

WalkSQL 项目

# WalkSQL 定义

文档编号: SQL01

版本 1.0

日期: 2003 年 9 月 17 日

# WalkSQL 技术手册

WalkSQL 的执行分解为两步，附表化和附表实现。附表化形成由标准 SQL 语句和附表表示函数 WalkSQLExTable()组成的准 SQL 语句，WalkSQLExTable()内部实现附表的创建和内容填充。附表化的关键问题是分析 WalkSQL 语句的结构，而附表实现则是数据计算。

## 0. 命名解释

### 0.1. 文档术语解释

SQL	DBMS 支持的 SQL 标准及其语句，并可以插入附表表达函数 WalkSQLExTable()。
WalkSQL	根据语义环境意义不同： 1. WALK 的一个功能模块。 2. WALK 系统中使用数据查询语句的标准。 3. WalkSQL 可以解释的，带有 WALK 对象特征的，需要转换后才能被 DBMS 系统支持的 SQL 语句。
附加表	在处理 WalkSQL 过程中生成的用于保存 WALK 对象特征的数据表。
附表	附加表的简称。
源表	附加表中 WALK 对象特征数据的来源表。
附表化	WalkSQL 中源表及其字段名转换成附加表名及其字段名的过程，但并不是数据执行过程。
附表实现	实现对附表的创建和内容填充。
WalkSQL 转换	把 WalkSQL 语句转换为 SQL 语句的过程。
特征	分为解析特征和合成特征，是 WALK 中地物、注记和式样内容及其特点。
特征函数	带参数特征一种表述，因为有些特征需要参数（如 OBJ9I）。
附表表达函数 WalkSQLExTable()	是在 SQL 中表达附表及其结构和来源的函数。
OBJ,OBJ1,OBJ2	地物基本特征，1:1 地物基本特征，1:n 地物基本特征。
OBJGM,OBJGM1,OBJGM2	地物几何特征，1:1 地物几何特征，1:n 地物几何特征。
OBJ9I	地物相交特征函数。
OBJGMS	地物统计特征。
OBJGEO	
E,N,H(X,Y,Z)	点坐标表示法，一般用 X、Y、Z 表示。
特征 n 数	一个地物/标注/式样中一种特征产生的特征值数目。
WALKSQL_AUTOID	附加表自动增量字段。

### 0.2. 程序表达规范

#### 0.2.1. 命名原则

以面向对象的思想进行命名。

#### 0.2.1.1. 函数命名原则

- 1) 函数名 := <对象><动词><备注>
- 2) 说明:
  - I. 每个部分首字母大写;
  - II. 对象表达函数操作对象;
  - III. 动词表达函数目的;
  - IV. 备注为其他需要说明的意义;

#### 0.2.1.2. 变量命名原则

变量名 := <对象>[<子对象>]<属性>[<子属性>]

#### 0.2.1.3. 常用对象

- 1) Obj, Obj1, Obj2, Objgm, Objgm1, Objgm2, Obj9i, ObjGms, ObjGeo  
特征对象名。
- 2) AssTable, SurTable  
附加表, 源表。
- 3) Sql, WalkSql  
普通 SQL 语句, 带特征的 SQL 语句。

#### 0.2.1.4. 常用动词

- 1) Set, Get  
对单个值的操作。
- 2) Initial  
初始化。
- 3) Select, Create, Update, Insert  
对数据表操作。
- 4) Translate  
转换, 翻译。
- 5) Deal  
处理, 解释。
- 6) Asstablize  
附表化。
- 7) Fill  
填充内容(如记录)。

## 1. 地物解析特征

从地物二进制 Geometry 字段中解析出具体含义的特征。

### 1.1. 基本特征

在 WalkSQL 中表达形式: <特征>:= [<层地物表名>].[OBJ][GM].<特征名>。

#### 1.1.1. 数值特征

##### 1.1.1.1. 1:1 数值特征

针对一个地物的属性值, 简称 OBJ1。

OBJ.AREA,	//面积
OBJ.PERIMETER,	//周长
OBJ.CX,	//中心点 X 坐标

OBJ.CY,	//中心点 Y 坐标	
OBJ.GEOTYPE,	//几何类型	int
OBJ.POINTCOUNT,	//点数	int
OBJ.PARTSCOUNT,	//Parts 个数	int
OBJ.POINTALLCOUNT,	//顶点数	int
OBJ.MINX,	//外接边界左下角 X 坐标	
OBJ.MINY,	//外接边界左下角 Y 坐标	
OBJ.MAXX,	//外接边界右上角 X 坐标	
OBJ.MAXY,	//外接边界右上角 Y 坐标	

**地物的其他特征** (楼宇, 2003-11-04)

OBJ.HAS_ARC,	//地物含圆弧数
OBJ.HAS_CURVE,	//地物含曲线数
OBJ.NEAT_CHECK,	//几何数据检查, 结果:

0—无问题  
 1—零长线 |  
 2—零面积环 |  
 4—线重点 |  
 8—有自交或单刺 |  
 16—空几何 (空线、空面等) |  
 32—环不闭合  
 64—有非法线型

#### 1.1.1.2. 1:n 数值特征

针对地物几何中的所有点, n 的值是组成地物几何的点数目, 简称 OBJ2。

OBJ.POINTX,	//点的 X 坐标
OBJ.POINTY,	//点的 Y 坐标
OBJ.POINTH,	//点的 Z 坐标, 也就是 H 高程坐标
OBJ.POINTN,	//点编号, 与 PARTSN, POINTSN 定位点
OBJ.DISTANCE,	//点所在边边长
OBJ.DISTANCEN,	//点所在边长集序号
OBJ.DISTANCET,	//点所在边长线段类型 int
OBJ.PARTSN,	//点所在边长 Parts 编号 int
OBJ.POINTSN	//点所在边长 Points 编号 int

针对一个地物的多个 CELLID, n 的值是

OBJ.CELLID	//地物隐式索引格号
------------	------------

#### 1.1.2. 几何特征

##### 1.1.2.1. 1:1 几何特征

针对一个地物的特征, 简称 OBJGM1。

OBJ.GM_CENTRO,	//中心点
OBJ.GM_BOX,	//最小外部矩形
OBJ.GM_BUFFER,	//缓冲区, 特征函数参数(缓冲距离, 顶点类型, 拐点类型), 暂时不实现

**几何数据整理后的 OBJ:** (楼宇, 2003-11-04)

OBJ.GM\_NEATEN(<opMask>, <整理精度>, <对齐位数>[, {区域边界}])

区域边界::={最小 E, 最小 N, 最小 H, 最大 E, 最大 N, 最大 H}

opMask::={ 1—零长线 |  
 2—零面积环 |  
 4—线重点 |  
 8—圆弧转折线 |  
 16—曲线转折线 |  
 32—圆弧曲线加密为折线}

**地物自交和单刺的可能发生点:** (楼宇, 2003-11-04)

**OBJ.GM\_SEFTINTER**(<判定精度>)

#### 1.1.2.2. 1:n 几何特征

针对一个地物的所有组成部分, 简称 OBJGM2。

OBJ.GM\_POINT, //所有 Point

OBJ.GM\_POINTS, //所有 Points

OBJ.GM\_PARTS, //所有 Parts

**弧段化结果:** (楼宇, 2003-11-04)

~~OBJ.GM\_ARC, //所有弧段化后的 Parts~~

**OBJ.GM\_SEGMENT,** //所有线段或 3 点弧

#### 1.2. 相交特征

在 WalkSQL 中表达形式:

<特征>:=OBJ9I.<特征名>(<层 1 地物表名>, <层 2 地物表名>)。

针对两个层地物表, 所以这些特征的使用需要参数, 如 OBJ9I.CONTAIN (LAFeatures, LBFeatures), 这些特征函数只能用于 WalkSQL 的 Where 子句作为条件使用, 其返回值为 TRUE 或 FALSE。

OBJ9I.EQUAL //相等

OBJ9I.CONTAIN, //包含

OBJ9I.INTERSECT, //相交

OBJ9I.OVERLAP //有重叠

**两两地物的 9I 关系:** (楼宇, 2003-10-31)

**OBJ9I.RESULT**(<层 1 地物表名>, <层 2 地物表名>, <Mask9i>)

Mask9i: 下列值的组合

(0 为两地物无关)

2 -- 相等, wk9iEqual

4 -- 包含, wk9iContain

8 -- 相交, wk9iIntersect

16 -- 相触, wk9iTouch

32 -- 内触, wk9iInnerTouch

64 -- 外触, wk9iOuterTouch

**地物与点 XY 的包含关系:** (楼宇, 2003-12-6)

**OBJ9I.CONTAINPTXY**(<层 1 地物表名>, <点表>(<点表 ID>, <点表 X>, <点表 Y>, <bTouch>))

bTouch -- 0 不包括 Touch(相触)条件, 1 含相触条件

该语句的执行结果得到满足 9I 条件的层 1 地物的 FeatureID 和点表的 ID。

**OBJ9I.CONTAINPTXY** 与 **OBJ9I** 的其他函数有较大的不同, 它有 6 个参数, 而附表化

时对应的输出只有前 2 个。程序实现中作为特例进行处理。

### 1.3. 统计特征

在 WalkSQL 中表达形式: <特征>:=[<层地物表名>].OBJGMS.<特征名>({<分类字段>})。

这些特征的结果是根据一定分组条件分组后一组地物的结果, 结果类型是新的地物几何, 需要参数, 参数是分组条件字段, 如 OBJGMS.UNION(区 ID, 街坊 ID), 就是把每个区的每个街坊的地物进行统计操作。出现这类特征的语句中不允许出现同级的 Group by 子句, 因为参数已经具有次含义, 同时 Select 子句不允许出现与 Group by 含义冲突的字段, 与 SQL 标准中一致。

OBJGMS.UNION,	//几何并
OBJGMS.INTERSECT,	//几何交, 若交为空集, 不生成新地物
OBJGMS.COMBINE,	//返回多个地物几何的集合, 也是地物几何的一种
OBJGMS.CENTRO,	//求多个 Geometry 合并后的 Centriod

## 2. 地物合成特征

根据点序列坐标数据表 S 生成地物几何, S 表的每一行表示一个点, S 表中必须有点的 X, Y, Z 坐标列。与解析特征不同, 这个特征的前缀不是层地物表表名, 而是 S。

### 2.1. 点地物合成

S.OBJGEO.POINT(<X>,<Y>,<H>, FEATUREID>)

特征函数名: OBJGEO.POINT。

功能: 根据点集数据表 S 的生成点地物。

<X>: S 表中代表点 X 坐标的字段名

<Y>: S 表中代表点 Y 坐标的字段名

<H>: S 表中代表点 Z 坐标的字段名

<FEATUREID>: S 表中标识点的代码字段名

例子: 从中国城市表生成数维市场层的地物

Update 数维市场 Set Geometry = 中国城市.ObjGeo.Point(L, B, 0, 区划代码)

Where 中国城市.城市 = 数维市场.城市

### 2.2. 通用地物合成

S.OBJGEO.LineString(<X>,<Y>,<H>,<LineType>,<GeoType>,  
<FeatureID>,<PartsNo>,<PointsNo>,<PointOrder>,  
<Filter>)

特征函数名: OBJGEO.LINESTRING

功能: 根据 S 表中的点记录流生成地物几何。

参数:	<X>:	in, S 表中点的 X 坐标字段名
	<Y>:	in, S 表中点的 Y 坐标字段名
	<H>:	in, S 表中点的 Z 坐标字段名
	<LineType>:	in, 采用 OBJ.DISTANCET 定义
	<GeoType>:	in, 0—点, 1—线, 2—面
	<FeatureID>:	in, S 表中地物标识字段名。

<PartsNo>: in, S 表中地物 Parts 的标识字段名。  
<PointsNo>: in, S 表中地物 Parts 的 Points 的标识字段名。  
<PointOrder>: in, S 表中地物 Parts 的 Points 的 Point 的排序标识。  
<Filter>: in, S 表的过滤条件字符串。

备注: G.featureid 用来与源表 S 连接, 数值完全相等。

### 3. WalkSQL 格式与处理

#### 3.1. 格式要求

- 1) 形参以@代表。  
例: select \* from LayerFeatures where obj.area > @minval
- 2) 如果在 WalkSQL 中只出现一个数据表, 特征前面可以不出现表名, 否则必须以表名为前缀。  
例: LayerFeatures.OBJ.Area。
- 3) 如果 Where 子句中出现 OBJ9I 并且有不同优先级的操作符, 必须用小括号, 使得括号外的所有条件为同一优先级, 即可以顺序处理。
- 4) 嵌套语句必须用小括号。

#### 3.2. 处理基本过程

- 1) 将形参替换为实参。
- 2) 自上而下逐条取以 GO 分隔 WalkSQL 进行附表化。
  - I. 由内向外 WalkSQL 附表化。
  - II. 重复步骤 I, 直到 SQL 中无 OBJ 内部函数。
- 3) 重复步骤 2, 直到对所有 WalkSQL 完成附表化。
- 4) 对所有 SQL 语句进行语法检查。

#### 3.3. 实现层次

- 1) 使用附表表达函数 WalkSQLExTable()实现 WalkSQL 向 SQL 的转换, 即附表化。
- 2) SQL 执行中通过 WalkSQLExTable()内部对不同类别特征的附表创建和内容填充实现附表。
- 3) 最后 SQL 基于 WalkSQLExTable()的结果完成执行过程。

### 4. WalkSQL 附表化

转换过程是把 WalkSQL 通过附表化形成 SQL 语句, 但并不执行 SQL 语句。包括三个部分:

- 用附表表达函数 WalkSQLExTable()表达需要的附加表及其结构;
- 用附加表字段替代特征, 消除 WALKSQL 中的特征, 生成 SQL;
- 生成附加表清除语句。

#### 4.1. 特征处理顺序

WalkSQL 转换过程中, 依次转换以下特征。

OBJGEO. 附加表中生成地物几何及其标识字段内容。

OBJ9I. 附加表中生成两层地物对应关系字段内容, 之前还需要与 OBJ.

	特征的附加表，在 OBJ.特征转换时进行。
OBJ.	附加表中生成对应于字段内容，用附加表字段替换。
OBJGMS.	附加表中生成地物几何及其标识字段内容。

## 4.2. 附表表达函数

WalkSQLExTable(), 在 WalkSQL 处理过程中需要产生附加表都通过这个函数实现，一个附表表达函数表达一个附表。

### 4.2.1. 函数定义

WalkSQLExTable (<Op>, <附表名>(<字段名>),  
                   <源表名>(<字段表达式 | Walk 特征>),  
                   <条件用表>, <附加条件>);

功能： 从源表中读取几何数据，直接或者间接解析生成附加表。

参数： <Op>:               in, 为对附加表的操作（可以是数字或文字），包括：  
                               0/Create—创建附表；  
                               1/Insert—向附表添加数据；  
                               2/Update—更新附表。  
       <附表名>:           in, 即将产生的附加表表名。  
       <字段名>:           in, 附加表中的字段名序列，这些字段保存源表转换后的内容，其类型根据特征判断。字段名与特征存在对应关系。但 WALKSQL\_AUTOID 没有对应。  
       <源表名>:           in, 附加表内容的来源表，一般是层的地物表，对于需要多表来源的特征（如 OBJ9I），需要多个源表名。  
       <Walk 特征>:       in, 依据源表的 Walk 特征函数名序列。  
       <条件用表>:       in, 用于辅助源表选取记录的数据表表名序列，可以为空。  
       <附件条件>:       in, 源表记录选择条件，不能使用特征函数名，可以为空。

例子： 存放合成特征的附加表

```
WalkSQLExTable(
    0,
    G (zdh, geometry),
    村宗地 features(zdh, OBJGEO.LineString(X, Y, 0, 1, 2, zdh, 0, 0, xh)),
    ,
)
```

zdh —— 宗地号

X —— 界址点 X 坐标

Y —— 界址点 Y 坐标

xh —— 界址点的宗内序号

备注：

- 1) WalkSQL 附表实现过程将根据特征函数的规定，确定附加表的字段名和字段类型。如本例中，G 表的 Geometry 由 OBJGEO.LineString 函数生成。
- 2) 如果附加表的字段名数目少于特征函数数目，WalkSQL 转换过程中将自动填入附加表的字段名，回填 WalkSQLExTable 函数，以帮助用户使用正确的附加表。
- 3) WalkSQLExTable 中的附加条件项用于向附加表填充数据时作为 Where 子句。附加条件可为空，可不必加 Where 字（WalkSQL 会过滤掉在句首出现的 Where）。



- 4) **如果有多个源表**，每个源表标识字段名和字段名数目相等，按源表顺序排列。
- 5) 每个表达函数只能表达一个源表的一类特征（并且 n 数目相等）。

#### 4.2.2. 内部执行过程

##### 4.2.2.1. Create

WalkSQLExTable (Create,.....)

- 1) 创建附表
- 2) 读取源表数据
- 3) 附表内容生成
  - I. 从源表中每读取一条，计算特征，然后执行插入语句：
  - II. Insert <附表名> (<id>, <字段名>, ...) Values(<id>, <特征 1 值>, <特征 2 值>, ...).

##### 4.2.2.2. Insert

WalkSQLExTable (Insert,.....)

向附表添加数据，要求附表已存在，执行过程同 Create，但不需要创建附表。

##### 4.2.2.3. Update

WalkSQLExTable (Update,.....)

更新附表数据，要求附表已存在，由 WalkSQL 内部对附表数据更新。

读取源表数据过程同 Create。

附表内容生成

- I. 从原表每读取一条，计算特征，然后执行更新语句：
- II. Update <附表名> Set 附表名.字段名 1 = 源表名.特征 1 值, ...  
Where <附表名.id>=<源表.id>

#### 4.3. 实例分析

首先使用 WalkSQLExTable 形成附加表生成语句，而后用附加表内容替换源表特征，最后生成进行附表清理语句。

附表化实例：

- 1) WalkSQL 语句

```
Select FeatureId, Zdh, Qlr, Obj.Area
```

```
From ZdFeatures
```

```
Where Obj.Area > 1000.0 AND 辖区=2
```

- 2) 经附表化扩展为一个 WalkSQLExTable 语句和两个 SQL 语句：

WalkSQLExTable (Create,

Area(ObjFeatureId, Obj\_Area),

ZdFeatures(FeaturId, Obj.Area))

GO

```
Select FeatureId, Zdh, Qlr, Area.Obj_Area
```

```
From Area, ZdFeatures
```

```
Where Area.ObjFeatureId = ZdFeatures.FeatureId AND
```

```
(Area.Obj_Area > 1000.0 AND 辖区=2)
```

GO

```
Drop Area
```

由于原 SQL 的 Where 中有 Obj，所以生成 WalkSQLExTable 时丢弃了整个条件。

- 3) 为提高效率，用户可重写 SQL 语句为：

```

WalkSQLExTable (Create,
                  Area(ObjFeatureId, Obj_Area),
                  ZdFeatures(FeaturId, Obj.Area),
                  ,
                  辖区=2)

GO

Select  FeatureId, Zdh, Qlr, Area.Obj_Area
From    Area, ZdFeatures
Where   Area.ObjFeatureId = ZdFeatures.FeatureId  AND
        (Area.Obj_Area > 1000.0)

GO

Drop   Area

```

由于在 WalkSQLExTable 中使用了附加条件 ‘辖区=2’，执行效率得到很大的提高。

## 5. 特征应用范围

### 5.1. 子句特征

```

Select:  OBJ, OBJGM, OBJGMS, OBJGEO
Where:   OBJ1, OBJ9I
Order by: OBJ1

```

### 5.2. 特征互斥

就是同一子句中不能同时出现的特征。基本规则是 1:1 的特征可以和其他特征一起使用，1:n 的特征在 n 值不同的情况下不能一起使用。OBJGMS 和 OBJGEO 比较特殊，限定不能与其他特征一起使用。

OBJ2~OBJGM2

OBJ2.CELLID~OBJ2 的其他特征

OBJGMS~其他任何特征

OBJGEO~其他任何特征

不同来源表的特征不能同时出现在 Select 和 Order by 语句中。

### 5.3. 附表化特征组织

- 1) 同一层次、同一表、特征 n 数相同的特征提取为一个特征表达函数表达。
- 2) 如果出现 OBJ9I, OBJ.MINX, OBJ.MINY, OBJ.MAXX, OBJ.MAXY 与 OBJ1 特征同时提取。

## 6. 附表化分类说明

在 WalkSQL 附表化以后，需要对 WalkSQLExTable() 中的特征进行内部解释，生成真正的附列表，并充实其内容。

### 6.1. OBJGEO

以 S.OBJGEO 为例，其结构为：

S (id, x, y, ann);

要将 S 表中的 ann（文字）按其点位（X，Y）构成 GEOMETRY，插入 Tfeatures 表，构成地物。

其 SQL 语句为：

```
Insert into TFeatures
    (Featureid, Geometry, Createtime, Styleid, Featurename)
Select    S.id, S.ObjGeo.Point(x,y,,id), Date(), 0, Left(S.ann,32)
FROM      S
```

#### 6.1.1. 附表化处理

```
WalkSqlExTable(CREATE,
    w_Geo_S_1(ID, Geometry),
    S(ID, OBJGEO.POINT(X,Y,,ID)),
    ,
    )
GO
INSERT INTO
    TFEATURES    (FEATUREID,    GEOMETRY,    CREATETIME,    STYLEID,
FEATURENAME)
SELECT
    S.ID, w_Geo_S_1.Geometry, DATE(), 0, LEFT(S.ANN,32)
FROM
    w_Geo_S_1, S
WHERE
    S.ID = w_Geo_S_1.ID
GO
Drop Table [w_Geo_S_1]
```

1) 附表表达函数及其参数示意如下：

```
WalkSqlExTable(
    Insert,
    G(ID, GEOMETRY),
    S(ID, OBJGEO.POINT(X, Y, , ID)),
    ,
    )
GO
Insert into TFeatures
    (Featureid, Geometry, Createtime, Styleid, Featurename)
Select    S.id, G.Geometry, Date(), 0, Left(S.ann,32)
FROM      G, S
WHERE     S.id = G.id
```

2) 将 S.OBJGEO.Point (...) 替换成 G.Geometry。

3) 在 Where 子句中添加 S. ID = G.FeatureID

#### 6.1.2. 附表实现

1) 生成一张附表 G := (ID, Geometry)。

- 2) G 记录填充
  - I. 在 WalkSQLExTable()内部执行查询
 

```
select ID, X, Y
from S
```
  - II. 从首记录开始遍历该视图
 ID 不同于前一记录，结束前一 Geomtry 并插入 G 表；构造新 Geometry。

## 6.2. OBJ1

以 LayerFeatures.OBJ1 为例。

Select FeatureID, Obj.Area

From AFeatures

### 6.2.1. 附表化处理

```
WalkSqlExTable(
  CREATE,
  w_Obj_A_1(ObjFeatureId, OBJ_AREA),
  AFEATURES(FeatureId, OBJ.AREA),
  ,
)
GO
SELECT
  FEATUREID, w_Obj_A_1.OBJ_AREA
FROM
  w_Obj_A_1, AFEATURES
WHERE
  w_Obj_A_1.ObjFeatureId = AFEATURES.FeatureId
GO
Drop Table [w_Obj_A_1]
```

- 1) 附表表达函数及其参数示意如下：

WalkSQLExTable (Create,

AssTable (FeatureID, OBJ\_Area),

AFeatures( FeatureID, OBJ.Area)。

- 2) 将 AFeatures.OBJ.Area 替换成 AssTable.特征字段。
- 3) 在 Where 子句中添加 AFeatures.FeatureID = AssTable.FeatureID。

### 6.2.2. 附表实现

- 1) 生成一张附表 AssTable := (FeatureID, {特征})。
 因为这是 1:1 的特征，每条记录可以以 FeatureID 为标识字段，特征字段可以多个。
- 2) AssTable 的记录填充过程。
  - I. 在 WalkSQLExTable()内部执行查询
 

```
Select FeatureID, Geometry
from LayerFeatures
```
  - II. 从首记录遍历该视图
    1. 调用 WalkGeom 函数计算各特征的单一值。
    2. 向 AssTable 插入一条特征值记录。

## 6.3. OBJ2

以 LayerFeatures.OBJ2 为例。

```
Select  FeatureID, Obj.PointX
```

```
From    LayerFeatures
```

#### 6.3.1. 附表化处理

```
WalkSqlExTable(  
    CREATE,  
    w_Obj_LAYER_1(ObjFeatureId, OBJ_POINTX),  
    LAYERFEATURES(FeatureId, OBJ.POINTX),  
    ,  
    )  
GO  
SELECT  
    FEATUREID, w_Obj_LAYER_1.OBJ_POINTX  
FROM  
    w_Obj_LAYER_1, LAYERFEATURES  
WHERE  
    w_Obj_LAYER_1.ObjFeatureId = LAYERFEATURES.FeatureId  
GO  
Drop Table [w_Obj_LAYER_1]
```

1) 附表表达函数及其参数示意如下：

```
WalkSQLExTable (Create,  
                AssTable(WALKSQL_AUTOID, FeatureID, OBJ2_Value),  
                LayerFeatures( , FeatureID, OBJ.POINTX))
```

2) 将 LayerFeatures. OBJ.POINTX 替换成 AssTable.特征字段。

3) 在 Where 子句中添加 LayerFeatures.FeatureID = AssTable.FeatureID。

#### 6.3.2. 附表实现

- 1) 生成一张附表 AssTable := (WALKSQL\_AUTOID, FeatureID, {特征})。  
因为这是 1:n 的特征, 每条记录需要以 WALKSQL\_AUTOID 为标识字段, FeatureID 用来区分记录来源地物, 特征字段可以多个。
- 2) AssTable 的记录填充过程。
  - I. 在 WalkSQLExTable()内部执行查询  

```
Select FeatureID, Geometry  
from LayerFeatures
```
  - II. 从首记录遍历该视图
    1. 调用 WalkGeom 函数计算各特征的 n 个值。
    2. 循环插入 n 条特征值记录。

#### 6.4. OBJGM1

以 LayerFeatures.OBJGM1 为例。

```
Select  FeatureID, Obj.Gm_Box
```

```
From    LayerFeatures
```

#### 6.4.1. 附表化处理

```
WalkSqlExTable(  
    CREATE,  
    w_Obj_LAYER_1(ObjFeatureId, OBJ_GM_BOX),
```

```

        LAYERFEATURES(FeatureId, OBJ.GM_BOX),
    ,
)
GO
SELECT
    FEATUREID, w_Obj_LAYER_1.OBJ_GM_BOX
FROM
    w_Obj_LAYER_1, LAYERFEATURES
WHERE
    w_Obj_LAYER_1.ObjFeatureId = LAYERFEATURES.FeatureId
GO
Drop Table [w_Obj_LAYER_1]

```

1) 附表表达函数及其参数示意如下:

WalkSQLExTable (Create,

AssTable(FeatureID, OBJGM1\_Value),

LayerFeatures(FeatureID, OBJGM1..).

2) 将 LayerFeatures.OBJGM1 替换成 AssTable.特征字段。

3) 在 Where 子句中添加 LayerFeatures.FeatureID = AssTable.FeatureID。

#### 6.4.2. 附表实现

1) 生成一张附表 AssTable := (FeatureID, {特征})。

因为这是 1:1 的特征，每条记录可以以 FeatureID 为标识字段，特征字段可以多个。

2) AssTable 的记录填充过程。

I. 在 WalkSQLExTable()内部执行查询

Select FeatureID, Geometry

from LayerFeatures

II. 从首记录遍历该视图

1. 调用 WalkGeom 函数计算各特征的单一几何特征体。

2. 向 AssTable 插入一条特征几何特征体记录。

#### 6.5. OBJGM2

以 LayerFeatures.OBJGM2 为例。

Select FeatureID, Obj.Gm\_Points

From LayerFeatures

##### 6.5.1. 附表化处理

```

WalkSqlExTable(
    CREATE,
    w_Obj_LAYER_1(ObjFeatureId, OBJ_GM_POINTS),
    LAYERFEATURES(FeatureId, OBJ.GM_POINTS),
    ,
)
GO
SELECT
    FEATUREID, w_Obj_LAYER_1.OBJ_GM_POINTS
FROM
    w_Obj_LAYER_1, LAYERFEATURES

```

WHERE

w\_Obj\_LAYER\_1.ObjFeatureId = LAYERFEATURES.FeatureId

GO

Drop Table [w\_Obj\_LAYER\_1]

1) 附表表达函数及其参数示意如下:

WalkSQLExTable (Create,

AssTable(WALKSQL\_AUTOID, FeatureID, OBJGM2\_Value),

LayerFeatures(FeatureID, OBJGM2...))。

2) 将 LayerFeatures.OBJGM2 替换成 AssTable.特征字段。

3) 在 Where 子句中添加 LayerFeatures.FeatureID = AssTable.FeatureID。

### 6.5.2. 附表实现

1) 生成一张附表 AssTable := (WALKSQL\_AUTOID, FeatureID, {特征})。

因为这是 1:n 的特征, 每条记录需要以 WALKSQL\_AUTOID 为标识字段, FeatureID 用来区分记录来源地物, 特征字段可以多个。

2) AssTable 的记录填充过程。

I. 在 WalkSQLExTable() 内部执行查询

Select FeatureID, Geometry  
from LayerFeatures

II. 从首记录遍历该视图

1. 调用 WalkGeom 函数计算各特征的 n 个几何特征体。

2. 循环插入 n 条几何特征体记录。

## 6.6. OBJ9I

以 Afeatures 和 Bfeatures 的 Contain 特征为例。

Select Afeatures.featureid, Afeatures.obj.area, Afeatures.obj.perimeter

From Afeatures, Bfeatures

where OBJ9I.Contain(Bfeatures, Afeatures)

### 6.6.1. 附表化处理

//求 Bfeatures 表的地物几何外接盒 BOX

WalkSqlExTable(CREATE,

w\_MM\_B\_9(FEATUREID, OBJ\_MINX, OBJ\_MINY, OBJ\_MAXX, OBJ\_MAXY),

BFEATURES(FEATUREID, OBJ.MINX, OBJ.MINY, OBJ.MAXX, OBJ.MAXY),

,

)

GO

//求 Afeatures 表的地物几何外接盒 BOX

WalkSqlExTable(CREATE,

w\_MM\_A\_9(FEATUREID, OBJ\_MINX, OBJ\_MINY, OBJ\_MAXX, OBJ\_MAXY),

AFEATURES(FEATUREID, OBJ.MINX, OBJ.MINY, OBJ.MAXX, OBJ.MAXY),

,

)

GO

//利用 Afeatures 和 Bfeatures 的 Box 关系缩小选择范围, 建立 A 与 B 的 9i 关系

WalkSqlExTable(CREATE,

w\_CONTAIN\_B\_A\_1(L1Id, L2Id),

```

OBJ9I.CONTAIN(BFEATURES, AFEATURES),
(w_MM_B_9, w_MM_A_9),
[BFEATURES].FEATUREID=[w_MM_B_9].FEATUREID      AND
[AFEATURES].FEATUREID=[w_MM_A_9].FEATUREID      AND
[w_MM_B_9].OBJ_MINX<=[w_MM_A_9].OBJ_MINX        AND
[w_MM_B_9].OBJ_MINY<=[w_MM_A_9].OBJ_MINY        AND
[w_MM_B_9].OBJ_MAXX>=[w_MM_A_9].OBJ_MAXX        AND
[w_MM_B_9].OBJ_MAXY>=[w_MM_A_9].OBJ_MAXY)
/* obj9i.result(...)
WalkSqlExTable(CREATE,
w_CONTAIN_B_A_1(L1Id, L2Id, Mask9i),
OBJ9I.Result(BFEATURES, AFEATURES, <Mask9i 常量>),
(w_MM_B_9, w_MM_A_9),
[BFEATURES].FEATUREID=[w_MM_B_9].FEATUREID      AND
[AFEATURES].FEATUREID=[w_MM_A_9].FEATUREID      AND
[w_MM_B_9].OBJ_MINX<=[w_MM_A_9].OBJ_MINX        AND
[w_MM_B_9].OBJ_MINY<=[w_MM_A_9].OBJ_MINY        AND
[w_MM_B_9].OBJ_MAXX>=[w_MM_A_9].OBJ_MAXX        AND
[w_MM_B_9].OBJ_MAXY>=[w_MM_A_9].OBJ_MAXY)
*/
GO
WalkSqlExTable(
    CREATE,
    w_Obj_A_1(ObjFeatureId, OBJ_AREA, OBJ_PERIMETER),
    AFEATURES(FeatureId, OBJ.AREA, OBJ.PERIMETER),
    ,
)
GO
SELECT
    AFEATURES.FEATUREID,
    w_Obj_A_1.OBJ_AREA,
    w_Obj_A_1.OBJ_PERIMETER
FROM
    w_Obj_A_1, w_CONTAIN_B_A_1, AFEATURES, BFEATURES
WHERE
    w_Obj_A_1.ObjFeatureId = AFEATURES.FeatureId  AND
    (AFEATURES.FeatureId = w_CONTAIN_B_A_1.L2Id  AND
    BFEATURES.FeatureId = w_CONTAIN_B_A_1.L1Id)
GO
Drop Table [w_MM_B_9]
GO
Drop Table [w_MM_A_9]
GO
Drop Table [w_CONTAIN_B_A_1]

```



GO

Drop Table [w\_Obj\_A\_1]

### 6.6.2. 附表实现

- 1) 生成一张附表 F={AFID, BFID}。

表示 Afeatures 表中 FeatureID 为 AFID 的地物，与之满足 9I 条件的 Bfeatures 表中的地物的 FeatureID 为 BFID。

- 2) AssTable 的记录填充过程

- I. 在 WalkSQLExTable()内部执行查询

1. Select featureid, geometry

- From LayerAFeatures

- 作为视图 Aview。

2. Select featureid, geometry

- From LayerBFeatures

- 作为视图 Bview。

- II. 计算视图中每个地物的上下左右边界，如果 WalkSQLExTable 参数里已经提供了过滤条件，则不重新计算上下左右边界，仅利用参数中提供的表内容。

- III. 对 Aview 记录循环

- 在 Bview 中边界相连的范围那计算 9I 关系。

### 6.7. OBJGMS

以 LayerFeatures 表，以字段 SID 为统计字段，LayerFeatures.UNION(SID)为例。

Insert into TFeatures (Featureid, styleid, Geometry, createtime)

Select sid, 0, LayerFeatures.ObjGms.Union(sid), Date()

From LayerFeatures

#### 6.7.1. 附表化处理

WalkSqlExTable(CREATE,

w\_Gms\_LAYERFEATURES\_1(SID, Geometry),

LAYERFEATURES(SID, OBJGMS.UNION(SID)),

,

)

GO

INSERT INTO

TFEATURES (FEATUREID, STYLEID, GEOMETRY, CREATETIME)

SELECT

SID, 0, w\_Gms\_LAYERFEATURES\_1.Geometry, DATE()

FROM

w\_Gms\_LAYERFEATURES\_1, LAYERFEATURES

WHERE

LAYERFEATURES.SID = w\_Gms\_LAYERFEATURES\_1.SID

GO

Drop Table [w\_Gms\_LAYERFEATURES\_1]

- 1) 附表表达函数及其参数示意如下：

WalkSQLExTable(Create,

AssTable(Featureid, Geometry),

```

LayerFeatures(sid , OBJGMS.UNION(sid)),
,
)

```

- 2) 把 LayerFeatures.OBJGMS.UNION 替换为 AssTable.Geometry。
- 3) 在 Where 子句中添加 LayerFeatures.SID = AssTable.SID 的条件。

```

Insert into TFeatures (Featureid, styleid, Geometry, createtime)
Select Featureid, 0, Geometry, Date()
From AssTable

```

### 6.7.2. 附表实现

- 1) 生成一张附表 F={SID, Geometry}。  
表示统计条件为 SID 的地物几何为 Geometry。
- 2) AssTable 的记录填充过程
  - I. 在 WalkSQLExTable()内部执行查询
 

```

Select featureid, sid, geometry
From LayerFeatures
Order by SID

```
  - II. 遍历该试图
    1. 遇到不同的 SID, 对之前一个 SID 的所有几何执行统计操作。
    2. 向 AssTable 插入一条记录

## 7. 典型 WalkSQL 附表化

地物内容以 A 和 B 两层为例,其层地物表名为 AFeatures 和 Bfeatures;附表操作以 Create 为例, Insert 与 Update 操作类似;条件用表以 ConditionTable 为例。

### 7.1. 单表单类实例

一种比较单纯和常用的 WalkSQL。

#### 7.1.1. 基本特征

#### 7.1.2. 统计特征

### 7.2. 多表多类

### 7.3. 嵌套语句

## 8. 注记解析特征

从层注记表的 Annotation 字段解析特征,目前仅支持 1:1 特征,定位及其处理类似与地物特征 OBJ1。

ANN.TEXT,	//文本串
ANN.TEXTHEIGHT,	//文本串的子高
ANN.TEXTANGLE,	//文本串的角度
ANN.TEXTFEATUREID,	//FeatureID
ANN.LOCTYPE,	//定位类型
ANN.LOCX,	//定位点X

ANN.LOCY, //定位点Y  
允许出现这些特征的WalkSQL子句包括select, where, order by。

## 9. 式样解析特征

从层式样表达的 Style 字段解析特征，目前仅支持 1:1 特征，定位及其处理类似与地物特征 OBJ1。

SYM.STYPE, //式样类型  
SYM.SCOLOR, //式样颜色  
SYM.SUSERID, //用户自定义式样在SymbolFactory中的id，若无为0  
允许出现这些特征的WalkSQL子句包括select, where, order by。