空间数据库原理及开发技术

第3章 空间数据管理

中南大学地信系 李光强
QQ：41733233

教学目标

- 熟悉空间参考系的概念及其作用
- 掌握矢量、栅格两类数据的管理模式
- 熟悉空间元数据的概念及其作用
- 熟悉PostGIS管理矢量数据的常用方法

教学重点

- 矢量数据管理模式
- 栅格数据管理模式

教学内容

- 3.1 空间参考系
- 3.2 矢量数据管理模式
- 3.3 栅格数据管理模式
- 3.4 空间元数据管理
- 3.5 PostGIS管理矢量数据

3.1 空间参考系

- 3.1.1 空间参考系概念
- 3.1.2 空间参考系表达
- 3.1.3 空间参考系相关操作

3.1.1 空间参考系概念

- 只有使用统一的坐标系统作为空间参照系，表达和存储空间要素的地理位置和空间属性，才能保证空间数据的交换和共享，不同要素类之间才能进行空间分析、计算和处理。
- ISO组织的定义
  - 将**坐标系**（coordinate system，CS）定义为用于分配空间位置坐标值的数学规则集
  - 将**空间参考**（spatial reference，SR）定义为真实世界中位置的描述
  - 将**参考系统**（reference system， RS）定义为准确说明空间位置信息所需的元数据
  - 将**坐标参考系**（coordinate reference system，CRS） 定义为空间对象相关的坐标系基准，包括原点位置的参数或参数集、尺度、坐标系/轴的方向

3.1.1 空间参考系概念

- OGC组织的定义
  - 继承和扩展了ISO的坐标参考系的定义，将坐标参考系称之为**空间参考系**（spatial reference system，SRS），还将其定义为用于精确测量地球表面位置坐标的一种框架，由坐标系和与地理空间相关的抽象数学参数组成，包括
    - 地球椭球体
    - 水平基准面
    - 地图投影（地理坐标系除外）
    - 原点
    - 测量单位

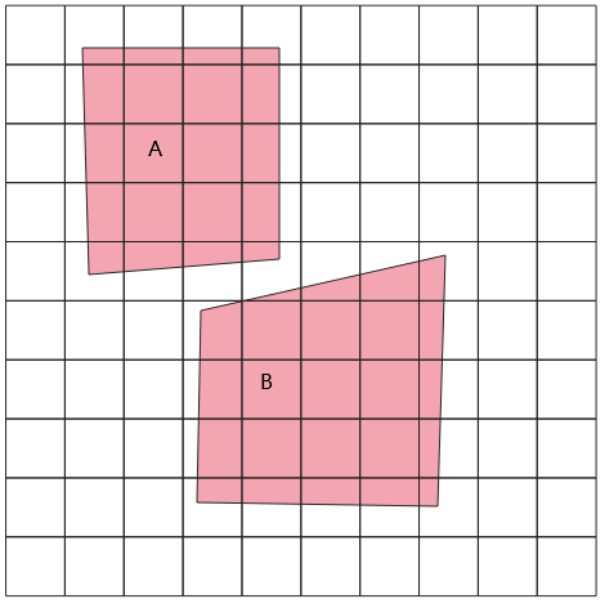
3.1.1 空间参考系概念

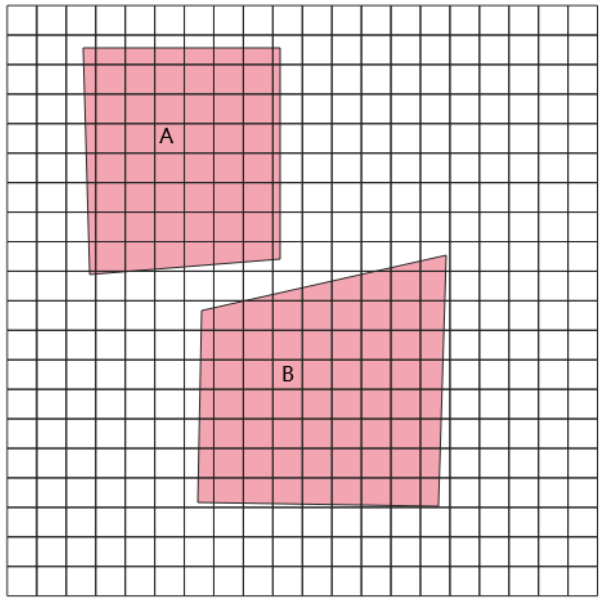
- 在空间数据库领域，**空间参考系**是用于存储空间要素类、栅格数据集和其它具有空间属性数据的基础，由坐标系、空间位置分辨率和容差组成
  - **(1)坐标系**：ISO将其定义为在*n*维度空间中，能够使用一组*n*个标量描述一个点的空间位置，这个*n*个标量的定义就构成对应的坐标系统。
    - **地理坐标系**：（又称大地坐标系）使用纬度（赤道以北或以南的度数）和经度（本初子午线以西或以东的度数）直接测量地球上位置的球坐标系
    - **地心坐标系**：是在地球体内建立的O-XYZ三维坐标系，原点O设在地球质心。X轴与首子午面与赤道面的交线重合，向东为正；Z轴与地球旋转轴重合，向北为正；Y轴与XZ平面垂直构成右手系
    - **投影坐标系**：又称平面坐标系或网格坐标系，是一种标准化的笛卡尔坐标系，将地球表面的一定区域模拟为一个平面，从任意原点O沿 X 和 Y 轴测量位置。
    - **工程坐标系**：又称本地坐标系或自定义坐标系，为小区域（通常是单个工程项目）定制的笛卡尔坐标系，地球的表面可以完全近似为平面，通过设定原点和坐标轴方向，测量小区域的空间位置

3.1.1 空间参考系概念

空间参考系组成(续)

- **(2)分辨率**：记录地理要素位置和形状时所使用的空间详细程度，是分隔要素坐标各分量值的最小距离。
  - 空间参考系的分辨率对要素的表达和要素空间关系都将产生重要影响。高空间分辨率值（网格单元尺寸较小）能够更加精细地反映地物细节
  - 不同分辨率下的空间关系可能会不同





3.1.1 空间参考系概念

- 空间参考系组成(续)
  - **(3)容差**：指位置坐标值之间的最小距离，也就是空间数据库管理系统能够区分的最小空间间隔。
    - 在ESRI的GeoDatabase模型中，容差是空间参考系的重要参数，是坐标之间的最小距离。
    - 例如，点要素P1在点要素P2的容差值范围内，则会将P1和P2视为同一位置。

3.1.2 空间参考系表达

(1)空间参考WKT表达

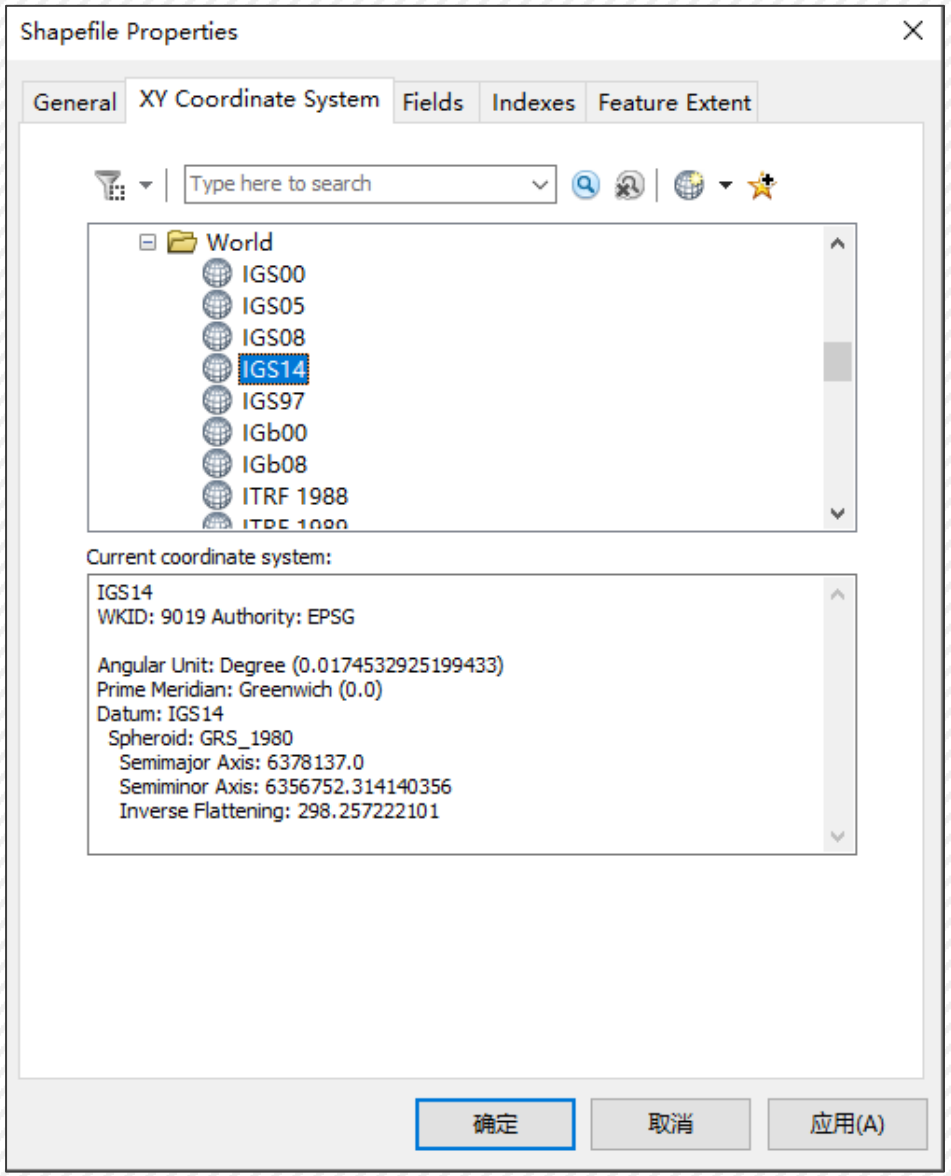
```
PROJCRS["WGS 84 (G1762) / UTM zone 31N 3D",
  BASEGEOGCRS["WGS 84",
    DATUM["World Geodetic System of 1984 (G1762)",
      ELLIPSOID["WGS 84",6378137, 298.257223563,LENGTHUNIT["metre",1.0]]
    ]
  ],
  CONVERSION["UTM zone 31N 3D",
    METHOD["Transverse Mercator (3D)"],
    PARAMETER["Latitude of origin",0.0,
      ANGLEUNIT["degree",0.0174532925199433]],
    PARAMETER["Longitude of origin",3.0,
     ANGLEUNIT["degree",0.0174532925199433]],
    PARAMETER["Scale factor",0.9996,SCALEUNIT["unity",1.0]],
    PARAMETER["False easting",500000.0,LENGTHUNIT["metre",1.0]],
    PARAMETER["False northing",0.0,LENGTHUNIT["metre",1.0]]
  ],
  CS[Cartesian,3],
    AXIS["(E)",east,ORDER[1]],
    AXIS["(N)",north,ORDER[2]],
    AXIS["ellipsoidal height (h)",up,ORDER[3]],
    LENGTHUNIT["metre",1.0]
]
```

3.1.2 空间参考系表达

- (1)空间参考WKT表达(续)
  - ESRI的SHP文件里的prj文件使用WKT表达地图投影系，示例：

GEOGCS[“GCS_WGS_1984”,DATUM[“D_WGS_1984”,SPHEROID[“WGS_1984”,6378137.0,298.257223563]],PRIMEM[“Greenwich”,0.0],UNIT[“Degree”,0.0174532925199433],AUTHORITY[“EPSG”,4326]]

  - ESRI为每个定义的空间参考系WKT设定了唯一编码，称为WKID



3.1.2 空间参考系表达

- (2)空间参考PROJ表达
  - PROJ 是一个坐标转换程序，但PROJ程序使用Proj文本表达空间参考系
  - PROJ语法简洁明了，已被空间数据管理领域广泛接受和使用。
  - 例如，WGS-84空间参考系的PROJ文本定义如下：

```
+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs
```

3.1.2 空间参考系表达

- (3)空间参考系标识
  - 欧洲石油测绘组 (European Petroleum Survey Group， EPSG)收集整理了全球通用的空间参考系参数和坐标转换描述，并给每个空间参考系设定了唯一的编号，称为**空间参考标识**（spatial reference system identifier, **SRID**）。
  - EPSG编制的空间参考系参数及其SRID已被PostGIS、Oracle spatial、SQL Server spatial和mySQL spatial等空间数据库管理系统使用。
  - 例如
    - GCS_WGS_1984坐标系的SRID=4326
    - GCS_China_Geodetic_Coordinate_System_2000坐标系的SRID=4490

3.1.2 空间参考系表达

- (4)空间数据库中空间参考系的管理
  - OGC的SFA规范规定了空间参考系在空间数据库中的管理方法，定义了**SPATIAL_REF_SYS**数据表结构，用于存储已定义的空间参考系的信息。
  - **SPATIAL_REF_SYS**数据表包括以下字段：
    - （1）SRID ：空间参考系统标识符，主码。
    - （2）AUTH_NAME ：空间参考系统机构名称。
    - （3）AUTH_SRID ：权限特定空间参考系统标识符。
    - （4）SRTEXT ： 空间参考系的WKT描述。

3.1.2 空间参考系表达

中南大学 CENTRAL SOUTH UNIVERSITY

SPATIAL_REF_SYS数据表数据示例：

srid	auth_name	auth_srid	srtext	proj4text
3819	EPSG	3819	GEOGCS["HD1909",DATUM["Hungarian_Datum_1909'	+proj=longlat +ellps=bessel +towgs84=595.48,121.69,515.35,4.115,-2.9383,0.853,-3.408 +no
3821	EPSG	3821	GEOGCS["TWD67",DATUM["Taiwan_Datum_1967",SPH	+proj=longlat +ellps=aust_SA +no_defs
3824	EPSG	3824	GEOGCS["TWD97",DATUM["Taiwan_Datum_1997",SPH	+proj=longlat +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +no_defs
3889	EPSG	3889	GEOGCS["IGRS",DATUM["Iraqi_Geospatial_Reference_	+proj=longlat +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +no_defs
3906	EPSG	3906	GEOGCS["MGI 1901",DATUM["MGI_1901",SPHEROID[	+proj=longlat +ellps=bessel +towgs84=682,-203,480,0,0,0,0 +no_defs
4001	EPSG	4001	GEOGCS["Unknown datum based upon the Airy 1830	+proj=longlat +ellps=airy +no_defs
4002	EPSG	4002	GEOGCS["Unknown datum based upon the Airy Modi	+proj=longlat +ellps=mod_airy +no_defs
4003	EPSG	4003	GEOGCS["Unknown datum based upon the Australian	+proj=longlat +ellps=aust_SA +no_defs
4004	EPSG	4004	GEOGCS["Unknown datum based upon the Bessel 184	+proj=longlat +ellps=bessel +no_defs
4005	EPSG	4005	GEOGCS["Unknown datum based upon the Bessel Mc	+proj=longlat +a=6377492.018 +b=6356173.508712696 +no_defs
4006	EPSG	4006	GEOGCS["Unknown datum based upon the Bessel Na	+proj=longlat +ellps=bess_nam +no_defs
4007	EPSG	4007	GEOGCS["Unknown datum based upon the Clarke 185	+proj=longlat +a=6378293.645208759 +b=6356617.987679838 +no_defs
4008	EPSG	4008	GEOGCS["Unknown datum based upon the Clarke 186	+proj=longlat +ellps=clrk66 +no_defs
4009	EPSG	4009	GEOGCS["Unknown datum based upon the Clarke 186	+proj=longlat +a=6378450.047548896 +b=6356826.621488444 +no_defs
4010	EPSG	4010	GEOGCS["Unknown datum based upon the Clarke 188	+proj=longlat +a=6378300.789 +b=6356566.435 +no_defs
4011	EPSG	4011	GEOGCS["Unknown datum based upon the Clarke 188	+proj=longlat +a=6378249.2 +b=6356515 +no_defs
4012	EPSG	4012	GEOGCS["Unknown datum based upon the Clarke 188	+proj=longlat +ellps=clrk80 +no_defs
4013	EPSG	4013	GEOGCS["Unknown datum based upon the Clarke 188	+proj=longlat +a=6378249.145 +b=6356514.966398753 +no_defs
4014	EPSG	4014	GEOGCS["Unknown datum based upon the Clarke 188	+proj=longlat +a=6378249.2 +b=6356514.996941779 +no_defs
4015	EPSG	4015	GEOGCS["Unknown datum based upon the Everest 18	+proj=longlat +a=6377276.345 +b=6356075.41314024 +no_defs

3.1.3 空间参考系相关操作

- ISO和OGC组织不仅定义了空间参考系及其表达形式，还定义了空间参考系常用的操作
  - **ST_SRID**操作用于查看或显示空间数据的空间参考系标识。例如：

    SELECT **ST_SRID**(geom) FROM building
  - **ST_SetSRID**操作用于设置空间数据的空间参考标识。例如:

    UPDATE building SET geom=**ST_SetSRID**(geom,3857);
  - **ST_Transform**操作将原空间参考系的空间数据转换到新空间参考系的坐标值，并将SRID设置为新SRID。例如：

    UPDATE building SET geom= **ST_Transform**(geom,4326);

3.2 矢量数据管理模式

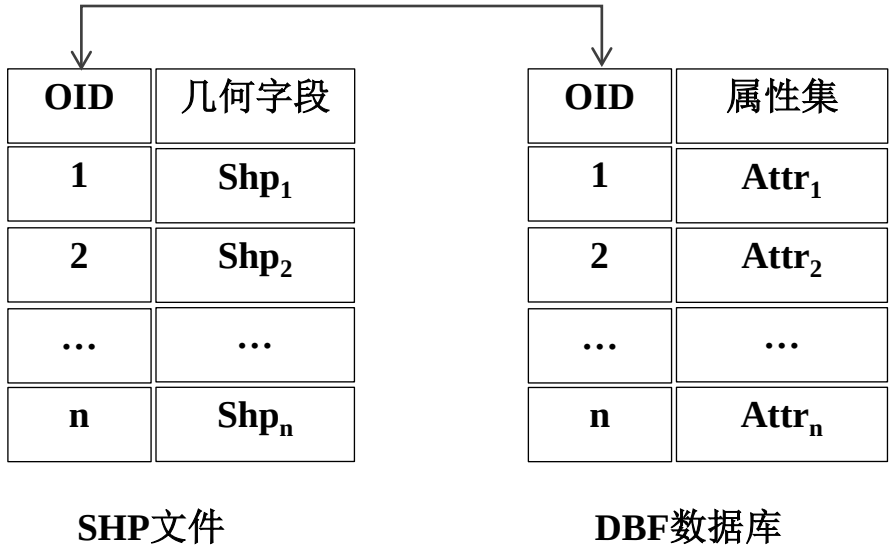
- 3.2.1 文件管理模式
- 3.2.2 文件数据库混合管理模式
- 3.2.3 全关系数据库管理模式
- 3.2.4 空间数据网关管理模式
- 3.2.5 对象关系数据库管理模式

3.2.1 文件管理模式

- 空间数据文件管理模式是按照特定数据结构表达和存储空间要素的几何信息及其专题属性的文件管理方式。
- 常用的格式有
  - MapInfo文件
  - DWG文件
  - MapGIS文件
  - KML和KMZ文件
  - GeoXML
  - GeoJSON文件

3.2.2 文件数据库混合管理模式

- 文件-数据库混合管理模式使用几何图形文件存储要素的几何信息，使用关系数据库存储要素的属性信息，二者使用要素的标识码建立关联。
- ESRI的Shape存储模式是一种典型的文件-数据库管理模式，空间几何数据存储在shp文件（简称形文件）中，属性数据存储在关系数据库dbf中，二者利用要素ObjectID进行关联。

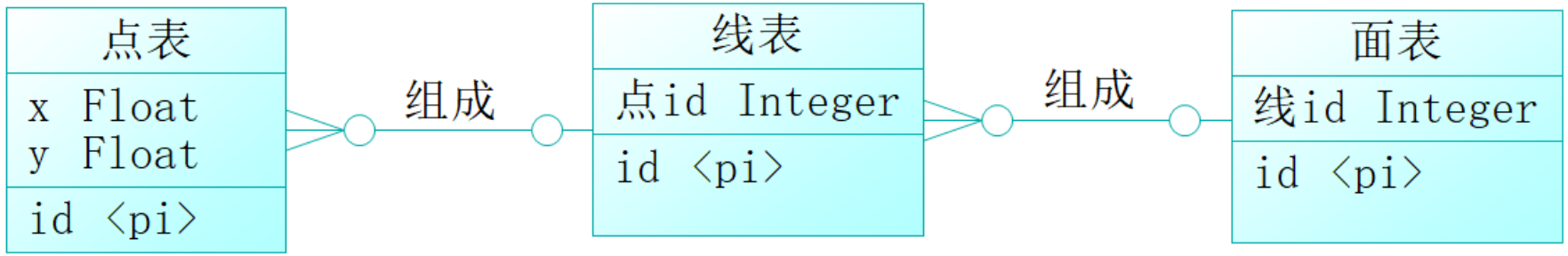


3.2.2 文件数据库混合管理模式

- 混合管理模式的特点
  - 文件结构与应用程序相对独立，空间数据的管理不受应用程序的制约，应用较为方便、快捷，可以简单、灵活地表达现实世界各种实体及其相互间的关系；
  - 具有严密的数学基础和操作代数基础，数据间的关系具有对称性；
  - 使用表格管理属性数据，方便查询。
- 缺点
  - 不利于空间数据的共享和并发访问，难以适应网络环境下的空间数据共享和互操作；
  - 空间数据和属性数据分离管理，数据冗余且数据重复；
  - 数据缺乏集中管理，文件缺乏弹性，数据的维护只能由应用程序承担；
  - 数据的安全性和完整性无法保障

3.2.3 全关系数据库管理模式

- 全关系数据库管理模式是用关系型数据库存储和管理空间要素数据的方式。
- 根据空间数据的组成特点，一个要素类可以映射成两类不同的关系数据表，即**专题属性表和空间图形表**，利用主码与外码联接组成要素类的全部信息。
- 非结构化的空间图形数据来说，需使用多张数据表关联存储空间图形数据，包括点表、线表、面表等，



3.2.3 全关系数据库管理模式

- 特点
  - 将空间数据全部映射并存储到关系型数据库中，数据管理简单，数据冗余小，节省存储空间；
  - 可以充分发挥数据库管理的优点，保障数据安全，提高数据并发访问效率；
  - 可以直接使用关系型结构化查询语句查询空间数据，提高了数据访问的便捷性。
- 缺点
  - 对于线要素、面要素来说，数据访问需要多次的连接查询，大大降低数据访问效率；
  - 数据表之间的复杂关联关系影响了空间数据的维护

3.2.4 空间数据网关管理模式

- 空间数据网关（spatail data gateway）将空间几何数据转换为特殊的数据结构并存储在关系数据库中，实现了空间几何数据和属性数据的一体化存储。
- 例如
  - ESRI公司的空间数据引擎（Spatial Data Engine，SDE）是使用最广泛的空间数据网关之一。ArcSDE使用OGC制定的WKB（well known binary）数据结构存储空间图形数据，然后将WKB存入数据库的长二进制字段.

3.2.4 空间数据网关管理模式

- 特点
  - 能够使用关系数据库一体化、集中存储空间数据的几何图形和属性信息，省去了图形数据与属性数据之间的繁琐联结，存取速度快；
  - 使用网关转换几何图形数据，能够将GIS与数据管理分离，提高了GIS应用程序的开发效率。
- 缺点
  - 受到数据网关的限制，用户需要支付购买数据网关软件的费用，数据部署和数据迁移较困难。

3.2.5 对象关系数据库管理模式

- 基于ORDB的空间数据库管理系统扩展空间几何字段类型，研发支持OGC空间查询的空间操作函数，实现空间数据的结构化查询、分析等操作。
- 空间要素的专题属性可以直接使用关系型数据库的字段存储，使得空间要素能够集中存储在一条记录中，已成为空间数据库的主流管理模式：
  - 微软公司的SQL Server、甲古文公司的Oracle，以及开源数据库PostGIS和MySQL，都具有这种空间数据的管理功能。
- 基于ORDB的空间数据管理模式的空间要素可以表达为：

$$\text{Feature}=\{A_1,A_2,\ldots A_n,\text{geom}\}$$

3.2.5 对象关系数据库管理模式

中南大学 CENTRAL SOUTH UNIVERSITY

- 特点
  - 数据管理更加简单，使用扩展字段存储空间图形数据，更加节省存储空间；
  - 避免了空间要素多次联接查询，进一步提升空间要素检索效率；
  - 使用空间查询语言可以让上层GIS应用程序省去空间计算的算法设计和编码花销，增强了空间查询和分析的稳定性。
- 缺点
  - 对象关系数据库管理模式的空间图形类型不能由用户定义，复杂的分析计算需要由用户自己开发。

3.3 栅格数据管理模式

- 3.3.1 文件管理模式
- 3.3.2 文件-数据库混合管理模式
- 3.3.3 数据库管理模式

3.3.1 文件管理模式

- GIS领域常用的栅格数据类型有很多种，包括卫星影像、数字高程模型、数字正射影像、扫描文件、数据栅格图形、不规则三角网等
- 常见的栅格文件有BMP、DEM、ENVI文件、ERDAS文件、GDB、GRD、JPG、PNT、SRTM格式、TIFF等.
- 常用的栅格数据管理结构有栅格数据集和镶嵌数据集。
  - **栅格数据集**也是多波段栅格数据的组织格式，每个波段由一系列像素（像元）数组组成，每个像素都有一个值。
  - **镶嵌数据集**是一组以目录形式存储并以单个镶嵌影像或单个影像（栅格）的方式显示或访问的栅格数据集（影像）。

3.3.2 文件–数据库混合管理模式

- 典型代表是ESRI的个人地理数据库管理栅格数据，栅格数据集被转换为 image (.img) 文件，存储到图像数据库 (image database, IDB) 文件夹中。
- 栅格数据集的配置参数存储在个人地理数据库的数据表中。
- 个人地理数据库中的业务表的每行都指向已存储的栅格数据集，即存储栅格镶嵌数据的路径位置，所以对镶嵌数据集的操作不会影响已存储的栅格文件

3.3.3 数据库管理模式

- 这种管理模式能够将像素值、空间参考和其他元数据一并存储在关系型数据库中.
- 在将栅格数据转换存到数据库时，系统将根据用户定义的尺寸（如128×128pix）均匀地将每个波段数据分割为像素块，从而实现栅格数据的高效存储和检索。
- PostGIS管理栅格数据示例
  - PostGIS数据库系统自带了raster2pgsql命令，使用该命令可以将栅格数据转换并存入PostGIS数据库。

3.4 空间元数据管理

- 3.4.1空间元数据
- 3.4.2 空间元数据的管理

3.4.1 空间元数据

- **元数据**（Meta data）
  - 元数据是描述其它数据的数据，包括数据资源的功能特征、访问模式、访问结果和运行性能等。
  - 元数据是识别数据资源的基础，为信息资源建立机器可理解的框架体系，是数据传输、共享、交换和互操作的基础。
  - 元数据还是评价数据资源的依据，有助于实现信息资源的有效发现和查找。
  - 元数据是从信息资源中抽取出来用于说明其特征、内容的结构化数据。

3.4.1 空间元数据

空间元数据

- ISO/TC 211将其定义为：空间元数据是“关于空间数据内容、质量、条件及其它特征的数据”。
- 空间元数据（spatial meta data）是描述空间数据内容、质量、表示方法、空间参考、管理方式、操作方法等特征的数据，是实现地理空间信息理解、评价和共享的基础。

3.4.1 空间元数据

空间元数据的作用

- （1）空间元数据标识了空间数据的名称、数据生产单位、管理单位、数据内容、现势性、数据获取和使用的限制等。
- （2）空间元数据描述了数据质量信息，包括空间位置精度、垂直精度、属性精度、逻辑一致性、完备性等特征。
- （3）空间元数据描述了空间表示类型、矢量空间表示信息、栅格空间表示类型和影像空间表示类型等信息。
- （4）空间元数据描述了空间数据的空间参照系类型、地理标识码标准系统、高程基准、比例尺、空间范围等。
- （5）空间元数据描述了要素的类型定义、类型名称、类型标识码、属性定义、属性名称和属性语义等。

3.4.2 空间元数据的管理

- 空间元数据组织方式
  - GIS领域使用图层管理和存储空间数据，每个图层（要素类）包括若干个空间要素。
  - 空间元数据包括数据库级元数据、空间数据集级元数据、图层级元数据、要素级元数据等多个层次。
- OGC定义的空间元数据
  - OGC的SFA规范定义了GEOMETRY_COLUMNS和SPATAIL_REF_SYS空间元数据表.
    - **GEOMETRY_COLUMNS**数据表存储了数据库中所有空间数据表的描述信息
    - **SPATIAL_REF_SYS**表定义了空间数据库已知的所有空间参照系统

3.5 PostGIS 管理矢量数据

- 3.5.1 PostGIS的矢量数据类型
- 3.5.2 空间数据表管理
- 3.5.3 空间矢量数据的管理
- 3.5.4 空间矢量的数据访问与输出

3.5.1 PostGIS的矢量数据类型

- 矢量数据类型
  - PostGIS支持的OGC-SFA定义的全部矢量类型，包括**地理图形**（geography）和**几何图形**（geometry）基本类型。
  - （1）地理图形数据类型
    - Geography数据类型使用大地坐标系（或称为地理坐标）存储全球尺度的空间数据，大地坐标系使用角度单位。
  - （2）几何图形数据类型
    - Geometry数据类型通常使用投影坐标系（或称平面坐标系）存储局部较小地理范围的空间数据，投影坐标常用米（meter）、千米（kilometer）等为单位。

3.5.2 空间数据表管理

- 空间数据列类型
  - 空间数据表是指至少包含一个Geography或Geometry字段用于存储空间矢量数据的数据表。
  - PostGIS的Geometry字段类型
    - **异构空间列**（heterogeneous spatial columns）
    - **同构空间列**（homogeneous spatial columns）
    - **继承**（inheritance）
    - **分区列**（partitioning）

3.5.2 空间数据表管理

- 空间数据表的创建
  - 异构空间列数据表的创建语法如下：

```
CREATE TABLE [IF NOT EXISTS] <空间表名>
(
  [<属性列名>  <数据类型>[<列级完整性约束条件> ],]
  <几何列名>  geometry [ <完整性约束条件> ]
  [,<表级完整性约束条件>]
);
```

3.5.2 空间数据表管理

- 空间数据表的创建
  - 示例：

```
CREATE TABLE IF NOT EXISTS road (
        id              serial4             PRIMARY KEY,
        road_name       VARCHAR ( 50 )      NOT NULL,
        road_code       VARCHAR ( 10 )      NOT NULL,
        road_grade      INT                 DEFAULT 0,
        road_class      INT                 DEFAULT 0,
        geom            geometry            NOT NULL
);
```

3.5.2 空间数据表管理

- 空间数据表的创建
  - 同构空间列数据表的创建语法如下：

```
CREATE TABLE [IF NOT EXISTS] <空间表名>
(
  [<属性列名>  <数据类型>[<列级完整性约束条件> ],]
  <几何列名>  geometry (<几何类型>,<SRID>) [ <完整性约束条件> ]
  [,<表级完整性约束条件>]
);
```

3.5.2 空间数据表管理

- 空间数据表的创建
  - 示例：

```
CREATE TABLE IF NOT EXISTS road (
  id          serial4              PRIMARY KEY,
  road_name   VARCHAR ( 50 )       NOT NULL,
  road_code   VARCHAR ( 10 )       NOT NULL,
  road_grade  INT                  DEFAULT 0,
  road_class  INT                  DEFAULT 0,
  geom        geometry( LineString, 3857 )   NOT NULL
);
```

3.5.2 空间数据表管理

空间数据表的修改

- 空间数据表的修改语法与关系型数据表的修改相同，语法如下：

```
ALTER TABLE <表名>
  [ ADD COLUMN [IF NOT EXITS] <新列名> <数据类型> [ 完整性约束 ] ]
  [ DROP COLUMN <列名>]
  [RENAME <列名> TO <新列名> ]
  [ ALTER COLUMN <列名> type <数据类型> ] ;
```

3.5.2 空间数据表管理

- 空间数据表的修改，示例:
  - 例3-8. 如果要给数据表road添加浮点类型的linktype字段，语句如下：

    ALTER TABLE road ADD COLUMN IF NOT EXISTS linktype FLOAT DEFAULT 0 NOT NULL;
  - 例3-9. 如果要将数据表road的linktype字段类型改为INT，语句如下：

    ALTER TABLE road ALTER COLUMN linktype TYPE FLOAT;
  - 例3-10. 如果要将数据表road的linktype字段名称改为link_type，语句如下：

    ALTER TABLE road RENAME linktype TO link_type;
  - 例3-11. 如果要删除数据表road的link_type字段，语句如下：

    ALTER TABLE road DROP COLUMN IF EXISTS linktype;
  - 例3-12. 如果要将数据表road的geom字段类型改为SRID=4326，语句如下：

    ALTER TABLE road ALTER COLUMN geom TYPE geometry(LineString,4326);

3.5.2 空间数据表管理

- 空间数据表的删除
  - 空间数据表的删除与普通数据表删除一样，语法如下：

    DROP TABLE [IF EXISTS] <数据表名> [CASCADE | RESTRICT];
  - 例3-13. 删除road表的语句如下：

    DROP TABLE **IF EXISTS** road;

3.5.2 空间数据表管理

- 空间数据视图的管理
  - **视图**是从一个或几个基本表（或视图）导出的表，它与基本表（实表）不同，是一个虚表，不存储数据。
    - 视图显示的数据随基本表中的数据变化而变化。
    - **空间数据视图**是至少包含一个空间字段的视图。
    - 视图的操作包括定义和删除等。
  - 定义视图的语法：

```
CREATE VIEW <视图名>
    AS <子查询>
        [WITH  CHECK  OPTION] ;
```

3.5.2 空间数据表管理

- 空间数据视图的管理
  - 创建视图示例：

```
CREATE TABLE view_road
AS
SELECT
        road_name,st_transform ( geom, 4326 ) AS geom
FROM
        road;
```

  - 删除视图语法如下：

```
DROP VIEW <视图名>;
```

    - 例3-15. 删除视图view_road的语句如下：

```
DROP VIEW view_road;
```

3.5.3 空间矢量数据的管理

- 空间几何图形的构建
  - OGC的SFA定义了几何数据构建函数
    - ST_Point
    - ST_MakePoint
    - ST_MakeLine
    - ST_Polygon
    - ST_MakePolygon
    - ST_Collect
    - ST_MakeEnvelope
    - ST_GeometryFromText
    - ST_GeomFromText
    - ST_PointFromText
    - ST_LineFromText
    - ST_PolygonFromText
    - ST_GeomFromEWKT
    - ST_GeomCollFromText

3.5.3 空间矢量数据的管理

- （1）ST_Point函数
  - ST_Point函数根据输入坐标值参数构建点几何数据，语法如下：

    geometry ST_Point(float x, float y);
  - 例3-16. 创建坐标为（112.926500,28.164390）的点几何数据，语句如下：

    SELECT ST_AsText(ST_Point(112.926500,28.164390));
- （2）ST_Point函数
  - ST_MakePoint函数为重载函数，用于构建2维、3维几何点，基本语法如下：

    geometry ST_MakePoint(float x, float y);
  - 例3-17. 创建坐标为（112.926500,28.164390）的点几何数据，语句如下：

    SELECT ST_AsText(ST_MakePoint(112.926500,28.164390));

3.5.3 空间矢量数据的管理

- （3）ST_MakeLine函数
  - ST_MakeLine函数为重载函数，使用点、多点构建线串，基本语法如下：

    ```
    geometry ST_MakeLine(geometry geom1, geometry geom2);
    ```

  - 例3-18. 创建一条从点(112.926500, 28.164390) 到点 (112.925310, 28.172060)的线串，语句如下：

    ```
    SELECT
            ST_AsText ( ST_MakeLine (
                    ST_MakePoint ( 112.926500, 28.164390 )
                    , ST_MakePoint ( 112.925310, 28.172060 )
            ) );
    ```

3.5.3 空间矢量数据的管理

- （4）ST_Polygon函数
  - ST_Polygon函根据输入的线串和SRID构建多边形几何数据，语法如下：

    geometry ST_Polygon(geometry lineString, integer srid);
  - 例3-20. 创建SRID=4326的多边形，语句如下：

```
SELECT
  ST_AsText(
    ST_Polygon('LINESTRING(0 0, 1 0, 1 1, 0 1,0 0)'::geometry, 4326)
  );
```

3.5.3 空间矢量数据的管理

- （5）ST_MakePolygon函数
  - ST_MakePolygon函数为重载函数，用于构建简单多边形或带孔（岛）的多边形，语法如下：

    ①geometry ST_MakePolygon(geometry linestring);

    ②geometry ST_MakePolygon(geometry outerlinestring, geometry[] interiorlinestrings);
  - 例3-21. 从一个线串创建一个简单多边形，语句如下：

```
SELECT
  ST_AsText(
    ST_MakePolygon('LINESTRING(0 0, 1 0, 1 1, 0 1,0 0)'::geometry)
  );
```

3.5.3 空间矢量数据的管理

- （6）ST_Collect函数
  - ST_Collect函数为重载函数，用于构建复合几何图形，语法如下：
    ① geometry ST_Collect(geometry g1, geometry g2);
    ② geometry ST_Collect(geometry[] g1_array);
    ③ geometry ST_Collect(geometry set g1field);
  - 例3-22. 使用两点创建复合点几何图形，语句如下：

```
SELECT
    ST_AsText( ST_Collect( 'POINT(1 2)','POINT(-2 3)' ));
```

3.5.3 空间矢量数据的管理

- （7）ST_MakeEnvelope函数
  - ST_MakeEnvelope函数用于构建矩形几何数据，语法如下：

```
geometry ST_MakeEnvelope(float xmin,
                         float ymin,
                         float xmax,
                         float ymax,
                         integer srid=unknown);
```

  - 例3-23. 创建矩形几何数据的语句如下：

```
SELECT
    ST_AsText( ST_MakeEnvelope(10, 10, 11, 11, 4326) );
```

3.5.3 空间矢量数据的管理

- （8）ST_GeometryFromText和ST_GeomFromText函数
  - 都是将WKT字符串转换为几何图形，语法如下：
    ① geometry ST_GeometryFromText(text WKT);
    ② geometry ST_GeometryFromText(text WKT, integer srid);
    ③ geometry ST_GeomFromText(text WKT);
    ④ geometry ST_GeomFromText(text WKT, integer srid);
  - 例3-24. 创建点几何图形，语句如下：

```
SELECT
   ST_AsText(ST_GeometryFromText('POINT(112.9265 28.16439)'));
```

  - 例3-26. 创建SRID=4326的环状面几何图形，语句如下：

```
SELECT
  ST_AsText(ST_GeometryFromText(
    'POLYGON((10 10, 10 11, 11 11, 11 10, 10 10),
                  (10.5 10.5,10.5 10.8,10.8 10.8,10.8 10.5,10.5 10.5))  '
    ,4326));
```

3.5.3 空间矢量数据的管理

- （9）ST_GeomFromEWKT函数。
  - ST_GeomFromEWKT函数将EWKT（ Extended Well-Known Text representation ）文本转换成几何图形。语法如下：

    geometry ST_GeomFromEWKT(text EWKT);
  - 例3-27. 利用EWKT文本生成SRID=4326的线，语句如下：

```
SELECT
    ST_AsText(ST_GeomFromEWKT('SRID=4326;LINESTRING(1 2, 3 4)));
```

3.5.3 空间矢量数据的管理

- 空间几何图形的编辑
  - OGC定义了常用的图形编辑函数
    - ST_AddPoint
    - ST_CollectionExtract
    - ST_LineMerge
    - ST_MakeValid
    - ST_Multi
    - ST_RemovePoint
    - ST_Reverse
    - ST_SetPoint
    - ST_Snap
    - 等

3.5.3 空间矢量数据的管理

（1）ST_AddPoint函数

- ST_AddPoint函数为重载函数，用于向已有线串几何图形中添加顶点，语法如下：
  - ① geometry ST_AddPoint(geometry linestring, geometry point);
  - ② geometry ST_AddPoint(geometry linestring, geometry point, integer position);
    - 函数①是向LineString起点处添加一个顶点；函数②的第3个参数为指定添加顶点的位次，顶点序号从0开始。如果设置为-1，表示向线串追加一个顶点。
- 例3-28. 向已有线串添加一个顶点，语句如下：

```
SELECT
 ST_AsText(
  ST_AddPoint(
   ST_LineFromText('LINESTRING(1 2, 3 4)'),ST_PointFromText('POINT(0 0)')));
```

3.5.3 空间矢量数据的管理

- （2）ST_CollectionExtract函数。
  - ST_CollectionExtract函数为重载函数，用于从几何集合中抽取子图形。语法如下：
    ① geometry ST_CollectionExtract(geometry collection);
    ② geometry ST_CollectionExtract(geometry collection, integer type);
    - 该函数根据给定的图形的类别type参数抽取相应子图形，type参数可以是1~3的整数，1表示POINT、2表示LINESTRING、3表示POLYGON。如果没有设置该参数，则返回图形集合中维度最高的几何图形。
  - 例3-30. 从图形集合中抽取线串，语句如下：

```
SELECT
 ST_AsText (
   ST_CollectionExtract (
        ST_GeomFromText (
          'GEOMETRYCOLLECTION(LINESTRING(0 0, 1 1),LINESTRING(2 2, 3 3))' ), 2 ) );
```

3.5.3 空间矢量数据的管理

- （3）ST_LineMerge函数
  - ST_LineMerge函数用于将多个线串合并成一条线串，语法如下：

    geometry ST_LineMerge(geometry amultilinestring);
  - 例3-31. 将多线串合并成一条线，语句如下：

```
SELECT
  ST_AsText(
    ST_LineMerge(
      ST_GeomFromText(
        'MULTILINESTRING((-29 -27,-30 -29.7,-36 -31,-45 -33),(-45 -33,-46 -32))'
)));
```

3.5.3 空间矢量数据的管理

- （4）ST_MakeValid函数
  - ST_MakeValid函数为重载函数，处理无效（自相交）图形，基本语法如下：
    - geometry ST_MakeValid(geometry input);
  - 例3-32. 处理一自相交多边形为有效多边形，语句如下：

```
SELECT
  ST_AsText(
    ST_MakeValid('POLYGON((0 0,10 0,10 11, 11 10,0 10,0 0))'::geometry));
```

3.5.3 空间矢量数据的管理

- （5）ST_Multi函数
  - ST_Multi函数将输入的几何图形转复合几何图形类型，语法如下：

    geometry ST_Multi(geometry g1);
  - 例3-33. 将一个多边形转为复合多边形，语句如下：

```
SELECT
  ST_AsText(
    ST_Multi(
      ST_GeometryFromText(
        'POLYGON((10 10, 10 11, 11 11, 11 10, 10 10))',4326)
));
```

3.5.3 空间矢量数据的管理

- （6）ST_RemovePoint函数
  - ST_RemovePoint函数从线串中移除一个顶点，语法如下：

    geometry ST_RemovePoint(geometry linestring, integer offset);
    - 参数offset是要移除点的位次，PostGIS从0开始计位次。
  - 例3-34. 从线串中移除第一个顶点，语句如下：

```
SELECT
  ST_AsText(
    ST_RemovePoint(
      ST_LineFromText('LINESTRING(0 0,1 2,3 4)'),0));
```

3.5.3 空间矢量数据的管理

- （7）ST_Reverse函数
  - ST_Reverse函数是将线串或多边形顶点置为倒序，语法如下：

    geometry ST_Reverse(geometry g1);
  - 例3-35. 将一个线串的顶点置为倒序，语句如下：

```
SELECT
  ST_AsText(ST_Reverse(ST_LineFromText('LINESTRING(0 0,1 2,3 4)')));
```

3.5.3 空间矢量数据的管理

- （8）ST_SetPoint函数
  - ST_SetPoint函数用于替换线串中指定位次的顶点，语法如下：

```
geometry ST_SetPoint(geometry linestring,
                     integer zerobasedposition,
                     geometry point);
```

    - 参数zerobasedposition表示要替换的顶点位次（从0开始），参数point是要替换的顶点。
  - 例3-36. 将线串的起点改为（0，0），语句如下：

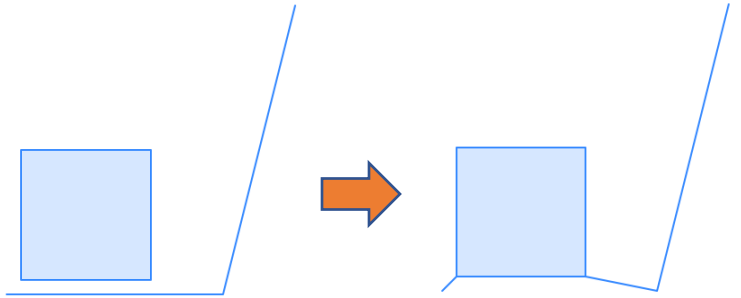
```
SELECT
  ST_AsText(ST_SetPoint(
     ST_LineFromText('LINESTRING(1 2,3 4)'),
     0,
     ST_PointFromText('POINT(0 0)')));
```

3.5.3 空间矢量数据的管理

- （9）ST_Snap函数
  - ST_Snap函数是将输入的几何图形向参考图形靠齐，即将输入的图形顶点向另一个参考图形的顶点靠齐。语法如下：

    geometry ST_Snap(geometry input, geometry reference, float tolerance);
    - 参数tolerance是捕捉容差距离，即将两个图形的顶点距离在该范围内时，才会将输入的图形顶点向参考图形顶点对齐。
  - 例3-37. 一条线串向一个多边形靠齐，如图3-8(a)所示。语句如下：

```
SELECT
  ST_AsText(
    ST_Snap(
      'linestring(0 0,15 0,20 20)'::GEOMETRY,
      'polygon((1 1,10 1,10 10,1 10,1 1))'::geometry,
      1.25));
```



3.5.4 空间矢量的数据访问与输出

- 空间几何图形的访问
  - PostGIS实现了
    - ST_Boundary
    - ST_Dump
    - ST_DumpPoints
    - ST_EndPoint
    - ST_Evelope
    - ST_GeometryN
    - ST_GeometryType
    - ST_IsClosed
    - ST_IsCollection
    - ST_IsEmpty
    - ST_IsSimple
    - ST_IsValid
    - ST_NumGeometries
    - ST_NumPoints
    - ST_X
    - ST_Y

3.5.3 空间矢量数据的管理

- （1）ST_Boundary函数
  - ST_Boundary函数返回图形的边界，语法如下：

    geometry ST_Boundary(geometry geomA);
  - 例3-38. 获取环状多边形的边界，语句如下：

```
SELECT
  ST_AsText(ST_GeometryFromText(
    'POLYGON((10 10, 10 11, 11 11, 11 10, 10 10),
     (10.5 10.5,10.5 10.8,10.8 10.8,10.8 10.5,10.5 10.5))',
    4326));
```

3.5.3 空间矢量数据的管理

- （2）ST_Dump函数
  - ST_Dump函数返回图形的所有组成子图形，语法如下：

    geometry_dump[] ST_Dump(geometry g1);
  - 例3-39. 显示复合图形的所有组成部件，语句如下：

```
SELECT
        ( A.p_geom ).PATH [ 1 ] AS PATH,
        ST_AsEWKT ( ( A.p_geom ).geom ) AS geom_ewkt
FROM
        ( SELECT ST_Dump ( ST_GeomFromEWKT ( 'POLYHEDRALSURFACE(
                                ((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)),
                                ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
                                ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
                                ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)),
                                ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)),
                                ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)))' )
         ) AS p_geom ) AS A;
```

3.5.3 空间矢量数据的管理

- （3）ST_DumpPoints函数
  - ST_DumpPoints函数返回图形的所有点，语法如下：

    geometry_dump[]ST_DumpPoints(geometry geom);
  - 例3-40. 获取一个线串的所有顶点，语句如下：

```
SELECT
        ( dp ).PATH [ 1 ] AS INDEX,
        ST_AsText ( ( dp ).geom ) AS wktnode
FROM
        ( SELECT *
          FROM ST_DumpPoints (
                  ST_GeomFromText ( 'LINESTRING(1 2, 3 4, 10 10)' ) )
         ) AS dp;
```

3.5.3 空间矢量数据的管理

- （4）ST_EndPoint函数
  - ST_EndPoint函数返回线串或圆线串（CircularLineString）的最后一个点，语法如下：

    geometry ST_EndPoint(geometry g);
  - 例3-41. 获取线串的最后一个点，语句如下：

    SELECT ST_AsText(ST_EndPoint('LINESTRING(1 1, 2 2, 3 3)'::geometry));

3.5.3 空间矢量数据的管理

- （5）ST_Evelope函数
  - ST_Evelope函数返回图形的外包矩形，语法如下：

    geometry ST_Envelope(geometry g1);
  - 例3-42. 返回线串的外矩形，语句如下：

    SELECT ST_AsText(ST_Envelope('LINESTRING(0 0, 1 3)'::geometry));
- （6）ST_GeometryN函数
  - ST_GeometryN函数返回图形集合中的第n个图元（序次从1开始），语法如下：

    geometry ST_GeometryN(geometry geomA, integer n);
  - 例3-43. 获取复合点集合的第2个点，语句如下：

```
SELECT
  ST_AsText (
    ST_GeometryN(
      ST_GeomFromEWKT ( 'MULTIPOINT(1 2 7, 3 4 7, 5 6 7, 8 9 10)' ), 2 ) );
```

3.5.3 空间矢量数据的管理

- （4）ST_EndPoint函数
  - ST_EndPoint函数返回线串或圆线串（CircularLineString）的最后一个点，语法如下：

    geometry ST_EndPoint(geometry g);
  - 例3-41. 获取线串的最后一个点，语句如下：

    SELECT ST_AsText(ST_EndPoint('LINESTRING(1 1, 2 2, 3 3)'::geometry));

3.5.3 空间矢量数据的管理

（5）ST_Evelope函数

ST_Evelope函数返回图形的外包矩形，语法如下：

```
geometry ST_Envelope(geometry g1);
```

例3-42. 返回线串的外矩形，语句如下：

```
SELECT ST_AsText(ST_Envelope('LINESTRING(0 0, 1 3)'::geometry));
```

返回结果：

```
st_astext
------------------------------
POLYGON((0 0,0 3,1 3,1 0,0 0))
```

3.5.3 空间矢量数据的管理

（6）ST_GeometryN函数

ST_GeometryN函数返回图形集合中的第n个图元（序次从1开始），语法如下：

```
geometry ST_GeometryN(geometry geomA, integer n);
```

例3-43. 获取复合点集合的第2个点，语句如下：

```
SELECT
  ST_AsText (
        ST_GeometryN(
        ST_GeomFromEWKT ( 'MULTIPOINT(1 2 7, 3 4 7, 5 6 7, 8 9 10)' ), 2 ) );
```

返回结果：

```
st_astext
-----------------------------
POINT Z (3 4 7)
```

3.5.3 空间矢量数据的管理

- （7）ST_GeometryType函数
  - ST_GeometryType函数返回国际标准“SQL多媒体及应用包-SQL/MM”定义的图形类型，包括ST_Point、ST_LineString、ST_Polygon、ST_MultiPolygon等等，语法如下：

    text ST_GeometryType(geometry g1);
  - 例3-44. 获取一个图形的类型，语句如下：

```
SELECT
  ST_GeometryType (
    ST_GeomFromText ( 'LINESTRING(1 2, 3 4, 10 10)' ) ) AS GeomType;
```

3.5.3 空间矢量数据的管理

- （8）ST_IsClosed函数
  - ST_IsClosed函数返回线串是否闭合，即首尾顶点是否一致，语法如下：

    boolean ST_IsClosed(geometry g);
  - 例3-45. 判断一个线串是否闭合，语句如下：

    SELECT ST_IsClosed('LINESTRING(0 0, 1 1)'::geometry);
- （9）ST_IsCollection函数
  - ST_IsCollection函数返回图形是否为集合类型，语法如下：

    boolean ST_IsCollection(geometry g);
  - 例3-46. 判断一个图形是否为集合类型，语句如下：

    SELECT ST_IsCollection('LINESTRING(0 0, 1 1)'::geometry);

3.5.3 空间矢量数据的管理

- （10）ST_IsEmpty函数
  - ST_IsEmpty函数判断图形是否为空，语法如下：

    boolean ST_IsEmpty(geometry geomA);
  - 例3-47. 判断一个图形是否为空，语句如下：

    SELECT ST_IsEmpty(ST_GeomFromText('GEOMETRYCOLLECTION EMPTY'));
- （11）ST_IsSimple函数
  - ST_IsSimple函数判断图形是否存在自相交，语法如下：

    boolean ST_IsSimple(geometry geomA);
  - 例3-48. 判断一个图形是否存在自相交，语句如下：

    SELECT ST_IsSimple(ST_GeomFromText('POLYGON((1 2, 3 4, 5 6, 1 2))'));

3.5.3 空间矢量数据的管理

- （12）ST_IsValid函数
  - ST_IsValid函数判为重载函数，用于判断图形是否有效，即是否存在自相交，功能与ST_IsSimple相似。语法如下：
    boolean ST_IsValid(geometry g);
  - 如果要获取更多关于图形无效的信息，可以使用ST_IsValidReason函数。
- （13）ST_NumGeometries函数
  - ST_NumGeometries函数返回图形集合中图形个数，语法如下：
    integer ST_NumGeometries(geometry geom);
  - 例3-48. 返回一个几何集合的子图个数，语句如下：

```
SELECT
  ST_NumGeometries(
    ST_GeomFromEWKT(
        'GEOMETRYCOLLECTION(MULTIPOINT(-2 3 , -2 2),
                            LINESTRING(5 5 ,10 10),
                            POLYGON((-7 4.2,-7.1 5,-7.1 4.3,-7 4.2)))')
) AS GeomN;
```

3.5.3 空间矢量数据的管理

- （16）ST_Points函数
  - ST_Points函数返回一个包括多个坐标值的图形（如线串、多边形）的所有顶点，返回类型为复合点（MultiPoint）。语法如下：

    geometry ST_Points(geometry geom);
  - 例3-51. 获取一个多边形所有顶点集，语句如下：

    SELECT ST_AsText(ST_Points('POLYGON((0 0,10 0,10 10,0 10,0 0))'::GEOMETRY));
- （17）ST_StartPoint函数
  - ST_StartPoint函数返回线串的第1个顶点，语法如下：

    geometry ST_StartPoint(geometry geomA);
  - 例3-52. 获取一个线串的起点，语句如下：

    SELECT ST_AsText(ST_StartPoint('LINESTRING(0 1, 0 2)'::geometry));

3.5.3 空间矢量数据的管理

- （18）ST_X函数
  - ST_X函数返回点的X坐标值，语法如下：

    float ST_X(geometry a_point);
  - 例3-53. 获取一个线串起点的X坐标值，语句如下：

    SELECT ST_X(ST_StartPoint(ST_GeomFromText('LINESTRING(0 0,1 1,2 2,3 3)')));
- （19）ST_Y函数
  - ST_Y函数返回点图形的Y坐标值，语法如下：

    float ST_X(geometry a_point);
  - 例3-54. 获取一个线串起点的Y坐标值，语句如下：

    SELECT ST_Y(ST_StartPoint(ST_GeomFromText('LINESTRING(0 0,1 1,2 2,3 3)')));

3.5.3 空间矢量数据的管理

- 空间几何图形数据的输出
  - PostGIS实现了
    - ST_AsText
    - ST_AsEWKT
    - ST_AsGeoJson
    - ST_AsKML
    - ST_AsGML
    - 等函数

3.5.3 空间矢量数据的管理

- （1）ST_AsText函数
  - ST_AsText函数为重载函数，返回空间图形数据的WKT文本，语法如下：
    ① text ST_AsText(geometry g1);
    ② text ST_AsText(geometry g1, integer maxdecimaldigits = 15);
    - 参数maxdecimaldigits用于设置输出的有效的小数位数，默认为15位。
  - 例3-55. 将一个点输出为WKT文本，要求保留2位有效小数位，语句如下：

```
SELECT ST_AsText('POINT(123456.123456 23123456.12345)'::geometry,2);
```

3.5.3 空间矢量数据的管理

- （2）ST_AsEWKT函数
  - ST_AsEWKT函数为重载函数，返回图形数据的EWKT文本，语法如下：
    ① text ST_AsEWKT(geometry g1);
    ② text ST_AsEWKT(geometry g1, integer maxdecimaldigits=15);
    - 参数maxdecimaldigits用于设置输出的有效的小数位数，默认为15位。
  - 例3-56. 将一个点输出为EWKT文本，要求保留2位有效小数位，语句如下：

```
SELECT ST_AsEWKT('SRID=3587;POINT(123456.123456 23123456.12345)'::geometry, 2);
```

中南大學
CENTRAL SOUTH UNIVERSITY

感谢您的观看!