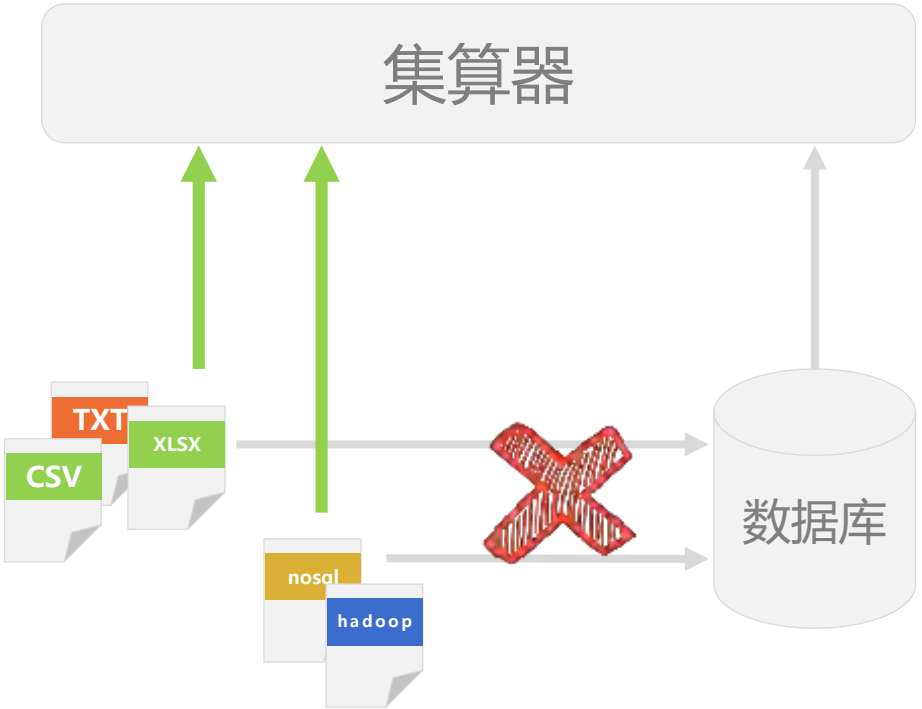
nosql,hadoop

json,xml

直接计算多样性数据

不需要建设专门的数据库及转入工作

结构简单、实时性更好



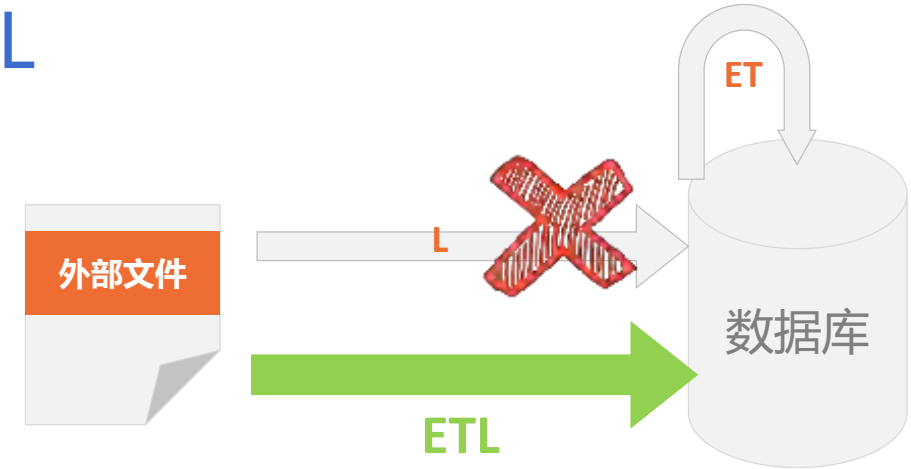
ETL过程

缺乏库外计算能力扰乱ETL过程

ETL ? ELT ? LET ?

加大数据库负担

库外计算实现合理的ETL



减少存储过程

存储过程的目的

数据整理

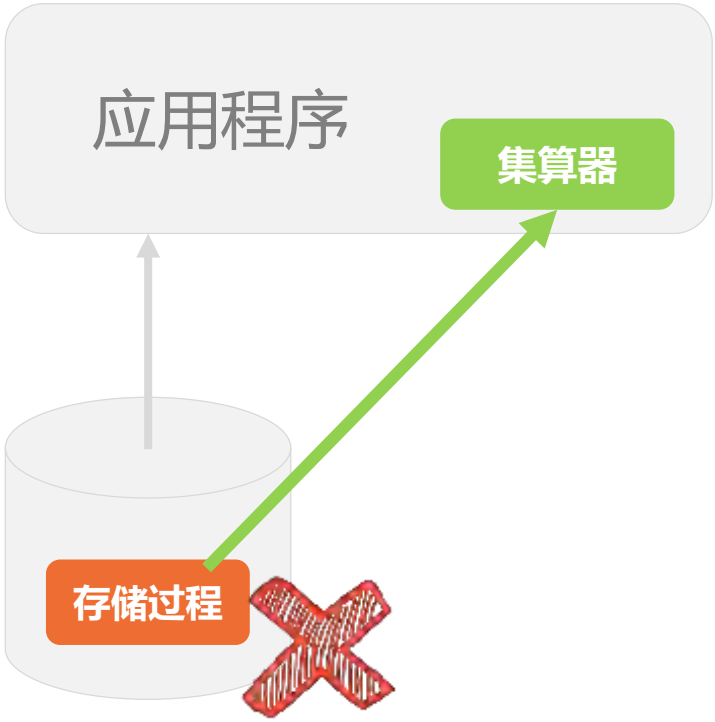
呈现准备

存储过程的问题

应用内与应用间耦合

安全性与易管理性

库外计算替代存储过程



减少冗余中间表

中间表的由来

运算复杂或数据量大

再次计算的能力

中间表的问题

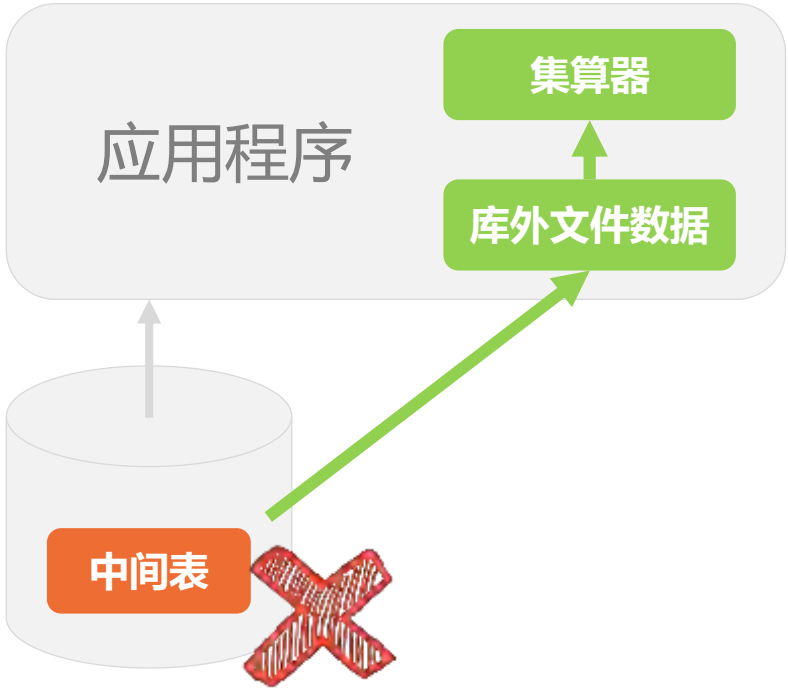
数量众多占用数据库资源

线性结构导致管理困难

库外计算将中间数据外置

计算不依赖于数据库，其它能力不需要

绑定应用、树状结构、易于管理



优化执行路径

复杂SQL的执行路径难以控制

库外计算优化SQL执行路径

自由控制执行步骤

部分运算移至库外进行

并行取数

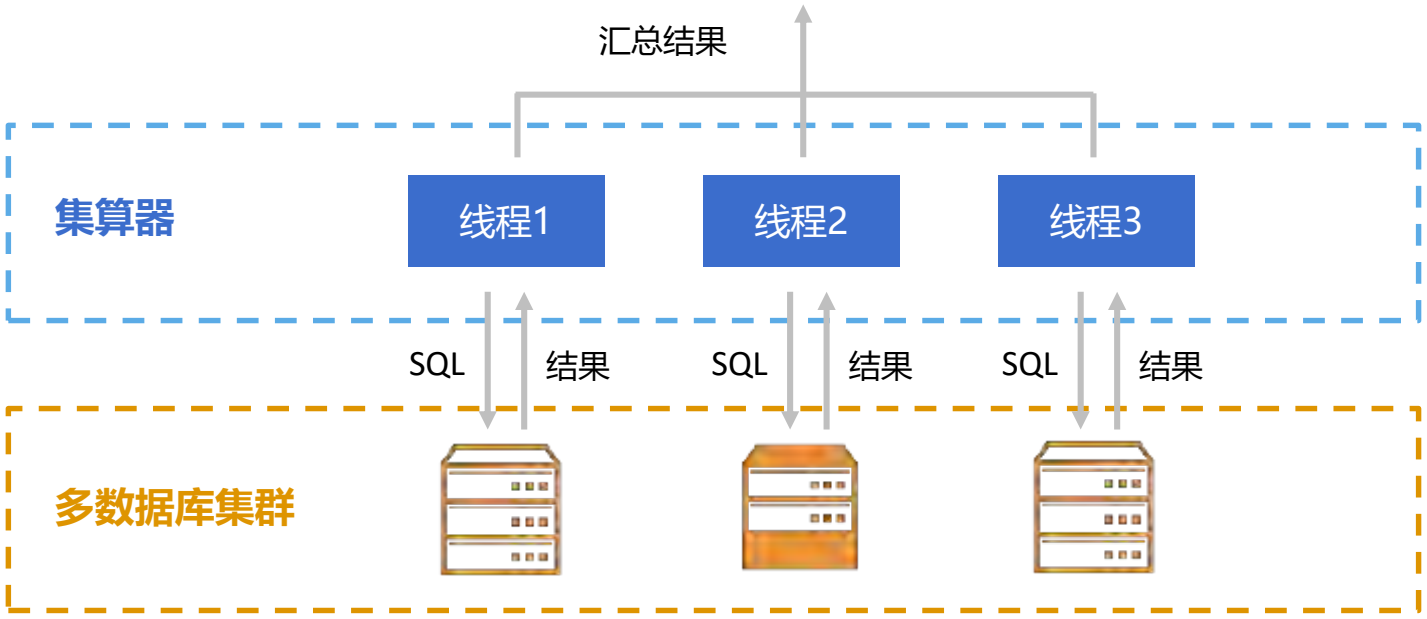
JDBC性能瓶颈

计算引擎多线程取数

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

跨库集群

异构数据库集群



T+0查询报表

T+0问题

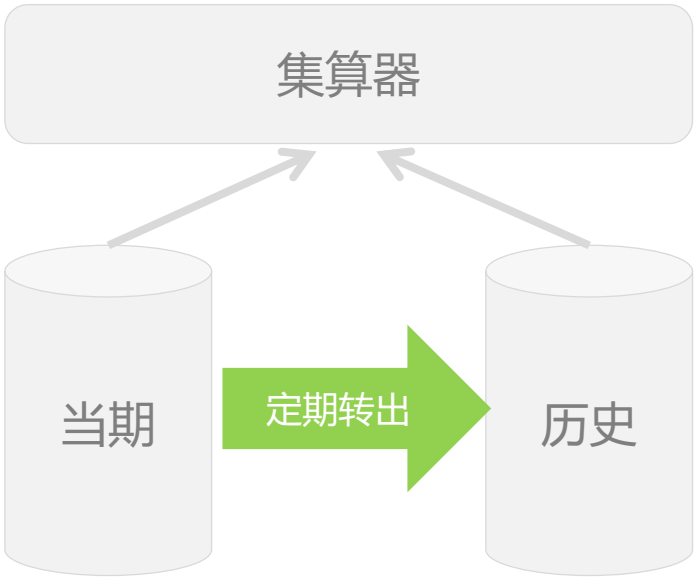
交易一致性要求关系数据库

历史与当期同库，数据量太大

历史与当期异库，跨库计算困难

库外计算实现并行跨库计算

历史数据还可文件化



数据中心

数据中心的特征与要求

数据库群

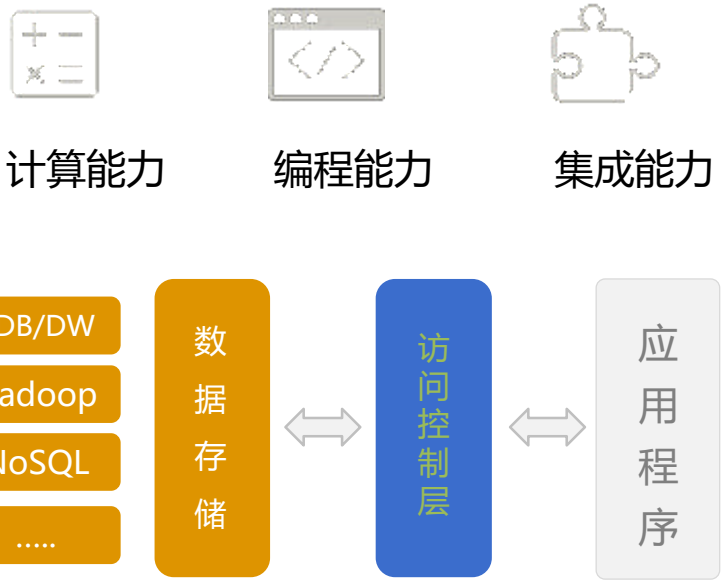
多样性数据源

服务式接口

访问受控

数据脱敏

集算器实现数据库中心访问层



目录
contents

库外计算引擎

集算器计算模型

关联查询问题

集算器计算模型

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

离散数据集 = 集合运算 + 游离成员

=> 彻底集合化/有序计算

=> 更高的开发效率和执行效率

分组子集

计算任务：用户在最后一次登录前三天内的登录次数

	A
1	=登录表. group(uid;~.max(logtime):last,~.count(interval(logtime,last)<=3):num)

针对分组子集的聚合运算较复杂时难以用简单聚合式写出，保留分组子集再结合分步计算则很容易

SQL不能保持子集，要用子查询在原集上附加信息，导致多次计算

```
1  WITH T AS
2          (SELECT uid,max(logtime) last FROM 登录表 GROUP BY uid)
3  SELECT T. uid,T.last,count(TT.logtime)
4          FROM T LEFT JOIN 登录表 TT ON T.uid=TT.uid
5          WHERE T.last-TT.logtime<=3 GROUP BY T.uid,T.last
```

非常规聚合

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;
```

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
```

有序分组

计算任务：一支股票最长连续上涨了多少天

	A	
1	=股票.sort(交易日).group@i(收盘价<收盘价[-1]).max(~.len())	

另一种和次序有关的分组，条件成立时产生新组

```
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 股票 ))
        GROUP BY 不涨日数)
```

分组有序计算

计算任务：找出连续上涨三天的股票

	A
1	=股票.sort(交易日).group(代码)
2	=A1.select((a=0,~.pselect(a=if(收盘价>收盘价[-1],a+1,0):3))>0).(代码)

分组子集与有序计算的组合

```
1  WITH A AS
2     (SELECT 代码,交易日, 收盘价-LAG(收盘价) OVER (PARITITION BY 代码 ORDER BY 涨幅) FROM 股票)
3  B AS
4     (SELECT 代码,
5          CASE WHEN 涨幅>0 AND
6               LAG(涨幅) OVER (PARTITION BY 代码 ORDER BY 交易日) >0  AND
7               LAG(涨幅,2) OVER PARTITION BY 代码 ORDER BY 交易日) >0
8               THEN 1 ELSE 0 END 三天连涨标志  FROM A)
9  SELECT distinct 代码 FROM B WHERE 三天连涨标志=1
```

聚合理解

从一个集合计算出一个单值或另一个集合都可理解为聚合
高复杂度的排序问题转换为低复杂度的遍历问题

	A	
1	=file(“data.txt”).cursor@t()	
2	=A1.groups(;top(10,amount))	金额在前10名的订单
3	=A1.groups(area;top(10,amount))	每个地区金额在前10名的订单

有序游标

复杂处理需要读出到程序内存中再处理

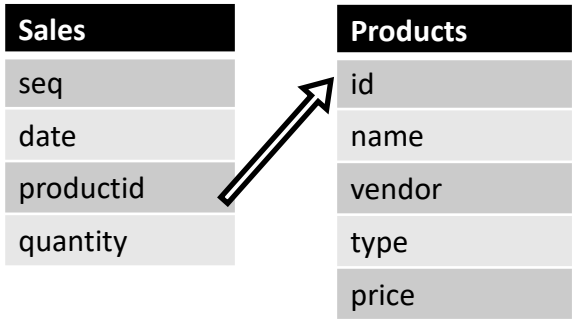
有序游标有效减少查找和遍历数量

	A	B	C
1	=file(“user.dat”).cursor@b()		/按用户id排序的源文件
2	for A1;id	…	/从游标中循环读入数据，每次读出一组id相同
3		…	/处理计算该组数据

外键指针化

外键需要随机小量频繁访问

内存指针查找大幅提高性能



	A	
1	=file(“Products.txt”).import()	读入商品列表
2	=file(“Sales.txt”).import()	读入销售记录
3	>A2.switch(productid,A1:id)	建立指针式连接，把商品编号转换成指针
4	=A2.sum(quantity*productid.price)	计算销售金额，用指针方式引用商品单价

	Java指针连接	Oracle
单表无连接	0.57s	0.623s
五表外键连接	2.3s	5.1s

外键序号化

序号化相当于外存指针化

	A	
1	=file(“Products.txt”).import()	读入商品列表
2	=file(“Sales.txt”).cursor()	根据已序号化的销售记录建立游标
3	=A2.switch(productid,A1:#)	用序号定位建立连接指针，准备遍历
4	=A3.groups(;sum(quantity*productid.price))	计算结果

不需要再计算Hash值和比较

有序归并

同维表和主子表连接可以先排序后变成有序归并

追加数据的再排序也仍然是低成本的归并计算

	A	
1	=file(“Order.txt”).cursor@t()	订单游标，按订单id排序
2	=file(“Detail.txt”).cursor@t()	订单明细游标，也按订单id排序
3	=joinx(A1:O,id;A2:D,id)	有序归并连接，仍返回游标
4	=A3.groups(O.area;sum(D.amount))	按地区分组汇总金额，地区字段在主表中，金额字段在明细子表中

目录
contents

库外计算引擎

集算器计算模型

关联查询问题

关联查询

SQL处理单表简单查询问题不大

有意义的查询经常是多表关联

SQL中表关联用JOIN运算实现

关系代数对JOIN运算的定义过于简单通用

笛卡尔积后再过滤

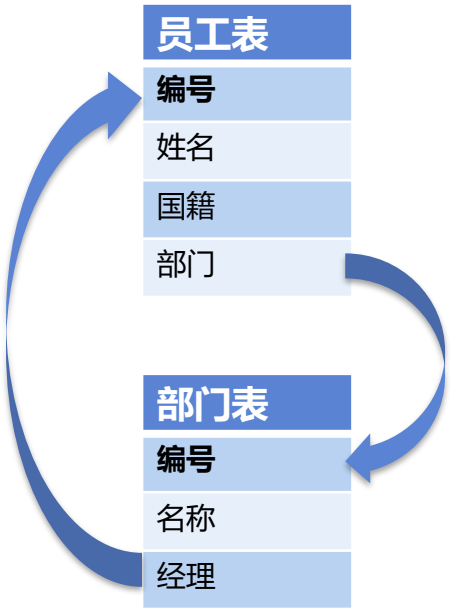
多表查询的理解和编写难度大

外键引用

查询目标：中国经理的美国员工

```
1  SELECT A.* FROM 员工表 A,部门表 B,员工表 C
2          WHERE A.国籍='美国'
3          AND C.国籍='中国'
4          AND A.部门=B.编号
5          AND B.经理=C.编号
```

```
1  SELECT * FROM 员工表
2          WHERE 国籍='美国' AND 部门.经理.国籍='中国'
```



同维表对齐

查询目标：所有员工的收入

```
1  SELECT 员工表.姓名,员工表.工资+经理表.津贴
2          FROM 员工表
3          LEFT JOIN 经理表 ON 员工表.编码=经理表.编号
```

```
1  SELECT 姓名,工资+津贴 FROM 员工表
```

主子表对齐

查询目标：订单及其客户与金额

```
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
```

```
1  SELECT 编号,客户,订单明细.SUM(价格) FROM 订单表
```

汇总后对齐

查询目标：按日期统计合同额、回款额和库存额

```
1  SELECT T1.日期,T1.金额,T2.金额 FROM
2  (SELECT 日期, SUM(金额) 金额 FROM 合同表 GROUP BY 日期 ) T1,
3  (SELECT 日期, SUM(金额) 金额 FROM 回款表 GROUP BY 日期 ) T2 ,
4  (SELECT 日期, SUM(金额) 金额 FROM 库存表 GROUP BY 日期 ) T3
5  WHERE T1.日期 = T2.日期 AND T2.日期=T3.日期
```

```
1  SELECT 合同表.SUM(金额),回款表.SUM(金额),库存表.金额 ON 日期
   FROM 合同表 BY 日期 JOIN 回款表 BY 日期 JOIN 库存表 BY 日期
```

合同表
编号
日期
客户
金额

回款表
序号
日期
来源
金额

库存表
序号
日期
数量
金额

混合情况

查询目标：按地区统计销售员人数和合同额

```
1  SELECT T1.地区,T1.数量,T2.金额 FROM
2  (SELECT 地区,COUNT(1) 数量 FROM 销售员 GROUP BY 地区 ) T1,
3  (SELECT 客户表.地区 地区,SUM(合同.金额) 金额 FROM 客户表,合同表
   WHERE 客户表.编号=合同表.客户 GROUP BY 客户表.地区 ) T2
4  WHERE T1.地区 = T2.地区
```

```
1  SELECT 销售员.count(1),合同表.sum(金额) ON 地区
     FROM 销售员 BY 地区 JOIN 合同表 BY 客户.地区
```

区分JOIN！

1

外键引用

外键属性化

2

同维主子表对齐

同维主子表等同化

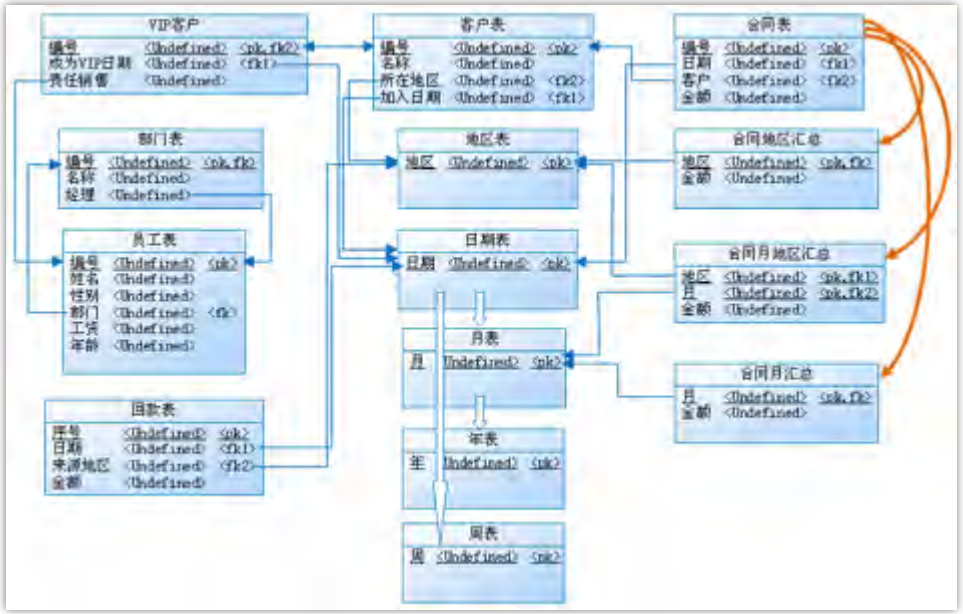
3

汇总对齐

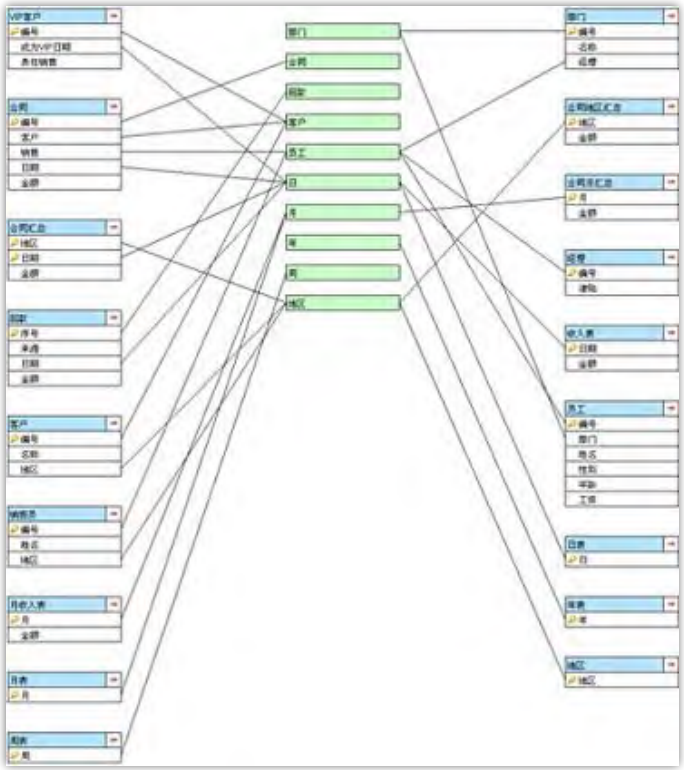
按维度自动对齐

低耦合的数据模型

网状结构的E-R图



线状结构的维关系图



类自然语言查询

上周 新增用户数，10月 北京 合同额 回款额，...

1. 关键词与字段匹配，拼SELECT和WHERE
2. 寻找表间关联路径，拼FROM和JOIN

SQL的JOIN机制下第2步非常困难

各种关联关系混在一起分不清

改变JOIN设计后关联清晰简单

外键、同维/主子表都变成了“单表”

汇总合并只要找到共同维度去对齐

本月消费超过20元的用户

```
SELECT T_1.PRODUCT_NO "用户",
T_1_1.HF0004 "消费" FROM
 DM_AUTORPT_BASE_INFO T_1 LEFT
 JOIN DM_AUTORPT_HIS_INDEX T_1_1
 ON T_1.PRODUCT_NO =
 T_1_1.PRODUCT_NO  AND T_1.YM =
T_1_1.YM WHERE
(MONTH(T_1.OP_TIME) =
MONTH(CURRENT TIMESTAMP)) AND
 (T_1_1.HF0004 > 20)
```

其它优势

降低出错率

避免无业务意义的完全叉乘

避免无关维度对齐

避免非维度随意分组

提高性能

维表内存化、序号化、多冗余

同维表主子表同序归并

汇总数据建设指导

Gdevops

全球敏捷运维峰会

THANK YOU！