全国高校地理信息科学实验教学系列丛书

空间数据库实验教程

程昌秀　张天媛　万幼　谷雷◎编著

清華大學出版社
北京

内 容 简 介

数据驱动是地理信息科学未来核心发展方向，空间数据库搭建与实操能力，也成为地理信息科学、测绘工程等专业从业者必备基础素养。本书沿 GIS 单机、C/S、B/S 至云端迭代脉络展开，融合蓝湖区域矢量地图、长江流域水系与地形、北京 POI 数据及出租车时空轨迹等真实地理数据集，设计多尺度、跨领域、贴合地理学科特色的完整实验体系。内容由浅入深，兼顾理论与实操，助力读者吃透空间数据库核心重难点，熟练掌握地理结构化查询语言(GSQL)编写、空间大数据分布式存储技术，以及生成式人工智能协同交互方法。本书适配地理科学、测绘工程等专业本科教学，亦可供行业技术人员查阅参考。

图书在版编目（CIP）数据

空间数据库实验教程 / 程昌秀等编著. -- 北京 ：清华大学出版社，2026. 8.
(全国高校地理信息科学实验教学系列丛书). -- ISBN 978-7-302-72641-8

Ⅰ. P208-33

中国国家版本馆 CIP 数据核字第 2026GK5114 号

责任编辑：秦 娜 王 华
封面设计：谢晓翠
责任校对：欧 洋
责任印制：宋 林

出版发行：清华大学出版社
网 址：https://www.tup.com.cn，https://www.wqxuetang.com
地 址：北京清华大学学研大厦 A 座 **邮 编**：100084
社 总 机：010-83470000 **邮 购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市天利华印刷装订有限公司
经 销：全国新华书店
开 本：185mm×260mm **印 张**：15 **字 数**：364 千字
版 次：2026 年 9 月第 1 版 **印 次**：2026 年 9 月第 1 次印刷
定 价：45.00 元

产品编号：116897-01 审图号：GS 京(2026)1703 号

全国高校地理信息科学实验教学系列丛书

编　委　会

丛书序

FOREWORD

值此“全国高校地理信息科学实验教学系列丛书”首批成果付梓之际，作为地理信息系统（GIS）教育领域从业者、本套丛书编委会顾问，我深感欣慰与振奋。

地理信息科学是支撑国土空间规划、智慧城市建设、时空大数据应用等国家战略领域的核心基础学科，其学科属性决定了实验教学在人才培养中的核心地位。实验教学是衔接课堂理论与行业实操的关键纽带，是学生将抽象原理转化为解决实际问题能力的必经之路，直接决定着地理信息专业人才的培养质量。数十年来，我国 GIS 教育事业蓬勃发展，专业布点持续增多，人才培养规模稳步提升，但在实践教学环节，始终存在教材体系碎片化、教学标准不统一、实操内容与产业需求衔接不紧密、国产 GIS 技术融入不足等共性痛点，成为专业人才培养质量进一步提升的瓶颈。

近年来，随着云计算、大数据、人工智能与三维可视化等技术的深度融入，GIS 产业正加速向地理空间智能方向演进；与此同时，信创战略全面推进，国产 GIS 平台已成为行业应用的主流选择，行业对人才的实践能力、技术适配性与创新素养提出了全新要求。新时代的 GIS 教育，急需一套体系完整、标准统一、贴合产业、适配国产技术的实验教学资源，为全国高校提供规范化的实践教学参照，推动 GIS 实践教学从“各自探索”走向“规范优质”。

正是基于这样的行业共识与教育使命，在 2024 年 12 月举办的首届 GIS 教育大会上，丛书编委会正式成立，全面启动本套丛书的编撰工作。在 2025 年 10 月举办的第二届 GIS 教育大会上，编委会完成与教材配套软件 SuperMap GIS 的资源深度对接，同步完善组织架构，聘任多位全国高校名师组织分册的编写，为丛书编撰筑牢了专业根基。历经一年多的系统筹备与深耕打磨，依托全国数十所高校的优势教学资源与行业头部企业的技术力量，首批教材终于与广大读者见面。这既是全国 GIS 教育界协同攻关的成果，也是产学研深度融合助力人才培养的有益探索，更标志着我国系统化的地理空间智能实践教学体系落地迈出了关键一步。

本套丛书首批推出《地理信息系统原理实验教程》《空间数据库实验教程》《GIS 空间分析实验教程》《时空大数据分析实验教程》《三维 GIS 实验教程》《遥感地学分析与应用实验教程》，全面覆盖 GIS 基础原理、数据管理、空间分析、智能挖掘、三维建模、遥感应用等核心教学模块，构建了从入门基础到前沿应用、从方法原理到工程实践的完整实验教学体系。丛书的编写体例与逻辑均基于以国产 GIS 软件为核心的实验平台，精选真实行业案例与科研数据，既注重核心原理的拆解阐释，又强化实操能力的系统训练，着力破解“会操作而不明机理”的教学难题，兼具教学适用性与产业适配性。

作为一套面向全国高校的实验教学丛书，其出版填补了国内地理信息科学领域标准化实验教学系列丛书的空白，能够为各高校 GIS 专业的实验教学、课程改革提供权威规范的

教学支撑，更将有力推动全国 GIS 实验教学的均衡化、高质量发展，助力构建产学研用一体化的人才培养体系，为我国地理信息产业持续输送高素质、应用型、创新型专业人才。

丛书编撰过程中，各分册作者凝心聚力、精益求精，开展了近百次线上线下研讨打磨，对实验内容、操作步骤、案例数据反复进行论证。编委会联动出版社严格落实出版规范，完成全流程三审三校，统一全套丛书的目录体系与装帧排版，全方位保障了丛书的专业性、规范性与系统性。

地理信息科学技术迭代迅速、应用边界持续拓展，实验教学资源也须与时俱进、动态更新。未来，编委会将紧密跟踪学科前沿发展与全国高校教学实践反馈，稳步推出更多方向的实验教程，同时对已出版教材进行常态化修订升级，保障丛书始终贴合学科发展节奏与教学实际需求。

教育事业薪火相传，教材建设永无止境。希望这套凝聚了全国 GIS 教育人心血的丛书，能够成为广大师生教学与学习过程中的得力助手，也期待广大读者与同行多提宝贵意见，共同推动丛书不断完善，为我国地理信息科学教育事业的发展添砖加瓦。

汤国安

2026 年夏于南京师范大学

前言

PREFACE

“空间数据库”是地理信息科学、测绘工程等专业人才培养体系中的核心基础课程，是构建地理空间数据管理能力、支撑 GIS 全链路技术应用的底层基石。随着数据驱动成为地理信息科学的核心发展方向，空间数据库的搭建、运维与应用能力，已成为相关专业学生与行业从业者必备的核心专业素养。课程兼具扎实的理论体系与极强的工程实践性，既要求学生掌握空间数据模型、查询优化等核心原理，又需要通过系统化实操训练，将理论转化为真实场景下的数据管理与分析能力，对学生的工程思维与实践能力提出了较高要求。

在数字经济与国土空间数字化建设浪潮下，地理信息科学正式进入以数据为核心的全新发展阶段。空间数据库作为地理信息系统的底层核心支撑，承担着自然资源调查、智慧城市建设、交通时空治理、流域生态监测等各类行业业务的数据存储、查询与智能分析职能，是数字孪生、实景三维、国土空间规划等重大工程不可或缺的技术基座。与此同时，伴随全国信创战略全面落地，国产化 GIS 软件替代进程持续提速，以 SuperMap GIS 系列软件为代表的国产地理信息平台已广泛应用于政务、自然资源、生态环境、交通等关键行业。产业变革倒逼人才培养体系同步转型，熟练掌握国产 GIS 平台空间数据库搭建、运维及时空大数据处理的实操能力，已成为相关专业人才培养的核心目标。

近年来，国内高校空间数据库课程建设持续推进，在理论体系完善、教学资源建设、教学方法创新等方面取得了一系列积极成果。但从实验教学层面看，仍存在与产业发展不相适配的现实瓶颈：一是市面上主流空间数据库实验教材大多基于国外 GIS 软件开发，无法适配现阶段国产化教学与产业应用需求；二是现有国产 GIS 相关图书多侧重三维开发与二次编程，缺少覆盖空间数据库全流程的分层实验设计，鲜有融入分布式存储、时空轨迹挖掘、生成式人工智能协同处理等前沿内容；三是教学中理论课堂与上机实操脱节问题突出，企业招聘时常反映应届生不熟悉国产空间数据库操作，人才供给与行业需求存在明显断层。在此时代与行业双重背景下，编写一套契合信创发展、依托国产 GIS 平台，体系完整的空间数据库实验教程，具备极强的现实必要性与应用价值。

本书由北京师范大学、武汉大学、河北师范大学、文山学院等多所高校 GIS 专业一线教师联合编著，依托团队多年国家一流本科课程教学积累、多项地理空间数据管理科研项目成果，以及全国高等学校 GIS 教学成果特等奖的实践经验编写而成，是《空间数据库管理系统概论》(第二版)的配套实验教材。编写团队秉持“技术演进为主线、国产平台为依托、真实场景为载体、能力进阶为目标”的建设思路，立足一线教学与产业实际需求，系统构建了梯度化的实验教学体系。

全书以 GIS 技术“单机—C/S—B/S—云”的演进架构为主线，依托 SuperMap GIS 等软件实验环境展开。书中选取蓝湖区域矢量地图、长江流域水系与地形、北京兴趣点(POI)数

据、海量出租车时空轨迹等多尺度、跨领域真实地理数据集，设计层次分明、贴合行业场景的系列实验。内容由浅入深，兼顾基础原理与前沿技术，读者可循序渐进地掌握 GSQL 空间查询编写、矢量与栅格高阶分析、远程数据库访问、Web 地图发布、海量时空数据分布式存储，以及空间数据与生成式人工智能交互等实操技能，真正打通理论学习与国产 GIS 平台工程实践的环节。

本书定位为地理科学、测绘工程等相关专业本科实验教学用书，同时可供自然资源、测绘、智慧城市领域技术人员开展技能学习与岗前培训。希望本书的出版，能够为地理信息科学及相关专业的空间数据库实验教学提供一套体系完备，贴合产业需求、技术前沿的实操参考，助力学生夯实空间数据管理理论基础，熟练掌握国产 GIS 平台工程应用技能，为后续专业学习与职业发展筑牢数据能力底座。本书编写过程中参考了诸多同行专家的学术成果与教学资料，清华大学出版社编辑团队为本书的打磨与出版提供了细致、专业的支持，在此一并表示衷心感谢！

受限于编者学识与研究范围，书中难免存在疏漏与不足之处，恳请各位师生、行业同仁批评指正，以便在后续修订中不断完善。

作者

2026 年 3 月

目录

CONTENTS

第1章 绪 论

1.1 空间数据库概述

空间数据库是以地球表面某一范围内的地理空间实体为对象，组织、存储和管理反映某一主题信息的数据集合。与传统数据库相比，空间数据库不仅保存地理实体的属性信息，还包含其空间位置、空间关系以及时空演变等信息，是地理信息系统（GIS）的核心支撑和基础。

空间数据库的应用非常广泛。例如，在土地利用和土地覆盖监测中，空间数据库可用于存储地块边界、用地类型及变化信息，实现动态监测与管理；在资源环境管理中，空间数据库可对水体分布、森林覆盖、土壤类型及污染源进行统一管理和分析；在城市规划与管理中，空间数据库可管理道路、建筑、绿地及公共设施数据，为科学决策提供依据；在交通运输领域，空间数据库可存储道路网、公交线路及实时交通流数据，用于导航、路径优化和支撑智能交通系统；在环境监测与灾害应急中，空间数据库可集成遥感影像、传感器数据及历史灾害记录，为风险分析和应急调度提供支持。

随着遥感、全球导航卫星系统（GNSS）、无人机及大数据技术的快速发展，空间数据呈现出数据量庞大、类型复杂、更新频繁的特点。这使得空间数据库的数据管理效率、查询响应速度以及跨部门和跨区域共享能力成为衡量信息化水平的重要指标。

空间数据库管理不仅依赖于数据库技术，还融合了地理信息科学、遥感技术、空间分析与可视化技术，属于交叉学科的重要组成部分。在本科 GIS 教学中，空间数据库的学习是理解空间数据管理、分析与可视化的基础，也是后续课程中进行空间建模、空间分析和决策支持的技术前提。

1.2 空间数据管理方式

1.2.1 文件系统管理方式

早期地理信息系统普遍采用文件方式存储空间数据，其中最具代表性的当属 Shapefile 格式。该方案将空间几何坐标存储于 .shp 文件，属性数据存放于 .dbf 文件，并通过 .shx 索引文件建立两者之间的关联。这种多文件分离存储的架构在技术发展初期以其实现简单、易于理解的特性得到广泛应用，但面对日益复杂的 GIS 应用需求，其局限性逐渐显现。

在安全性层面，文件系统缺乏细粒度的权限控制机制，文件易被篡改，且不具备用户认证与访问控制能力。用户一旦获得文件目录的访问权限，即可对整个数据集执行复制、修改甚至删除操作，难以实现读写权限的精细化管控。对于涉及敏感信息的地理数据而言，这一缺陷成为制约其行业应用的关键因素。

在数据一致性与完整性方面，由于空间几何与属性数据分别存储于不同文件中，任何涉及空间要素增删改的操作都需要同时维护两个文件，实践中操作不当或程序异常导致空间与属性数据脱节的现象时有发生，数据一致性差。文件系统无法定义主键、外键、值域约束等完整性规则，也难以约束多边形自相交、面与面重叠等空间拓扑关系，数据完整性不足，数据质量的控制完全依赖于应用程序编码。

并发控制能力的缺失进一步限制了文件方式的应用场景。文件系统不支持多用户同时读写同一数据集的锁机制，当一个用户编辑数据时，其他用户要么被迫等待，要么面临数据损坏风险，多用户操作易冲突。这一局限使得文件方式难以支撑需要多人协同工作的 GIS 项目，也无法适应现代 GIS 对高并发访问的需求。

在数据恢复与扩展性方面，文件方式同样面临挑战。文件系统缺乏事务日志机制，一旦文件因硬件故障、软件错误或人为误操作而损坏，往往只能依赖定期备份进行恢复，难以回到故障前的一致状态，数据恢复困难。单机文件系统无法有效利用分布式存储资源，当数据量突破单机处理能力时，性能急剧下降，难以满足大规模 GIS 应用的扩展需求。

1.2.2 数据库管理方式

空间数据库管理系统将空间数据与属性数据一体化存储于关系型数据库管理系统中，充分利用数据库管理系统(database management system，DBMS)内核技术构建空间数据的完整管理能力。PostgreSQL(简称 PgSQL)与 PostGIS 扩展的组合即是这一理念的典型实践，它将空间几何视为数据库原生数据类型，实现了空间与属性数据的统一存储与操作，从根本上解决了文件管理方式中数据分散、难以维护的问题。

在安全性方面，数据库方式依托成熟的用户/角色管理体系，支持用户认证、访问控制与数据加密机制，能够对访问权限进行精确到表、行、列的细粒度管理，有效防范未授权访问与数据泄露风险。

数据一致性与完整性的保障在数据库方式中得到了系统性实现。数据库事务管理确保操作的一致性，彻底消除了空间数据与属性数据脱节的问题。数据库支持的各类完整性约束，包括主键、外键、检查约束及空间关系验证，将数据质量控制从应用程序代码下沉至数据库内核，极大提升了数据的可靠性与可维护性。在并发控制方面，数据库管理系统内置的锁机制支持多用户同时访问，为高并发 GIS 业务场景提供了坚实保障。

在可靠性方面，数据库同样展现出文件管理无法比拟的优势。数据库提供备份、日志和恢复机制，管理员能够在系统故障或人为误操作后，将数据恢复至一致状态，有效降低数据丢失风险。在共享与互操作层面，基于结构化查询语言(structured query language，SQL)及地理结构化查询语言(geographic structured query language，GSQL)，不同应用程序可通过统一接口访问空间数据，支持开放地理空间联盟(open geospatial consortium，OGC)标准接口，具备跨平台、跨系统的数据共享能力。

1.2.3 两种管理方式对比

为直观呈现两种空间数据管理方式的核心差异，表 1-1 从数据存储方式、安全性、数据一致性、数据完整性、并发控制、数据恢复、数据共享、查询能力及适用场景九个维度进行了系统对比。

表 1-1 文件系统管理与数据库管理空间数据特性对比

特性	文件系统管理方式	数据库管理方式
数据存储方式	空间数据以文件(如 .shp)存储，属性数据用独立文件(如 .dbf)存储	空间与属性数据一体化存储在数据库中，如二进制大对象(binary large object，BLOB)类型
安全性	低，文件易被篡改，无权限控制	高，支持用户认证、访问控制、数据加密
数据一致性	差，空间与属性数据易脱节，更新不同步	强，事务管理确保操作一致性
数据完整性	不足，无法强制约束数据规则	强，支持完整性约束，如空间关系验证
并发控制	困难，多用户操作易冲突，无锁机制	高效，内置并发控制，支持多用户同时访问
数据恢复	困难，文件损坏后难修复	可靠，提供备份、日志和恢复机制
数据共享	难，封闭系统，互操作性差	易，支持标准接口，如 OGC/SQL-MM，互操作性强
查询能力	有限，依赖 GIS 软件特定功能	强大，支持扩展 SQL，如支持地理空间查询的 GSQL
适用场景	小规模、简单应用，如早期的 ArcView	中大规模、高要求应用，如智慧城市、实时分析

1.3 空间数据库体系结构

为实现对空间数据的高效管理、存储和分析，现代空间数据库通常采用分层体系结构。这种分层设计不仅有助于理解系统运行机制，也为不同实现方案提供了统一的技术框架。将系统功能划分为独立层次，每一层专注于特定职责，使得数据库具备良好的扩展性、可维护性和高性能处理能力。

物理存储层位于体系结构的最底层，负责空间数据在存储介质中的组织和管理。该层主要关注数据的存储格式、磁盘布局、数据块划分以及读写效率。空间数据通常体量巨大，尤其是遥感影像、数字高程模型(DEM)和多维栅格数据，数据量可以达到数百吉字节(GB)甚至太字节(TB)级别。因此，物理存储层不仅要保证数据安全和可靠性，还需要优化访问路径以提高读写性能。在实际中，物理存储层可以采用数据库存储结构。对于对象关系型数据库而言，空间数据通常被映射为数据库中的二进制大对象(BLOB)或专门的空间数据类型。其中，二进制大对象方式将完整的空间对象封装为二进制块进行存储，而专门的空间数据类型则允许数据库直接识别并处理点、线、面、栅格像元等几何结构。这些空间数据通过数据库存储引擎进行统一管理，从而充分利用数据库在事务、索引、并发控制和恢复机制等方面的成熟能力。

数据库管理层是空间数据库的核心，它为数据的存储、访问、事务管理和安全控制提供

支持。传统关系型数据库管理系统(relational database management system,RDBMS)通过表结构管理属性数据,但空间数据的非结构化特征、复杂拓扑关系和高维度属性要求对数据库内核进行扩展。数据库管理层通常引入空间数据类型、空间索引结构以及空间运算函数,从而实现对点、线、面(多边形)等空间对象的直接管理。例如,PostGIS 作为 PostgreSQL 的空间扩展,提供了一系列空间数据类型和函数,并支持创建索引以进行高效空间查询。类似地,Oracle Spatial 和 IBM DB2 Spatial Extender 通过扩展数据库内核,将空间对象与属性数据一体化管理。数据库管理层还负责事务处理、多用户并发控制、数据一致性维护以及故障恢复,保证空间数据库在复杂应用场景下的可靠性和稳定性。

空间服务层位于数据库管理层之上,负责为应用程序和用户提供标准化的数据访问接口与空间功能服务。通过这一层,桌面 GIS、Web GIS 以及行业应用系统可以方便地获取、查询、分析和处理空间数据。空间服务层可以实现对复杂空间操作的封装,例如空间叠加、缓冲区分析、最近邻搜索以及拓扑关系判断。在技术实现上,空间服务层通常提供 SQL 扩展(如空间结构化查询语言(SSQL))、应用程序编程接口(API)或者 Web 服务,使用户无须深入了解底层存储和索引结构即可进行空间分析。该层还可以进行数据格式转换和跨平台互操作,例如将 PostGIS 中的空间表直接载入 GIS 软件,或者通过空间数据引擎实现与不同数据库平台之间的数据交换。通过空间服务层,应用程序可以获得统一的访问方式,同时保证空间数据查询的效率和分析结果的准确性。

应用层面向最终用户,包括桌面 GIS 客户端、Web GIS 应用以及特定行业系统。用户通过这一层实现对空间数据库中数据的可视化、编辑、查询和分析。应用层不仅负责展示地理信息,还可以结合业务逻辑和专业算法提供决策支持。例如,在城市规划中,应用层可以调用数据库提供的空间函数,生成土地利用分析报告或交通网络优化方案;在环境监测中,应用层可进行污染扩散模拟或生态变化评估。应用层与空间服务层之间的交互通常通过标准接口进行,保障不同软件和平台的兼容性,实现数据共享。以 PostGIS + SuperMap iDesktopX 组合为例,SuperMap iDesktopX 作为应用客户端,可以直接连接 PostGIS 数据库,实现数据编辑、空间查询和地图制图功能;PostGIS 则在后台提供数据管理和高效查询服务,从而构成完整的融合式空间数据库解决方案。

整体而言,空间数据库的四层体系结构形成了自下而上的逻辑流程:物理存储层提供可靠的数据存储;数据库管理层保障数据安全性和完整性,同时实现空间数据处理;空间服务层提供标准接口和空间分析功能;应用层面向用户和业务需求,实现可视化和决策支持。这种分层结构的设计理念使得系统各层职责明确,便于独立升级和扩展,同时支持多种实现方式,如融合式方案(PostGIS+SuperMap iDesktopX)和寄生式方案。通过分层架构,空间数据库能够在处理大规模、高复杂度空间数据时保持高效性和可扩展性,并为 GIS 应用开发、跨平台数据共享以及行业定制化应用提供坚实的技术基础。

1.4 空间数据库应用方案

融合式与寄生式方案是两种典型的空间数据库技术方案,二者在技术基础、性能特征与适用场景上存在显著差异。寄生式方案依赖空间数据引擎(如 SuperMap SDX+)作为中间件,在通用关系型数据库之上添加独立的空间处理层,如 SuperMap iDesktopX 通过

SuperMap iObjects Java 提供的 API 实现空间数据的访问与操作。GIS 应用程序与中间件交互，由中间件负责空间数据的转换、存储与查询，空间数据以二进制大对象（BLOB 或 LOB）形式存放于数据库中，数据库仅管理属性数据。融合式方案则以数据库内核技术为核心，通过空间扩展 SQL（如 PostGIS）实现空间数据类型与函数的原生集成，将空间功能直接内置于数据库管理系统内部。GIS 应用程序通过扩展 SQL 直接访问数据库，空间与属性数据统一存储，数据库内核原生支持空间数据类型、索引与操作，实现高效的事务处理与空间查询。两类方案的体系结构差异如图 1-1 所示，它们各有侧重，共同构成了空间数据库技术体系的核心内容。

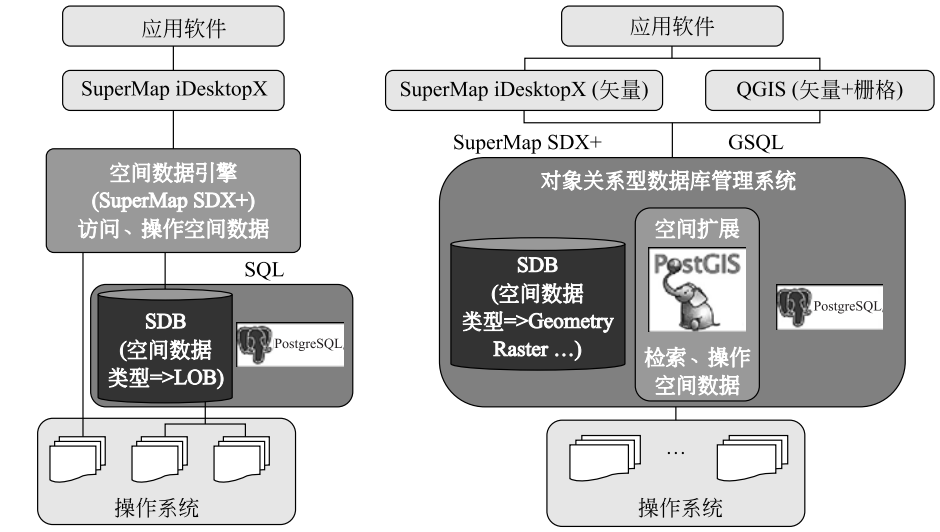


图 1-1 融合式与寄生式空间数据库体系结构对比

1.4.1 融合式应用方案

融合式空间数据库方案因结构清晰、部署简便而被广泛应用。该类方案以对象关系型数据库管理系统为基础，核心思想在于将空间数据管理能力原生集成到数据库中，使空间查询、分析与属性查询能够在统一的 SQL 环境下完成，从而充分利用数据库提供的事务管理、并发控制和安全机制。通过扩展空间数据类型、空间函数和空间索引，融合式方案实现了对矢量、栅格及多媒体空间数据的高效管理。相较于依赖中间件的寄生式空间数据引擎方案（图 1-1，寄生式方案通过空间数据引擎与 API 实现空间功能，空间数据以 SDB：LOB 形式存储），融合式方案技术路线清晰、实现成本低、操作简便。这里以 PostgreSQL＋PostGIS＋SuperMap iDesktopX 方案为例，介绍融合式空间数据库的具体构成与协同工作方式。

PostGIS 是对象关系型数据库系统 PostgreSQL 的空间扩展模块，由 Refractions Research 公司基于 GNU GPL 开源许可开发。PostGIS 在 PostgreSQL 上提供了一系列空间数据类型、空间函数以及运算符，使其能够支持标准的 OGC 空间数据模型。通过 PostGIS，用户可以在 SQL 中直接进行空间查询和分析，例如计算点与多边形的空间关系、

进行缓冲区分析或执行叠加运算，而无须额外的 GIS 软件进行数据中转。在实现上，PostGIS 将空间数据类型和空间函数使用 C 语言编写，并注册到数据库系统中，PostgreSQL 在启动时动态加载这些函数库，从而在数据库内核级别提供对空间数据的支持。同时，PostGIS 利用 R 树索引实现空间索引功能，该索引基于 PostgreSQL 中的广义搜索树（generalized search tree，GiST）框架，能够有效加速空间查询。PostGIS 的空间执行算法主要以元组级的嵌套循环连接实现，这在中小规模数据下性能良好。作为融合式方案的典型代表（图 1-1，融合式方案通过空间扩展 SQL 与 GSQL 实现原生空间数据类型支持），PostGIS 充分发挥了数据库内核技术的优势，利用 DBMS 的事务管理、并发控制与安全机制，支持扩展 SQL 实现标准化空间查询，并符合 OGC 标准，与 SuperMap iDesktopX 等专业 GIS 软件具有良好的互操作性。PostgreSQL 与 PostGIS 的组合是空间数据库体系中最成熟、最具代表性的方案之一。

SuperMap iDesktopX 作为专业桌面 GIS 软件，强调轻量化客户端与强大数据库后台的分工协作，与 PostGIS 形成了典型的融合式协同模式。在该模式中，数据库负责数据管理，桌面 GIS 软件负责应用与可视化。PostGIS 数据库承担数据存储、空间索引管理、空间函数计算和事务处理等核心任务，而 SuperMap iDesktopX 在可视化、交互与用户界面设计方面具有优势，同时具备更为复杂的空间分析能力。此外，SuperMap iDesktopX 作为跨平台桌面 GIS 平台，提供了丰富的数据处理与分析工具，支持海量空间数据的高效加载与显示，并兼容多种空间数据格式，能够与 PostGIS 数据库实现无缝对接，进一步拓展了融合式方案的应用场景。

融合式空间数据库方案在数据管理上将空间数据与属性数据统一存储于同一数据库中，保证数据的一致性与完整性，并可实现空间查询、分析和属性管理的整合操作。PostgreSQL 与 PostGIS 使系统具备良好的可扩展性，能够根据需求灵活扩展空间数据类型与函数。该方案部署简便、易于维护，适合中小规模数据集，能够完整呈现空间数据库的分层结构、空间数据类型及操作方法，为快速上手空间数据库技术提供了理想的实践平台。

1.4.2 寄生式应用方案

寄生式空间数据库方案以中间件技术为核心，通过在通用关系型数据库管理系统之上添加独立的空间处理层，实现空间数据与属性数据的分离存储与协同管理。如图 1-1 所示，该方案依赖空间数据引擎（如 SuperMap SDX+）作为中间件，通过 API 完成空间数据的转换、存储与查询，将空间操作请求解析后交由底层数据库处理或调用自研函数实现，再将结果封装返回至应用程序，空间数据以二进制大对象形式存储。相较于融合式方案，寄生式方案支持跨数据库平台，还能与特定 GIS 平台结合，在图层级空间处理效率方面具有优势，适合对数据库平台灵活性要求较高、需快速部署的应用场景。这里以 PostgreSQL + SuperMap iDesktopX 方案为例，介绍寄生式空间数据库的具体构成与协同工作方式。

在寄生式方案中，PostgreSQL 作为通用关系型数据库管理系统，负责存储属性数据及以 SDB：LOB 形式存放的空间数据。SuperMap iDesktopX 通过 SuperMap iObjects Java 提供的 API 调用内置的空间数据引擎（如 SuperMap SDX+），承担空间处理层的核心职能，

将用户在 GIS 客户端执行的空间操作(如缓冲区分析、叠加查询)转换为数据库可执行的 SQL 语句,并以二进制大对象格式完成空间几何数据在数据库中的写入与读取。此模式无须在数据库端安装任何空间扩展,空间计算主要依赖中间件引擎完成,数据库仅作为数据存储载体。这一设计使空间分析任务的执行依托中间件与 GIS 平台的深度优化,在特定空间操作场景下可获得较高的处理效率,同时保持了底层数据库平台的灵活性。

寄生式方案将空间处理能力封装于独立的中间件层,通过空间数据引擎实现空间操作指令与 SQL 语句的转换,为需要跨数据库平台部署、对特定空间操作效率要求较高的 GIS 应用提供了灵活的技术路径。该方案与专业 GIS 平台结合紧密,能够在图层级空间分析、实时渲染等场景中发挥性能优势,但其无法深度利用数据库的内核事务管理以及 SQL 扩展能力,数据共享与互操作能力相对弱于融合式方案。本书第 3 章部分内容(3.1.2 节、3.1.4 节、3.3.3 节、3.4 节)将基于 PostgreSQL+SuperMap iDesktopX 寄生式方案构建实验案例,帮助读者体会其在跨平台部署、特定空间操作优化及与 GIS 平台深度集成等方面的典型特征。

1.4.3 两种应用方案对比

融合式方案与寄生式方案在技术路径与应用定位上各具特点。融合式方案以数据库内核为核心,通过空间数据库扩展(如 PostGIS)实现空间数据类型与函数的原生集成,将空间功能直接内置于数据库管理系统,充分利用事务管理、并发控制与索引优化等机制,在数据一致性、查询效率及跨平台共享能力方面表现优异,适合大规模、高可靠性的现代 GIS 应用场景。寄生式方案则依赖空间数据引擎(如 SuperMap SDX+)作为中间件,在通用关系型数据库之上添加独立的空间处理层,通过 API 封装空间数据访问与操作,空间数据以二进制大对象形式存储,支持跨数据库平台部署,与特定 GIS 平台结合紧密,在图层级空间处理效率上具有优势,但难以利用数据库内核的完整能力,数据共享与互操作性相对较弱。两类方案的选择需综合考虑应用规模、性能需求与系统集成度,表 1-2 从技术基础、代表产品、主要优缺点及适用场景等维度对二者进行了系统对比。

表 1-2 寄生式与融合式空间数据库应用方案对比

特性	寄生式方案	融合式方案
技术基础	中间件技术,在通用 RDBMS 上添加空间处理层	数据库内核技术,空间功能直接集成到 DBMS 中
代表产品	ArcSDE、SuperMap SDX+、MapGIS SDE、TerraLib	Oracle Spatial、PostGIS、DB2 Spatial Extender
主要优点	支持跨数据库平台,灵活性高;与特定 GIS 平台结合紧密,图层级操作等空间处理效率高	充分利用 DBMS 内核技术,存取效率高;支持扩展 SQL,如 GSQL 或 SSQL;符合 OGC 标准,易于数据共享与互操作
主要缺点	难以利用事务管理等 DBMS 内核技术,难以支持扩展 SQL,数据共享与互操作性差	图层级空间处理性能相对较差,需额外优化;对特定 GIS 平台的依赖性较低,可能需定制开发
适用场景	需高效处理实时分析等特定空间操作,且对数据库平台灵活性要求高的场景	需高度数据一致性、强共享能力(如智慧城市平台),且强调标准化查询的大规模应用

本章对空间数据库的基本概念、体系结构及主要实现方案进行了概述。空间数据库不仅能够存储地理实体的属性信息，还能管理其空间位置、空间关系及时空演变特征，是 GIS 的核心支撑。相较于传统文件管理方式，空间数据库借助数据库管理系统成熟的内核技术，在数据一致性、并发控制、安全性、恢复机制及可扩展性等方面实现了质的飞跃。为了实现高效的数据管理与分析，现代空间数据库通常采用分层体系结构，从物理存储、数据库管理、空间服务到应用层逐层构建功能模块，各层职责明确、可独立扩展，能够满足大规模、高并发的 GIS 应用需求。

在具体实现上，空间数据库形成了融合式与寄生式两种典型方案。融合式方案以对象关系型数据库管理系统为基础，通过空间数据库扩展将空间数据类型、索引与操作原生集成至数据库内核，实现空间与属性数据的一体化存储，具有事务一致性强、标准化查询能力突出、易于跨平台共享的特点，适合大规模、高可靠性的现代 GIS 应用。寄生式方案则依赖空间数据引擎作为中间件，在通用关系型数据库之上添加独立的空间处理层，空间数据以二进制大对象形式存储，由中间件完成空间操作转换与封装，支持跨数据库平台部署，在图层级空间处理效率上具有优势，适用于需快速部署、对数据库平台灵活性要求较高的场景。

通过本章内容，读者可以掌握空间数据库的概念、分层体系结构以及融合式与寄生式两类典型实现方案的特点和应用场景，为后续空间数据管理、空间分析与 GIS 应用开发奠定理论基础。

第2章

蓝湖地区地图数据库的构建与查询

本章以蓝湖地区地图数据为例，介绍从地图到空间数据库几何对象模型的建模过程，并给出从建库到查询的 GSQL 代码。请读者完成附录 A～附录 C 中 PostgreSQL＋PostGIS＋SuperMap iDesktopX 的环境安装后开展本实验。注意每条 GSQL 语句执行时末尾需带分号。请读者体会空间数据库从设计、建设到查询的全过程，理解基于扩展数据类型的空间数据库解决方案，熟悉 PostgreSQL＋PostGIS 的相关概念和操作。

本章实验的物理实现和空间查询均在 PostgreSQL 和 PostGIS 的支持下实现。读者可以在 pgAdmin 中开展下述实验。请读者先用 pgAdmin 在 postgres 数据库中建立一个名为“bluelake”的模式，用于存放蓝湖地区的数据。后续蓝湖地区的数据表均存放于此模式中。

2.1 蓝湖地区地图数据库实验数据

图 2-1 是虚构的蓝湖地区整饰地图，其背景及包含信息如下：

(1) 图 2-1 所示区域是通用横轴墨卡托(universal transverse Mercator，UTM)投影坐标下的一个矩形区域。水平坐标是 POSC 水平坐标系统 32214 号。注意 WGS72/UTM 14 带东伪偏移值为 500000，单位为 m。

(2) “鹅岛”所在的“蓝湖”是该地区的重要因素。

(3) 区域内有一条从北到南的河道。注入蓝湖的河道叫“卡姆河”，从蓝湖流出的河道没有名字。

(4) 地图范围内有名为“阿诗顿”的区域。

(5) 森林管理区涵盖了蓝湖和阿诗顿部分区域。图中单车道的路、双车道宽的 5 号路和 4 车道宽的 5 号路以及地图边界围成了森林边界。其中，标记了树符号的“绿森林”等于森林范围减去蓝湖的范围。

(6) 5 号路贯穿并延伸出地图，双车道宽的路用粗黑线表示，4 车道宽的路用粗灰线表示。

(7) 75 号高速路用两条粗的黑线表示，即两条被隔离的双向通行的高速路，这两条线通常视为多线。

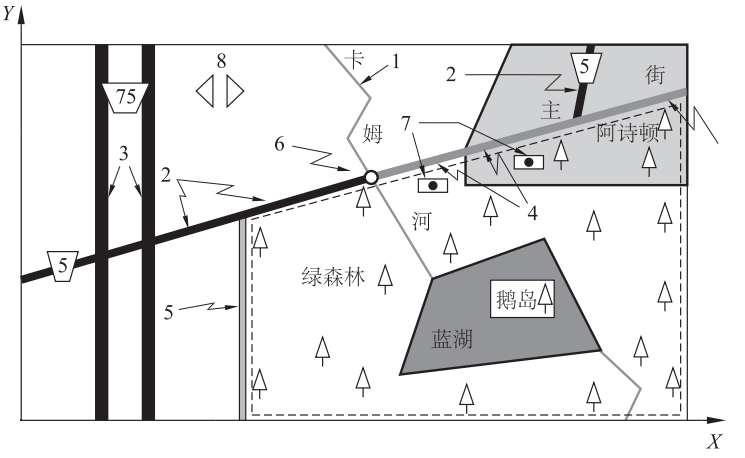
(8) 跨越卡姆河的桥叫“卡姆桥”，通常被视为点对象。

(9) 与 5 号路共享一段道路的主街是 4 车道宽。

(10) 沿着主街有两个建筑物，它们可以视为点，也可视为矩形区域。

(11) 单车道的路形成了森林边界的一部分，如两边带黑边的灰线所示。

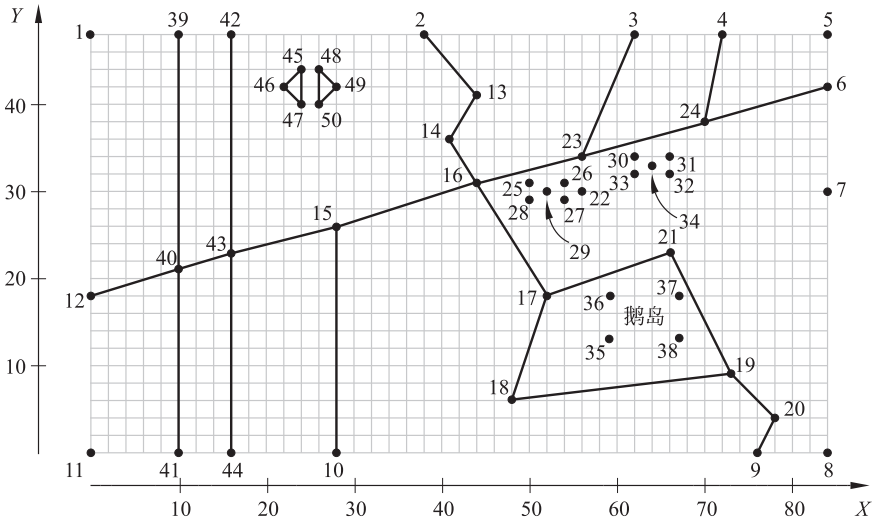
(12) 图 2-1 中有两个鱼池，并非独立存在，而是共为一体，视为多多边形。



X—东；Y—北；1—河道；2—5 号路；3—75 号高速路；4—主街；5—单车道的路；6—卡姆桥；7—建筑物；8—鱼池；双车道宽的 5 号路；4 车道宽的 5 号路。

图 2-1 蓝湖区域地图

图 2-1 中描述的点及其坐标如图 2-2 所示。



X—东；Y—北。

图 2-2 蓝湖地区数据集的点及其坐标(单位：m)

2.2 概念设计与逻辑设计

将蓝湖地区的空间对象抽象为桥、河流、湖泊、岛、路段、组合路、建筑物、池塘、区域九类空间实体。由于它们同时具有空间和非空间属性，故在后续的内容中也被称为“要素”。空间实体的属性及联系如图 2-3 所示。其中，桥被定义为点要素；路段、河流被定义为线要

素；考虑到组合路是由两条被隔离的双向高速路组成，故其被定义为多线要素；建筑物可以用点或多边形表示，故其缩略图为备选形状；池塘则为多多边形要素；尽管图中湖泊、岛、区域等要素尚未出现多多边形的情形，但考虑到后续新增要素可能会出现多多边形，故我们也将其定义为多多边形要素。上述实体各自属性如图 2-3 中与其相连的椭圆所示，其中双线椭圆的属性为主属性。

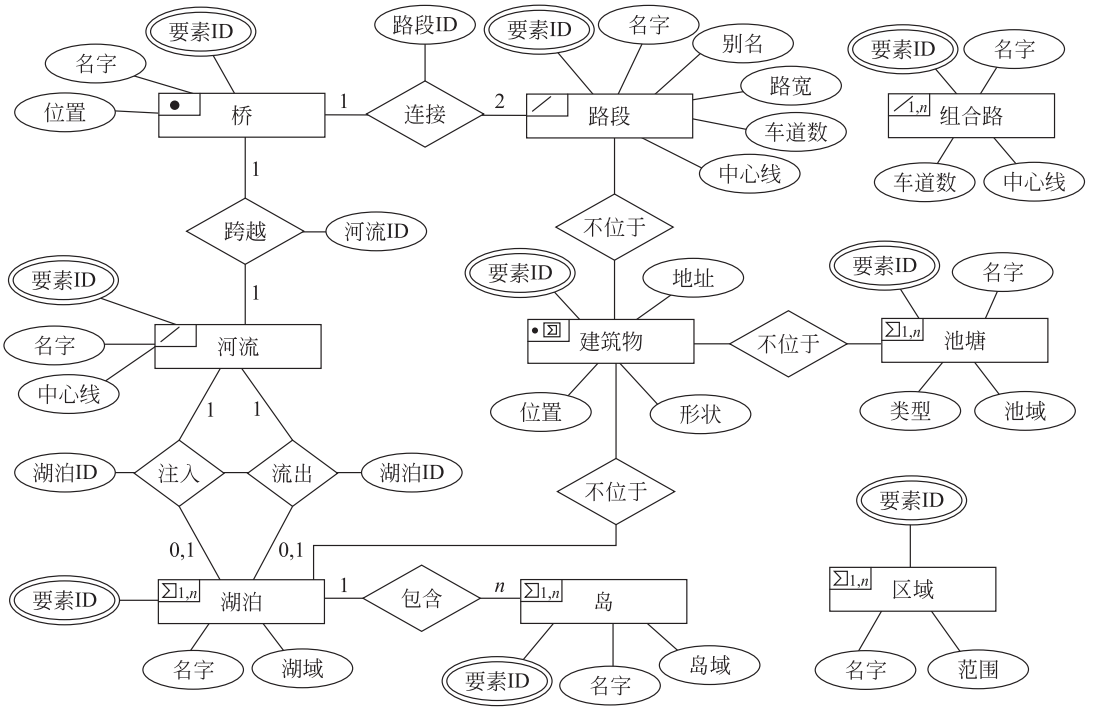


图 2-3　蓝湖地区的扩展 E-R 图

重点介绍实体之间的关系。考虑到同一道路的不同路段可能拥有不同的别名和车道数，将路段作为道路的最小建模单元。此外，图 2-3 中的 1 座桥连接 2 条路段，1 座桥跨越 1 条河流，1 条河流可能注入 0～1 个湖泊或从 0～1 个湖泊流出，建筑物不能位于湖泊或池塘中，也不能位于路段上。上述联系仅用于展示要素的行为规则和相关概念，不一定适用于其他空间应用。在数据库的实际设计中，读者应结合应用中的实际情况设计实体及联系。

根据图 2-3 中的扩展 E-R 图，可将每个要素都转换为带有几何类型的二维要素表，如表 2-1～表 2-9 所示。

表 2-1　湖泊(lakes)

字段(中)	字段(英)	字段类型	主键	外键	备　　注
要素 ID	fid	Integer	是		
名字	name	Char(64)			
湖域	shore	Geometry			其几何类型限定为 MultiPolygon

表 2-2 路段(road_segments)

字段(中)	字段(英)	字段类型	主键	外键	备 注
要素 ID	fid	Integer	是		
名字	name	Char(64)			
别名	aliases	Char(64)			
路宽	rwidth	Float			
车道数	num_lanes	Integer			
中心线	centerline	Geometry			其几何类型限定为 LineString

表 2-3 组合路(divided_route)

字段(中)	字段(英)	字段类型	主键	外键	备 注
要素 ID	fid	Integer	是		
名字	name	Char(64)			
车道数	num_lanes	Integer			
中心线	centerline	Geometry			其几何类型限定为 MultiLineString

表 2-4 桥(bridges)

字段(中)	字段(英)	字段类型	主键	外键	备 注
要素 ID	fid	Integer	是		
名字	name	Char(64)			
河流 ID	streamid	Integer		是	
路段 1ID	roadseg1id	Integer		是	空位 0
路段 2ID	roadseg2id	Integer		是	
位置	position	Geometry			其几何类型限定为 Point

表 2-5 河流(streams)

字段(中)	字段(英)	字段类型	主键	外键	备 注
要素 ID	fid	Integer	是		
名字	name	Char(64)			
流出湖泊 ID	fromlakeid	Integer			
注入湖泊 ID	tolakeid	Integer			
中心线	centerline	Geometry			其几何类型限定为 LineString

表 2-6 建筑物(buildings)

字段(中)	字段(英)	字段类型	主键	外键	备 注
要素 ID	fid	Integer	是		
地址	address	Char(64)			
位置	position	Geometry			其几何类型限定为 Point
形状	footprint	Geometry			其几何类型限定为 Polygon

表 2-7　池塘(pond)

字段(中)	字段(英)	字段类型	主键	外键	备　　注
要素 ID	fid	Integer	是		
名字	name	Char(64)			
类型	type	Char(64)			
池域	shores	Geometry			其几何类型限定为 MultiPolygon

表 2-8　岛(island)

字段(中)	字段(英)	字段类型	主键	外键	备　　注
要素 ID	fid	Integer	是		
岛名	name	Char(64)			
所在湖泊 ID	lakeid	Integer			
边界	boundary	Geometry			其几何类型限定为 MultiPolygon

表 2-9　区域(zone)

字段(中)	字段(英)	字段类型	主键	外键	备　　注
要素 ID	fid	Integer	是		
名字	name	Char(64)			其几何类型限定为 MultiPolygon
边界	boundary	Geometry			

需要注意如下几点：

(1) 如果空间数据库的几何类型足够丰富，要素表中的空间属性应定义为相应的几何类型。例如，“湖泊”表中的“湖域”字段应定义为 MultiPolygon 类型，“桥”表中的“位置”字段应定义为 Point 类型。但目前多数空间数据库管理系统不支持如此丰富的类型，而仅用 Geometry 类型代替所有的几何类型，故表中将空间属性的类型定义为 Geometry；不同要素表中几何类型的约束，由用户在应用层进行限制。

(2) 对于具有备选形状的建筑物，为其建立“位置”和“形状”两个 Geometry 字段，位置用于存放点状形式的建筑物，而形状用于存放多边形形式的建筑物。空间数据库中一个表可以存储多个几何字段，这也是空间数据库有别于文件地理数据库的特色所在。

(3) 对于图 2-3 中的“连接”“跨越”“注入”“流出”和“包含”关系，在数据库设计中可以对“桥”表中的“路段 1ID”和“路段 2ID”，“桥”表中的“河流 ID”，“河流”表中的“注入湖泊 ID”“流出湖泊 ID”，“岛”表中的“所在湖泊 ID”字段进行限制，但在系统实现中空间关系还需通过触发器或程序代码进一步约束。对于图 2-3 中的“不位于”关系，则只能通过触发器或开发者在应用程序中实现。

(4) 考虑到数据库管理系统对中文支持不足，建议用表 2-1～表 2-9 中的英文表名、字段名进行建库。

2.3 蓝湖地区地图数据库物理实现

2.3.1 建表前的准备工作

建表前，用下述 GSQL 语句将 WGS72/UTM 14 带的空间参考 WKT 标识插入 SPATIAL_REF_SYS 系统表中。

```
/* 空间参考系*/
INSERT INTO SPATIAL_REF_SYS VALUES(101, 'POSC', 32214, 'PROJCS["UTM_ZONE_14N",
GEOGCS["World Geodetic System 72",DATUM["WGS_72", ELLIPSOID["NWL_10D", 6378135,
298.26]],PRIMEM["Greenwich", 0], UNIT["Meter", 1.0]],PROJECTION["Transverse_
Mercator"],PARAMETER["False_Easting", 500000.0],PARAMETER["False_Northing",
0.0],PARAMETER["Central_Meridian", - 99.0],PARAMETER["Scale_Factor", 0.9996],
PARAMETER["Latitude_of_origin", 0.0],UNIT["Meter",1.0]]');
```

2.3.2 建表的 GSQL 语句

根据表 2-1～表 2-9 的描述，使用下述语句在空间数据库中建立用户定义的要素表。右击数据库，选择“Query Tool”，通过撰写 SQL 语句创建一个新模式 bluelake。注意，在未指定模式的情况下，默认将表建在 public 模式中。

```
/* 创建 bluelake 模式*/
CREATE SCHEMA bluelake;

/* 湖泊*/
CREATE TABLE bluelake.lakes (
fid INTEGER NOT NULL PRIMARY KEY,
name VARCHAR(64),
shore GEOMETRY('MULTIPOLYGON',101)
);

/* 路段*/
CREATE TABLE bluelake.road_segments (
  fid INTEGER NOT NULL PRIMARY KEY,
  name VARCHAR(64),
  aliases VARCHAR(64),
  num_lanes INTEGER,
  centerline GEOMETRY('LINESTRING',101)
);

/* 组合路*/
CREATE TABLE bluelake.divided_routes (
fid INTEGER NOT NULL PRIMARY KEY,
```

```
name VARCHAR(64),
num_lanes INTEGER,
centerlines GEOMETRY('MULTILINESTRING',101)
);

/* 桥*/
CREATE TABLE bluelake.bridges (
  fid INTEGER NOT NULL PRIMARY KEY,
  name VARCHAR(64),
  roadseg1id INTEGER REFERENCES bluelake.road_segments,
  roadseg2id INTEGER REFERENCES bluelake.road_segments,
  position GEOMETRY('POINT',101)
);

/* 河流*/
CREATE TABLE bluelake.streams (
  fid INTEGER NOT NULL PRIMARY KEY,
  name VARCHAR(64),
  fromlakeid INTEGER REFERENCES bluelake.lakes,
  tolakeid INTEGER REFERENCES bluelake.lakes,
  centerline GEOMETRY('LINESTRING',101)
);

/* 建筑物*/
CREATE TABLE bluelake.buildings (
  fid INTEGER NOT NULL PRIMARY KEY,
  address VARCHAR(64),
  position GEOMETRY('POINT',101),
  footprint GEOMETRY('POLYGON',101)
);

/* 池塘*/
CREATE TABLE bluelake.ponds (
  fid INTEGER NOT NULL PRIMARY KEY,
  name VARCHAR(64),
  type VARCHAR(64),
  shores GEOMETRY('MULTIPOLYGON',101)
);

/* 岛*/
CREATE TABLE bluelake.island (
  fid INTEGER NOT NULL PRIMARY KEY,
  name VARCHAR(64),
  lakeid INTEGER REFERENCES bluelake.lakes,
  boundary GEOMETRY('MULTIPOLYGON',101)
);
```

```
69.
70. /* 区域*/
71. CREATE TABLE bluelake.zone (
72.   fid INTEGER NOT NULL PRIMARY KEY,
73.   name VARCHAR(64),
74.   boundary GEOMETRY('MULTIPOLYGON',101));
```

2.3.3 建表后的系统表信息

建完上述 9 个要素表后，空间数据库的 SPATIAL_REF_SYS 表和 GEOMETRY_COLUMNS 视图的表头及其记录如表 2-10 和图 2-4 所示。

表 2-10 SPATIAL_REF_SYS 表的表头及相关记录

SRID	AUTH_NAME	AUTH_SRID	SRTEXT
…	…	…	…
101	"POSC"	32214	"PROJCS[""UTM_ZONE_14N"", GEOGCS[""World Geodetic System 72"", DATUM[""WGS_72"", ELLIPSOID[""NWL_10D"", 6378135, 298.26]], PRIMEM[""Greenwich"", 0], UNIT[""Meter"", 1.0]], PROJECTION[""Transverse_Mercator""], PARAMETER[""False_Easting"", 500000.0], PARAMETER[""False_Northing"", 0.0], PARAMETER[""Central_Meridian"", −99.0], PARAMETER[""Scale_Factor"", 0.9996], PARAMETER[""Latitude_of_origin"", 0.0], UNIT[""Meter"",1.0]]"
…	…	…	…

f_table_catalog character varying (256)	f_table_schema name	f_table_name name	f_geometry_column name	coord_dimension integer	srid integer	type character varying (30)
postgis_33_sample	bluelake	lakes	shore	2	101	MULTIPOLYGON
postgis_33_sample	bluelake	road_segments	centerline	2	101	LINESTRING
postgis_33_sample	bluelake	divided_routes	centerlines	2	101	MULTILINESTRING
postgis_33_sample	bluelake	bridges	position	2	101	POINT
postgis_33_sample	bluelake	streams	centerline	2	101	LINESTRING
postgis_33_sample	bluelake	buildings	position	2	101	POINT
postgis_33_sample	bluelake	buildings	footprint	2	101	POLYGON
postgis_33_sample	bluelake	ponds	shores	2	101	MULTIPOLYGON
postgis_33_sample	bluelake	island	boundary	2	101	MULTIPOLYGON
postgis_33_sample	bluelake	zone	boundary	2	101	MULTIPOLYGON

图 2-4 GEOMETRY_COLUMNS 视图的表头及相关记录

2.3.4 空间数据的插入

利用如下 GSQL 语句，将图 2-2 所示的空间数据分别插入上述 9 个用户定义的要素表中。

```
/* 插入数据*/
/* 湖泊*/
INSERT INTO bluelake.lakes VALUES (101, '蓝湖',
ST_GeomFromText('MULTIPOLYGON (((52 18, 66 23, 73 9, 48 6, 52 18),
(59 18, 67 18, 67 13, 59 13, 59 18)))', 101));
INSERT INTO bluelake.lakes VALUES (0, '图外其他湖泊');

/* 路段*/
INSERT INTO bluelake.road_segments VALUES(102, '路 5', NULL, 2,
ST_GeomFromText('LINESTRING(0 18, 10 21, 16 23, 28 26, 44 31)' ,101));
INSERT INTO bluelake.road_segments VALUES(103, '路 5', '主街', 4,
ST_GeomFromText('LINESTRING(44 31, 56 34, 70 38)' ,101));
INSERT INTO bluelake.road_segments VALUES(104, '路 5', NULL, 2,
ST_GeomFromText('LINESTRING(70 38, 72 48)' ,101));
INSERT INTO bluelake.road_segments VALUES(105, '主街', NULL, 4,
ST_GeomFromText('LINESTRING(70 38, 84 42)' ,101));
INSERT INTO bluelake.road_segments VALUES(106, '绿森林边路', NULL, 1,
ST_GeomFromText('LINESTRING(28 26, 28 0)',101));

/* 组合路*/
INSERT INTO bluelake.divided_routes VALUES(119, '路 75', 4,
ST_GeomFromText('MULTILINESTRING((10 48,10 21,10 0),
(16 0,16 23,16 48))', 101));

/* 桥*/
INSERT INTO bluelake.bridges VALUES(110, '卡姆桥', 102, 103, ST_GeomFromText
    ('POINT(44 31)', 101));

/* 河流*/
INSERT INTO bluelake.streams VALUES(111, '卡姆河', 0, 101,
ST_GeomFromText('LINESTRING(38 48, 44 41, 41 36, 44 31, 52 18)', 101));
INSERT INTO bluelake.streams VALUES(112, NULL, 101, 0,
ST_GeomFromText('LINESTRING(76 0, 78 4, 73 9)', 101));

/* 建筑物*/
INSERT INTO bluelake.buildings VALUES(113, '主街 123 号',
ST_GeomFromText('POINT(52 30)', 101),
ST_GeomFromText('POLYGON((50 31, 54 31, 54 29, 50 29, 50 31))', 101));
INSERT INTO bluelake.buildings VALUES(114, '主街 215 号',
ST_GeomFromText('POINT(64 33)', 101),
ST_GeomFromText('POLYGON((66 34, 62 34, 62 32, 66 32, 66 34))', 101));

/* 池塘*/
INSERT INTO bluelake.ponds VALUES(120, NULL, '思道哥池塘',
```

```
44. ST_GeomFromText('MULTIPOLYGON(((24 44, 22 42, 24 40, 24 44)),
45. ((26 44, 26 40, 28 42, 26 44)))', 101));
46.
47. /* 岛*/
48. INSERT INTO bluelake.island VALUES(109, '鹅岛',101,
49. ST_GeomFromText('MULTIPOLYGON (((67 13, 67 18, 59 18, 59 13, 67 13)))', 101));
50.
51. /* 区域*/
52. INSERT INTO bluelake.zone VALUES(117, '阿诗顿',
53. ST_GeomFromText('MULTIPOLYGON (((62 48, 84 48, 84 30, 56 30, 56 34, 62 48)))',
    101));
54. INSERT INTO bluelake.zone VALUES(118, '绿森林',
55. ST_GeomFromText('MULTIPOLYGON(((28 26, 28 0, 84 0, 84 42, 28 26),
56. (52 18, 66 23, 73 9, 48 6, 52 18)), ((59 18, 67 18, 67 13, 59 13, 59 18)))', 101));
```

2.3.5 蓝湖数据的查看

在 SuperMap iDesktopX 中加载空间数据库中的数据，合理设置叠放顺序后，空间数据情况如图 2-5 所示。

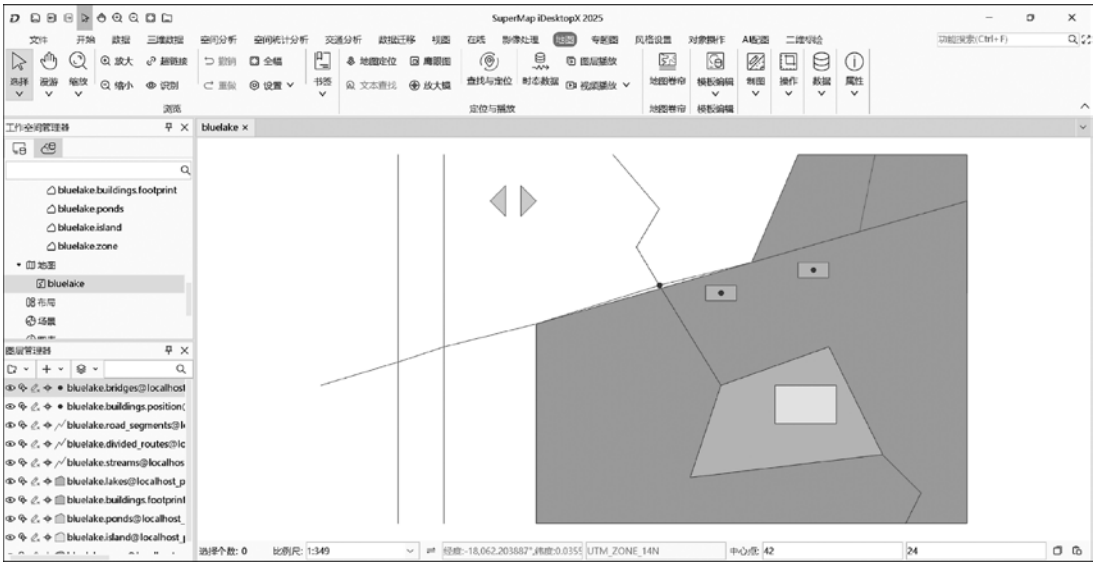


图 2-5 蓝湖地区的空间数据

2.4 蓝湖地区地图空间查询

2.4.1 单表空间查询

基于插入的空间数据，可进行单表空间查询操作。表 2-11 给出了一些常见的单表查询和连接查询的 GSQL 语句及其查询结果。

表 2-11 单表空间查询示例

序号	查询描述	GSQL 语句	查询结果
1	查找本数据库中所有的空间表	SELECT F_TABLE_NAME FROM GEOMETRY_COLUMNS;	lakes, road_segments, divided_routes, buildings, bridges, island, streams, ponds, zone
2	查找 streams 表的空间参考系授权	SELECT S. auth_srid FROM SPATIAL_REF_SYS S, GEOMETRY_COLUMNS G WHERE G. F_TABLE_SCHEMA = 'bluelake' AND G. F_TABLE_NAME='lakes' AND S. SRID=G. SRID;	32214
3	获得 75 号路的几何类型	SELECT ST_GeometryType(centerlines) FROM bluelake. divided_routes WHERE name='路 75';	'ST_MULTILINESTRING'
4	获得鹅岛的 WKT 表示	SELECT ST_AsText(boundary) FROM bluelake. island WHERE name='鹅岛';	'MULTIPOLYGON(((67 13,67 18,59 18,59 13,67 13)))'
5	判断名为“路 5”别名为“主街”的路段的几何属性是否为空	SELECT ST_IsEmpty(centerline) FROM bluelake. road_segments WHERE name='路 5' AND aliases='主街';	false
6	蓝湖的几何结构是否为简单	SELECT ST_IsSimple(shore) FROM bluelake. lakes WHERE name='蓝湖';	true
7	获得鹅岛的边界	SELECT ST_AsText(ST_Boundary(boundary)) FROM bluelake. island WHERE name='鹅岛';	'MULTILINESTRING (67 13,67 18,59 18,59 13,67 13)'
8	获得鹅岛的 MBR 边界	SELECT ST_AsText(ST_Envelope(boundary)) FROM bluelake. island WHERE name='鹅岛';	'POLYGON((59 13,59 18,67 13,67 13,59 13))'
9	获得卡姆桥的 x, y 坐标	SELECT ST_X(position), ST_Y(position) FROM bluelake. bridges WHERE name='卡姆桥';	44.00,31.00
10	获得 102 路段的起点和终点	SELECT ST_AsText(ST_StartPoint(centerline)), ST_AsText(ST_EndPoint(centerline)) FROM bluelake. road_segments WHERE fid=102;	'POINT(0 18)','POINT(44 31)'
11	判断鹅岛的 MBR 边界是否闭合	SELECT ST_IsClosed(ST_Boundary(boundary)) FROM bluelake. island WHERE name='鹅岛';	true
12	获得 106 路段的长度	SELECT ST_Length(centerline) FROM bluelake. road_segments WHERE fid=106;	26.00(m)
13	获得 102 路段中点的数目	SELECT ST_NumPoints(centerline) FROM bluelake. road_segments WHERE fid=102;	5
14	获得 102 路段中点第一个点	SELECT ST_AsText(ST_PointN(centerline,1)) FROM bluelake. road_segments WHERE fid=102;	'POINT(0 18)'
15	获得鹅岛的质心	SELECT ST_AsText(ST_Centroid(boundary)) FROM bluelake. island WHERE name='鹅岛';	POINT(63 15.5)

续表

序号	查询描述	GSQL 语句	查询结果
16	判断 PointOnSurface 函数返回鹅岛上的点是否在其边界内	SELECT ST_Contains(boundary,ST_PointOnSurface(boundary)) FROM bluelake. island WHERE name='鹅岛';	True
17	获得鹅岛的面积	SELECT ST_Area(boundary) FROM bluelake. island WHERE name='鹅岛';	40.00(m^2)
18	获得蓝湖内环的数目	SELECT ST_NumInteriorRing(shore) FROM bluelake. lakes WHERE name='蓝湖';	null
19	判断 75 号路的几何元素的数目	SELECT ST_NumGeometries(centerlines) FROM bluelake. divided_routes WHERE name='路 75';	2
20	获得 75 号路的第 2 个几何元素	SELECT ST_AsText(ST_GeometryN(centerlines,2)) FROM bluelake. divided_routes WHERE name='路 75';	LINESTRING(16 0,16 23,16 48)
21	获得 75 号路的长度	SELECT ST_Length(centerlines) FROM bluelake. divided_routes WHERE name='路 75';	96.00(m^2)
22	判断鹅岛是否与 WKT 表达构造的多边形相等	SELECT ST_Equals(boundary,ST_PolyFromText('POLYGON((67 13,67 18,59 18,59 13,67 13))',101)) FROM bluelake. island WHERE name='鹅岛';	true
23	判断 75 号路是否与阿诗顿相离	SELECT ST_Disjoint(centerlines,boundary) FROM bluelake. divided_routes d,bluelake. zone z WHERE d. name='路 75' AND z. name='阿诗顿';	true
24	卡姆河是否与蓝湖相接	SELECT ST_Touches(s. centerline, l. shore) FROM bluelake. streams s, bluelake. lakes l WHERE s. name='卡姆河' AND l. name='蓝湖';	true
25	判断主街 215 号的建筑物是否在阿诗顿内	SELECT ST_Within(b. footprint, z. boundary) FROM bluelake. buildings b, bluelake. zone z WHERE b. address='主街 215 号' AND z. name='阿诗顿';	false
26	判断 102 路段是否跨越 75 号路	SELECT ST_Crosses(r. centerline,d. centerlines) FROM bluelake. road_segments r, bluelake. divided_routes d WHERE r. fid=102 AND d. name='路 75';	true
27	判断鹅岛与阿诗顿是否符合"TTTTTTTTTT"的关系	SELECT ST_Relate(z. boundary, i. boundary, 'TTTTTTTTT') FROM bluelake. zone z, bluelake. island i WHERE z. name='阿诗顿' AND i. name='鹅岛';	false
28	获得卡姆桥与阿诗顿的距离	SELECT ST_Distance(position,boundary) FROM bluelake. bridges b, bluelake. zone z WHERE b. name='卡姆桥' AND z. name='阿诗顿';	12(m)

续表

序号	查询描述	GSQL 语句	查询结果
29	获得卡姆河与蓝湖相交部分的几何结构	SELECT ST_AsText(ST_Intersection(s.centerline,l.shore)) FROM bluelake.streams s,bluelake.lakes l WHERE s.name＝'卡姆河' AND l.name＝'蓝湖';	'POINT(52 18)'
30	获得蓝湖和鹅岛并集的几何结构	SELECT ST_AsText(ST_Union(l.shore,i.boundary)) FROM bluelake.lakes l,bluelake.island i WHERE l.name＝'蓝湖' AND i.name＝'鹅岛';	'POLYGON((66 23,73 9,48 6,52 18,66 23))'
31	获得蓝湖和鹅岛对称差集的几何结构	SELECT ST_AsText(ST_SymDifference(l.shore,i.boundary)) FROM bluelake.lakes l,bluelake.island i WHERE l.name＝'蓝湖' AND i.name＝'鹅岛';	'POLYGON((66 23,73 9,48 6,52 18,66 23))'
32	获得蓝湖的凸包	SELECT ST_AsText(ST_ConvexHull(shore)) FROM bluelake.lakes WHERE name＝'蓝湖';	'POLYGON((48 6,52 18,66 23,73 9,48 6))'

2.4.2　多表空间查询

在某些复杂的应用中,还可进行嵌套查询和集合查询操作。下面给出一些复杂空间查询的语句示例,前两个示例为嵌套查询,后一个示例为集合查询。

```
1. /* 找出距离 Stream 中河流中心线最近的一个区域*/
2. SELECT Z1.name, S1.Name
3. FROM bluelake.zone Z1, bluelake.streams S1
4. WHERE ST_Distance(Z1.Boundary, S1.Centerline) <  ALL (SELECT ST_Distance(Z2.
   Boundary, S1.Centerline)
5. FROM bluelake.zone Z2 WHERE Z1.Name<> Z2.name)
6.
7. /* 找出中心线长度超过"主街"的所有路段*/
8. SELECT RS1, name
9. FROM bluelake.road_segments RS1
10. WHERE ST_Length(RS1.Centerline)> ANY (SELECT ST_Length(RS2.Centerline)
11.          FROM bluelake.road_segments RS2
12.          WHERE  RS2.Name<> '主街')
13.
14. /* 列出数据库中所有池塘和湖泊的名字和形状*/
15. SELECT name, shores
16. FROM bluelake.ponds
17. UNION
18. SELECT name, shore
19. FROM bluelake.lakes
```

思考题

1. 查询地址为“主街 215 号”的建筑物 footprint 的几何类型，并判断该几何是否为空。
2. 判断“卡姆河”是否与“蓝湖”相离(不相接)。
3. 获取“路 75”的第 1 个几何元素，并计算该几何元素的长度。

第3章

长江流域数据库的空间查询与远程访问

本章以长江流域数据为例，学习高阶的 GSQL 查询，掌握在局域网环境下访问远程数据库服务器的方法，以及实现长江流域空间数据在 Web 环境下的共享与发布。开展实验前，请读者完成附录 A～附录 F 中 PostgreSQL＋PostGIS＋SuperMap iDesktopX＋QGIS＋SuperMap iServer＋SuperMap iPortal 等环境的安装与配置，保障后续实验顺利推进。


实验数据

3.1 长江流域数据库实验数据

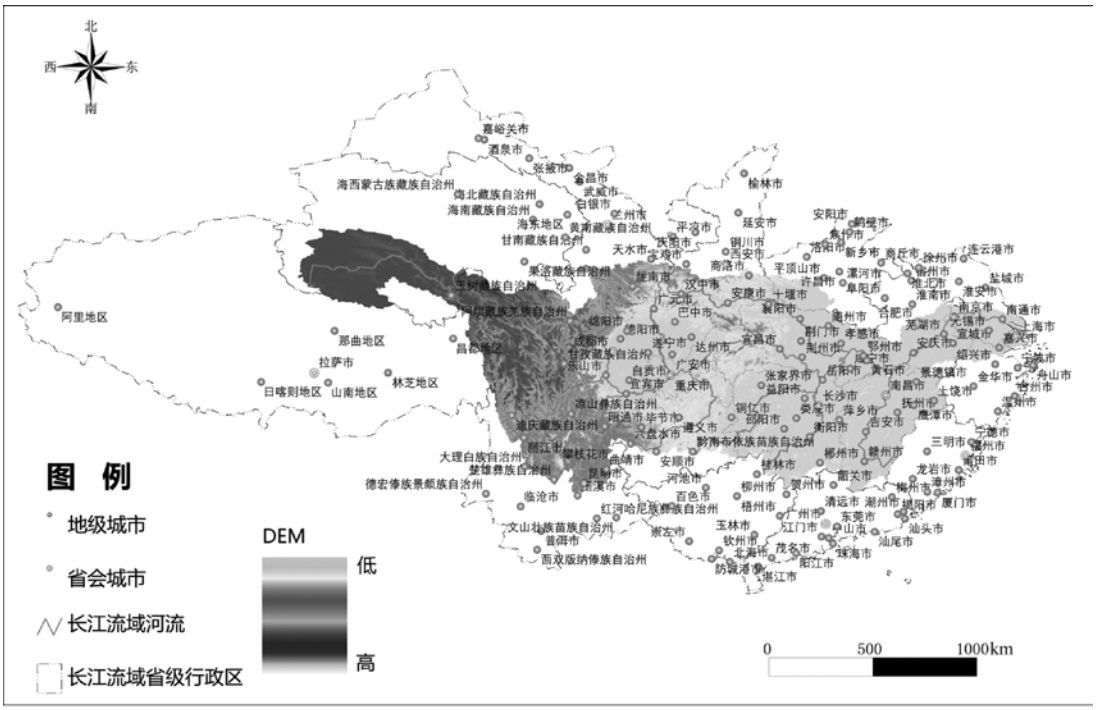
长江是亚洲和中国的第一大河，世界第三大河。发源于青藏高原唐古拉山脉各拉丹冬峰西南侧，干流自西向东流经青海、西藏、四川、云南、重庆、湖北、湖南、江西、安徽、江苏、上海 11 个省、自治区、直辖市，最终于上海市崇明岛附近注入东海。长江水系发达，支流众多，主要支流有雅砻江、岷江、沱江、嘉陵江、乌江、湘江、汉江、赣江等，构成了庞大的长江流域水系网络。

本节以长江流域省级行政区面状矢量数据（province. shp）、长江干流及支流线状矢量数据（river. shp）、长江流域地市级行政单位点状矢量数据（city. shp）以及长江流域 1km 分辨率数字高程模型（DEM）栅格数据（yangtze_dem. tif）为例，介绍 PostGIS、QGIS、SuperMap iDesktopX 的空间数据导入导出功能，上述空间数据和属性数据如图 3-1 所示，其中，city 表中的 province 字段参照 province 表中的 Name 字段，实现了市级与省级行政单位数据的关联匹配。数据的空间参考系均为 WGS84，对应的空间参考系 ID 为 4326。

3.1.1 融合式体系结构的矢量数据导入导出

1. 使用 PostGIS 的工具导入导出

为了将长江流域的数据放在同一个模式中，请读者先用 pgAdmin 在 postgis_36_sample 空间数据库中建立一个名为“yangtzeriver”的模式（图 3-2），后续长江流域空间数据表均导入此模式中。



彩图 3-1

city@DataSource ×

序号	*SmID	SmUserID	*SmX	*SmY	*SmGeom...	province	name	capital	GDP	population
1	1	0	110.353329	21.274276	BinaryData	广东省	湛江市	N	3,100.22	703.09
2	2	0	109.116673	21.485222	BinaryData	广西壮族自治区	北海市	N	1,276.91	185.32
3	3	0	108.335657	21.615472	BinaryData	广西壮族自治区	防城港市	N	732.81	104.61
4	4	0	110.921104	21.66634	BinaryData	广东省	茂名市	N	3,252.34	617.41
5	5	0	111.97798	21.86064	BinaryData	广东省	阳江市	N	1,360.44	260.29

province@DataSource ×

序号	*SmID	SmUserID	*SmArea	*SmPerim...	*SmGeoPa...	*SmGeom...	Name	code	GDP	population	area
1	1	0	236,492,323,...	4,820,807.41...	BinaryData	BinaryData	广西壮族自治区	450000	22,156.69	5,012.68	284,041.5619...
2	2	0	177,773,933,...	6,409,226.91...	BinaryData	BinaryData	广东省	440000	110,760.94	12,601.25	211,939.7329...
3	3	0	383,305,222,...	6,581,325.12...	BinaryData	BinaryData	云南省	530000	24,521.9	4,720.93	468,998.8459...
4	4	0	176,042,434,...	4,060,908.91...	BinaryData	BinaryData	贵州省	520000	17,826.56	3,856.21	221,996.3846...
5	5	0	122,289,751,...	5,197,766.10...	BinaryData	BinaryData	福建省	350000	43,903.89	4,154.01	152,260.2034...

river@DataSource ×

序号	*SmID	SmUserID	*SmLength	*SmTopoE...	*SmGeoPa...	*SmGeom...	NAME	mainstream	Shape_Leng
1	1	0	951,666.1300...	0	BinaryData	BinaryData	通天河	Y	951.666161
2	2	0	338,829.7817...	0	BinaryData	BinaryData	沱沱河	Y	338.829795
3	3	0	2,807,536.69...	0	BinaryData	BinaryData	长江	Y	2,807.53674
4	4	0	1,916,420.43...	0	BinaryData	BinaryData	金沙江	Y	1,916.42047
5	5	0	642,131.8994...	0	BinaryData	BinaryData	岷江	N	642.13191

图 3-1　长江流域空间数据和属性数据表

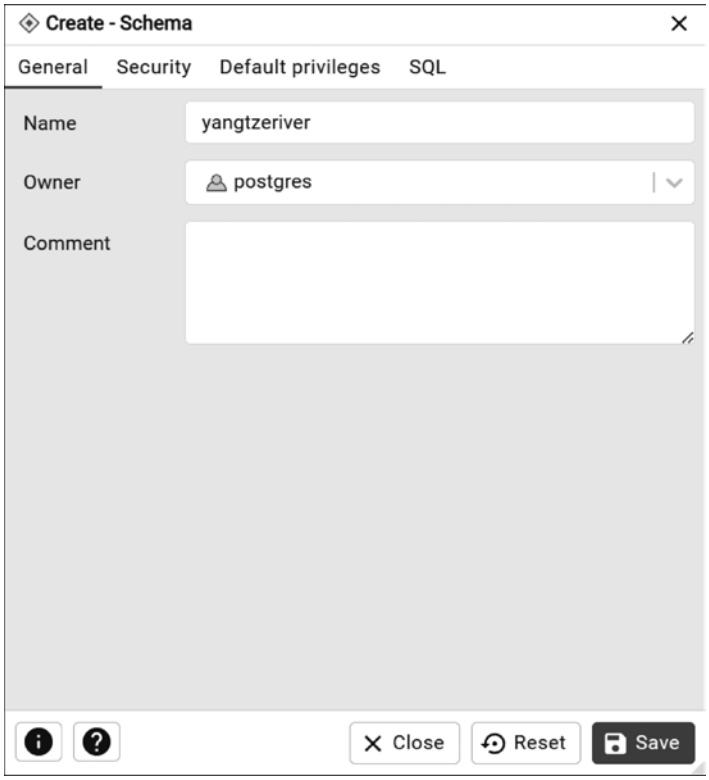


图 3-2　创建模式 yangtzeriver

PostGIS 作为 PostgreSQL 的空间扩展模块，自带 Shapefile 格式数据的导入导出工具，可直接实现矢量数据的数据库入库，具体步骤如下。

1）启动工具并建立数据库连接

在 Windows 系统中，找到 PostGIS 安装目录下的【PostGIS Bundle 3 for PostgreSQL x64 18】菜单并单击，启动【PostGIS Shapefile Import/Export Manager】工具主界面，如图 3-3 所示。单击界面中的【View connection details... 】，弹出数据库连接参数配置对话框，依次填写用户名（postgres）、密码、服务器地址（localhost）、端口（5432）、目标数据库（postgis_36_sample），填写完成后单击【OK】，如图 3-4 所示。若参数配置正确，工具主界面的 Log Window 区域会提示连接成功，完成数据库连接的建立，如图 3-5 所示。

2）选择 Shapefile 文件、设置参数并执行入库

在工具主界面单击【Add File】，在文件选择对话框中选中待导入的长江流域矢量文件：city. shp、province. shp、river. shp，如图 3-6 所示；返回主界面后，在导入文件列表中，为3 个文件统一设置空间参考系 SRID 为 4326，指定数据入库的目标模式为 yangtzeriver，对应数据表名分别设为 city、province、river，如图 3-7 所示。所有参数设置完成后，单击【Import】执行数据导入，若导入成功，Log Window 区域会显示“Shapefile import completed”的相关提示，完成矢量数据的 PostGIS 入库，如图 3-8 所示。

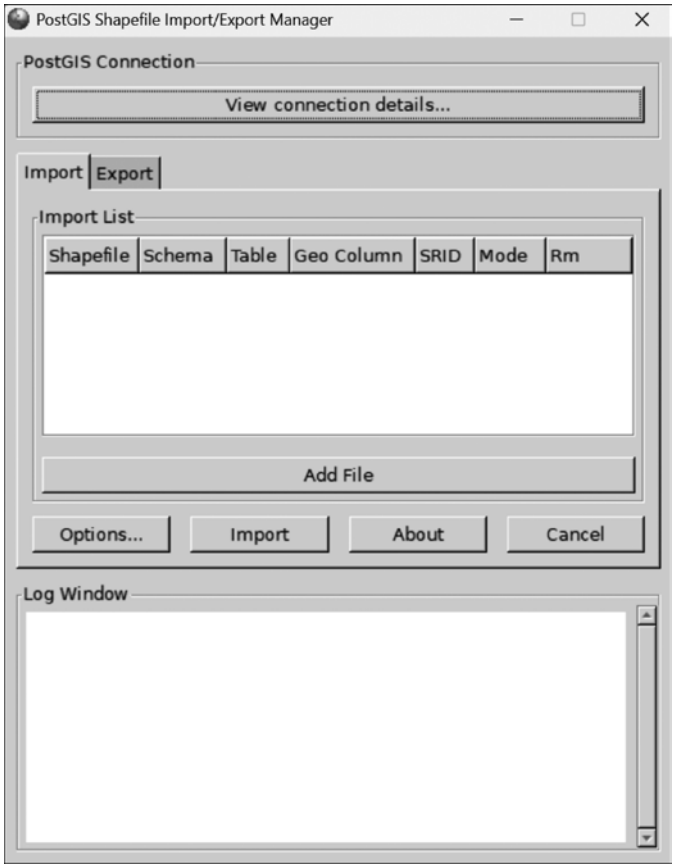


图 3-3　启动【PostGIS Shapefile Import/Export Manager】工具主界面

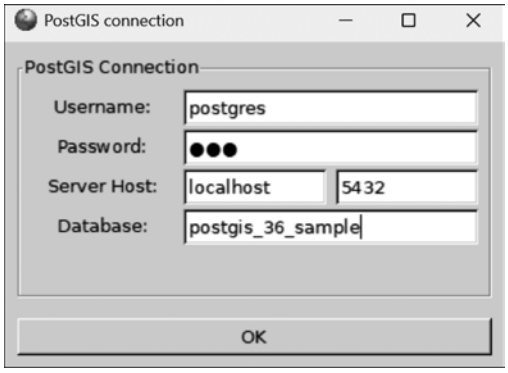


图 3-4　填写数据库连接参数配置对话框

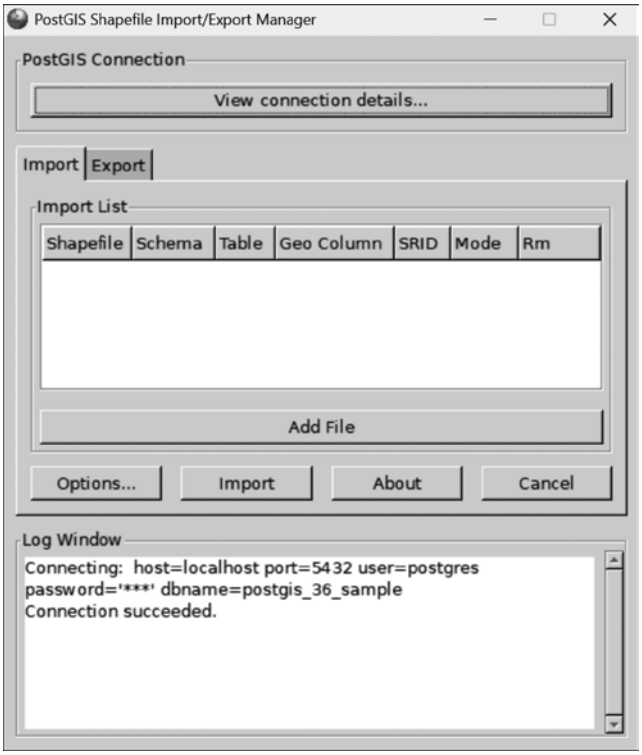


图 3-5　完成数据库连接的建立

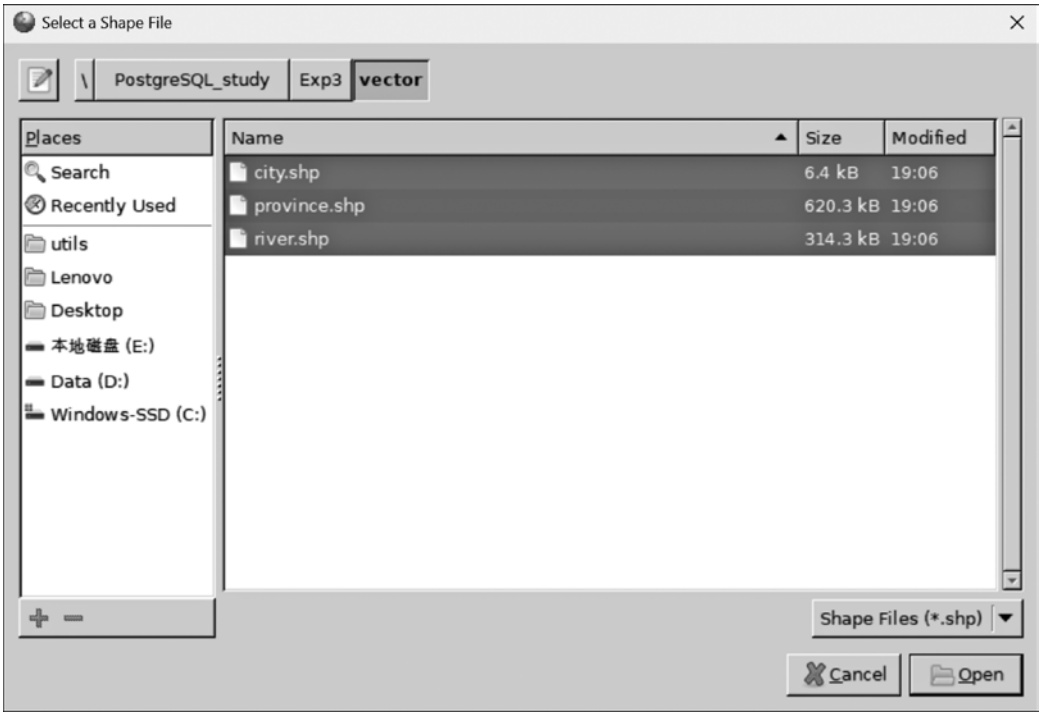


图 3-6　添加待导入的矢量文件

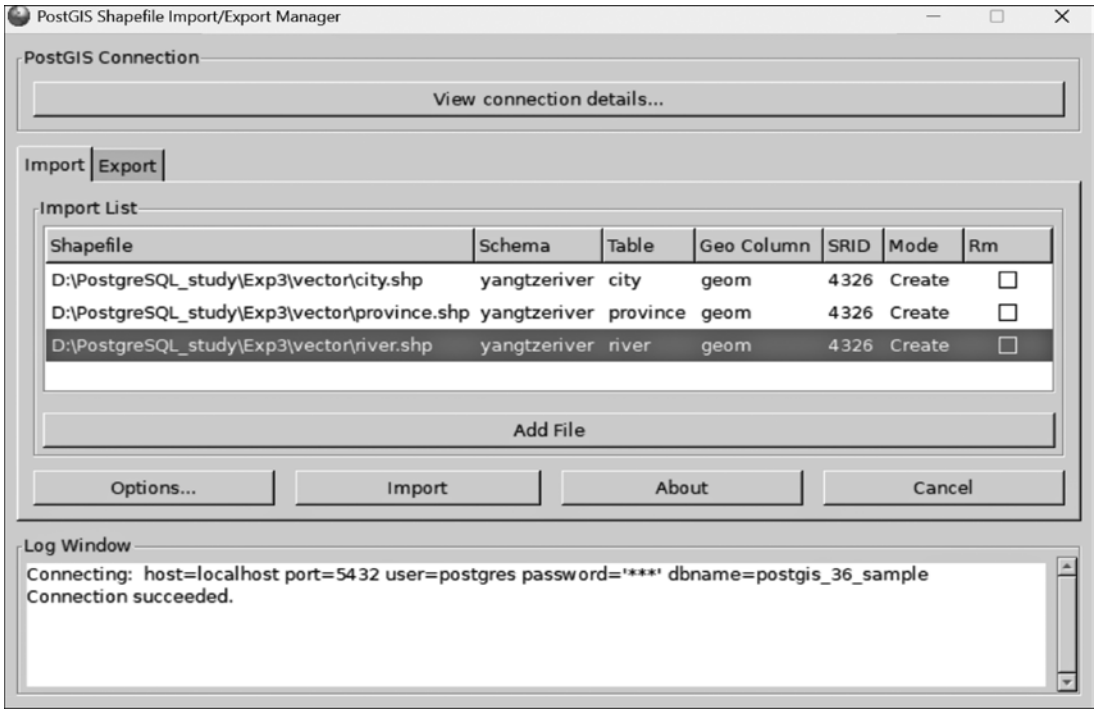


图 3-7　设置空间参考系，指定数据入库的目标模式

图 3-8　完成矢量数据的 PostGIS 入库

3）导入数据的查看

在 pgAdmin4 中找到 postgis_36_sample 的 yangtzeriver 模式，展开【Tables】目录，可以看到数据库中增加了刚导入的 city、province、river 表，如图 3-9 所示。右击“province”表，在快捷菜单中选择【View/Edit Data】→【All Rows】可以看到具体字段，如图 3-10 所示。这里我们可以看到 province 的空间数据已经包含了扩展的 Geometry 类型。在 PostGIS 中，Geometry 类型有 WKT、WKB 等数据格式，还有很多函数以及索引。关于空间数据，可以单击 Geometry 字段旁的地图图标查看，其结果如图 3-11 所示。

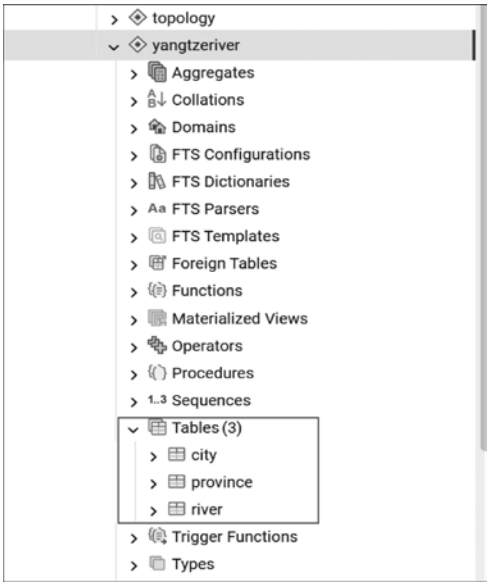


图 3-9　查看数据库中刚导入的 city、province、river 表

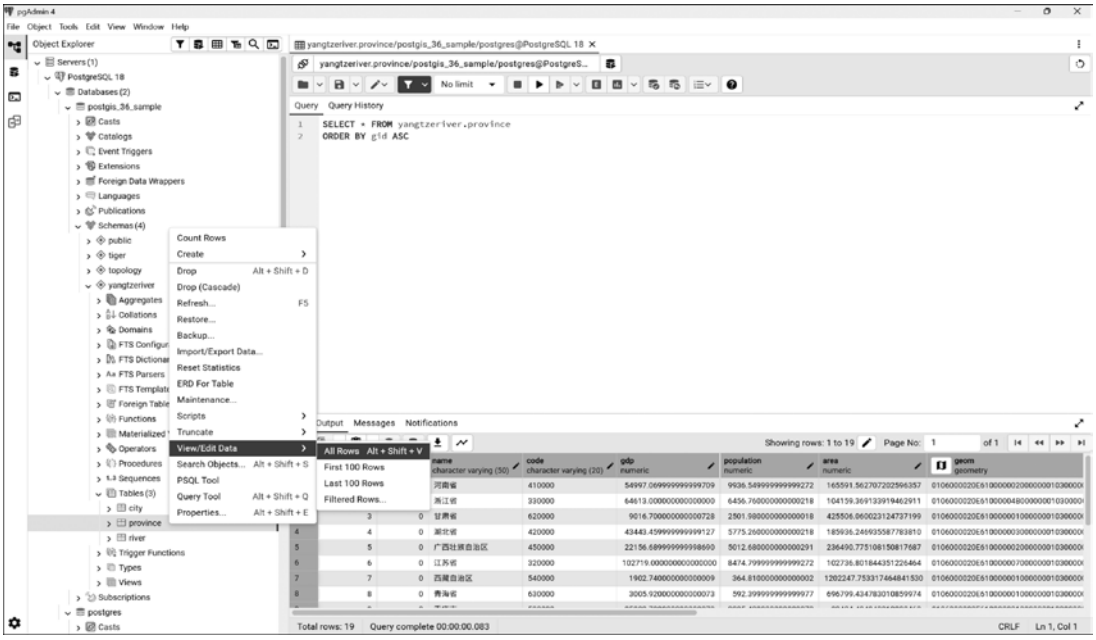


图 3-10　查看表格

4）矢量数据的导出

与导入的前期数据库连接步骤一致（已建立连接无须重复操作），在工具主界面切换至【Export】选项卡，单击【Add Table】，在弹出的表选择对话框中，点选 yangtzeriver 模式下需要导出的 city、province、river 数据表（对话框支持仅显示含地理列的表，便于快速筛选），如图 3-12 所示。选择完成后单击【OK】返回导出界面，单击【Export】，选择文件导出路径并确认，程序将自动完成数据库中空间数据表向 Shapefile 格式的导出，导出成功后 Log Window 会显示相应完成提示，如图 3-13 所示。

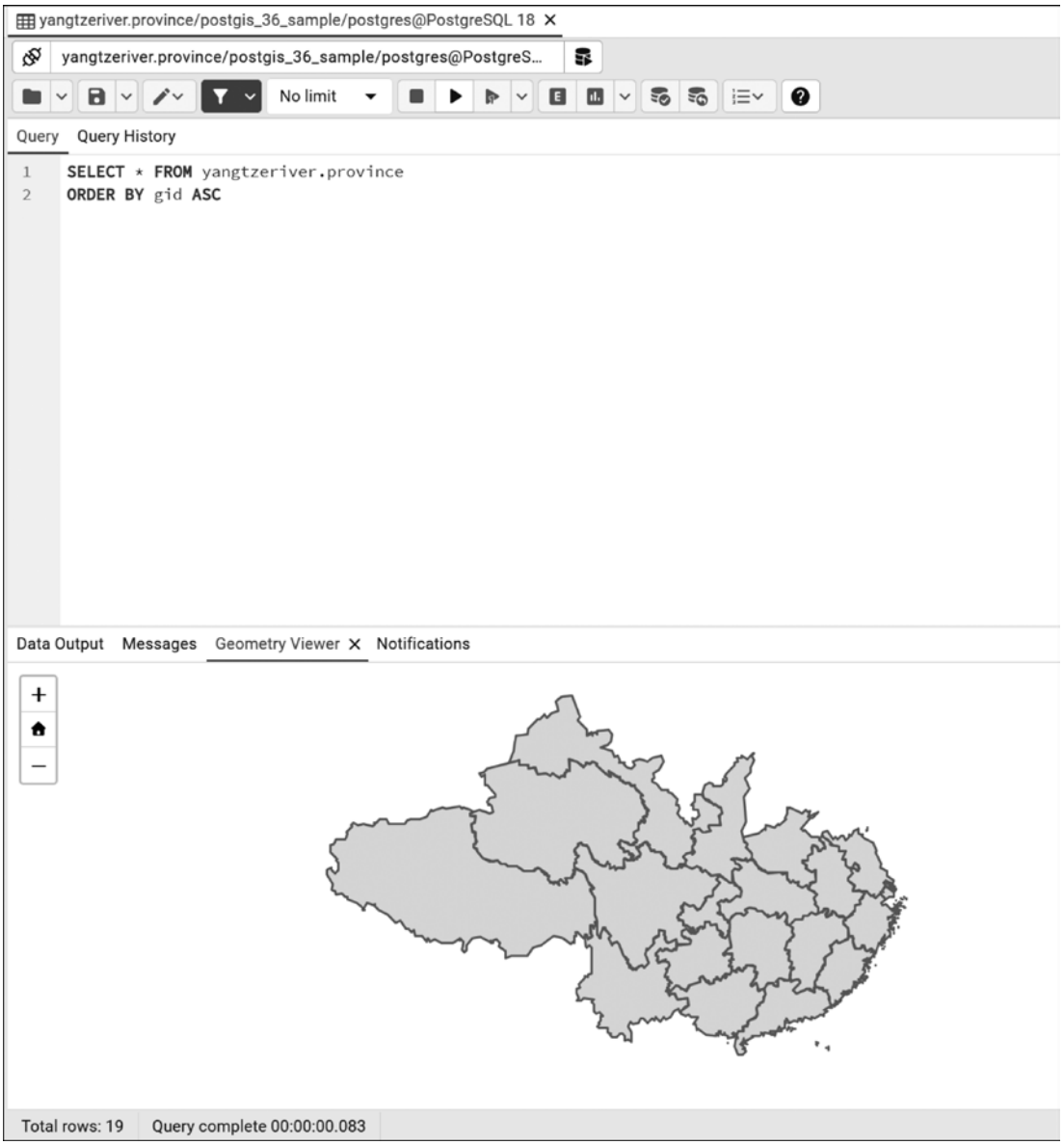


图 3-11　查看空间数据

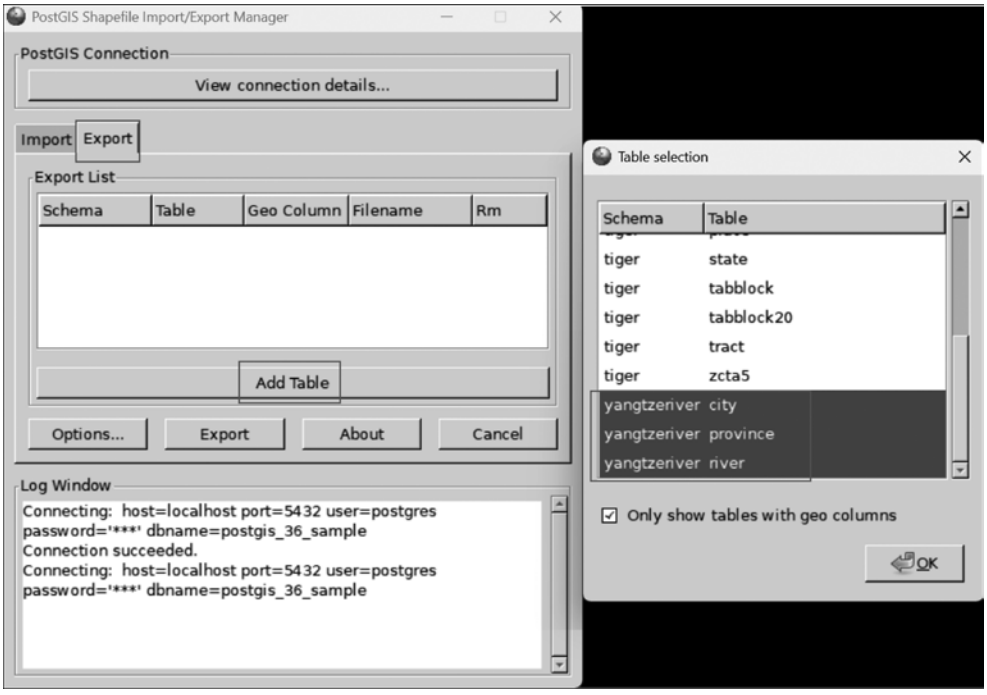


图 3-12　矢量数据导出

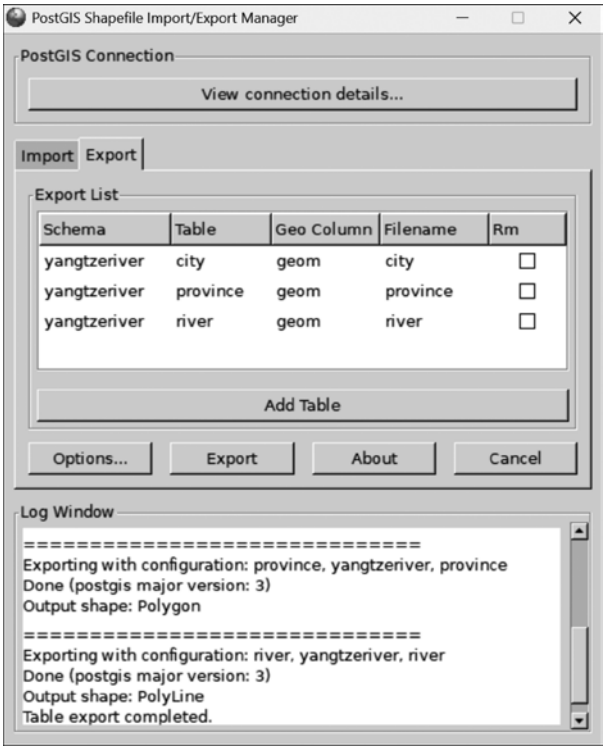


图 3-13　导出成功后 Log Window 显示相应完成提示

2. 使用 QGIS 的 DB Manager 导入导出

QGIS 是一款开源桌面 GIS 软件，支持空间数据库连接及矢量数据的导入与导出操作。首先在 Windows 开始菜单中找到 QGIS Desktop 并启动软件，进入主界面后，右击【PostgreSQL】，在弹出的菜单中选择【新建连接...】(图 3-14)。此时会弹出【新建 PostGIS 连接】对话框(图 3-15)，在对话框中填写相应的数据库连接信息，单击【OK】后，在登录窗口中输入用户名 postgres 及密码 sql，完成数据库连接。

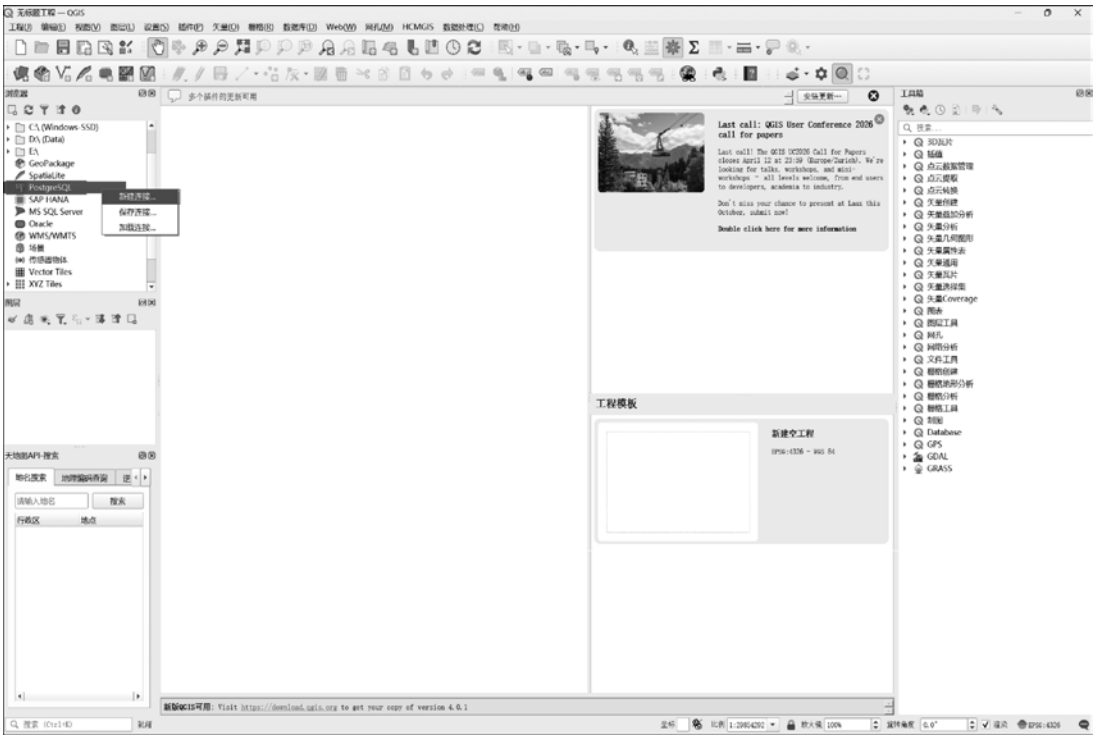


图 3-14　建立数据库连接

1) 矢量数据的导入

数据库连接成功后，在 QGIS 主界面，单击菜单栏【数据库】下的【数据库管理器...】，打开【数据库管理器】界面。在【数据库管理器】中选择目标数据库与目标模式，单击【导入图层或文件】(图 3-16)，弹出【导入矢量图层】对话框(图 3-17)。按照对话框提示，依次选择待导入的矢量文件，并设置对应的数据表参数，单击【OK】，即可完成数据导入至 PostGIS 数据库。

2) 矢量数据的导出

在【数据库管理器】中选中目标数据库模式下待导出的数据表，单击【导出到文件】(图 3-18)。在弹出的【导出到矢量文件】对话框中(图 3-19)完成相关参数配置，单击【OK】，即可将空间数据表导出为指定的文件格式。

3) 矢量数据的查看

在 QGIS 数据库管理器工具中，右击导入的 city/province/river 空间数据，逐一添加到画布，如图 3-20 所示，在 QGIS 软件界面即可查看空间数据，如图 3-21 所示。

图 3-15　数据库连接参数配置对话框

图 3-16　导入图层或文件

图 3-17　设置对应的数据表参数

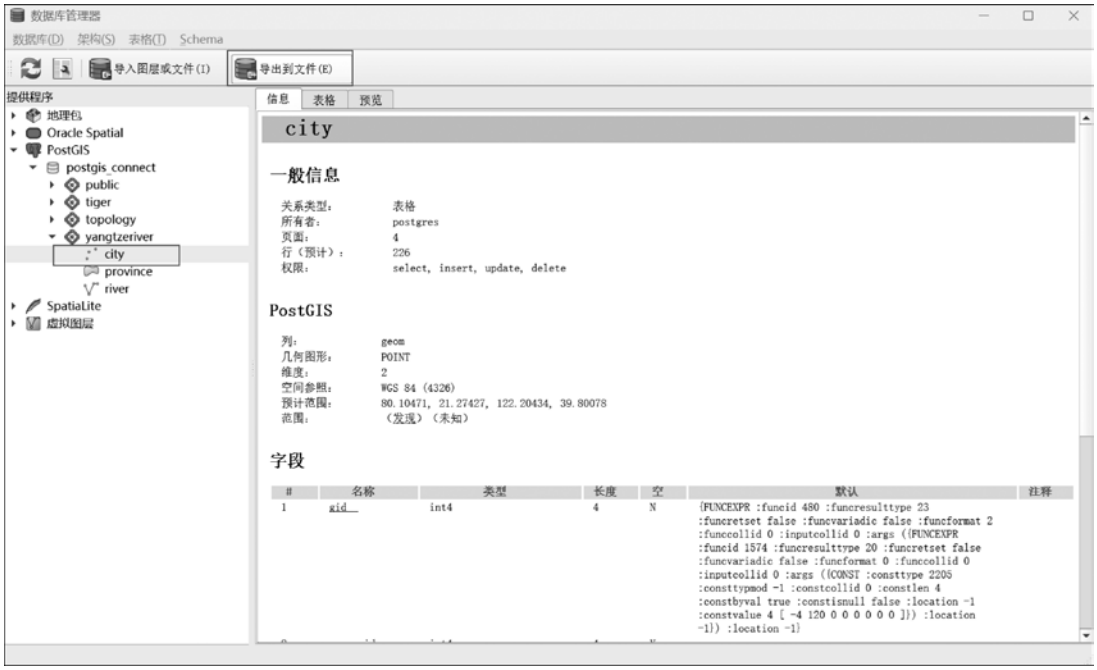


图 3-18　导出到文件

图 3-19　导出参数配置

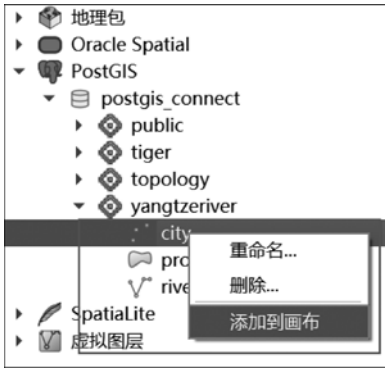


图 3-20　添加到画布

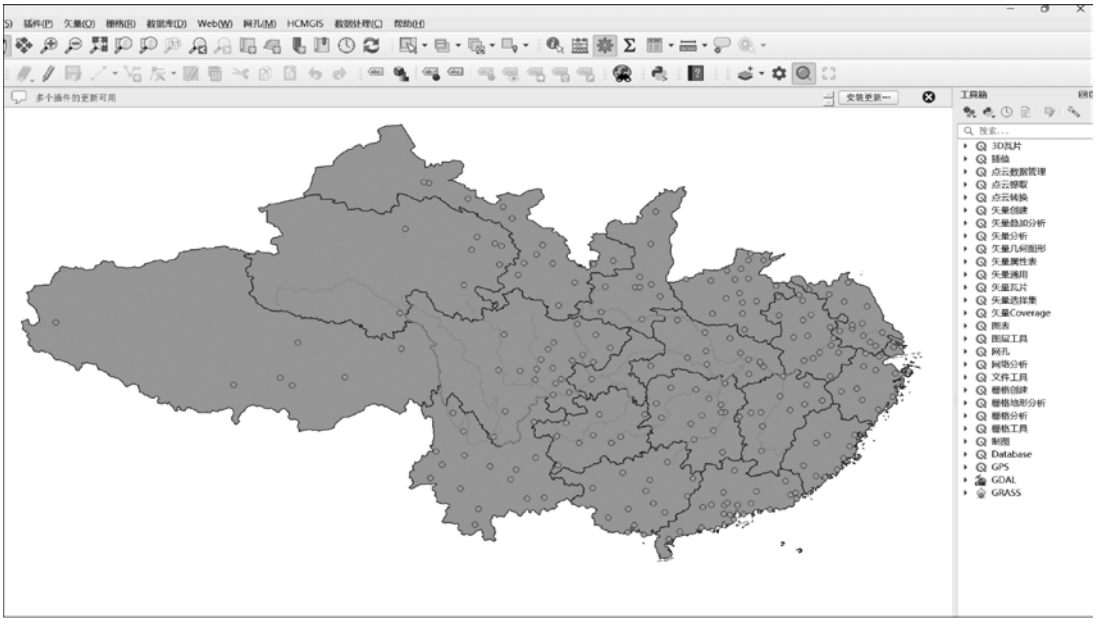


图 3-21　查看空间数据

3.1.2 寄生式体系结构下基于 SuperMap iDesktopX 的矢量数据导入导出

SuperMap iDesktopX 作为专业的 GIS 桌面软件，支持与 PostGIS 空间数据库的直连，可以可视化完成矢量数据的导入与导出操作，操作更贴合 GIS 数据处理的使用习惯，具体步骤如下。

1）启动软件并新建 PostGIS 数据库型数据源

打开 SuperMap iDesktopX 软件，在【开始】选项卡→【数据源】组→【数据库】下拉列表中，单击【新建数据库型数据源】，如图 3-22 所示。在弹出的【新建数据库型数据源】对话框中，选择“PostGIS”，并依次填写服务器地址（localhost）、数据库名称（postgis_yangtzeriver_supermap）、用户名称（postgres）、用户密码（sql）、Schema（public），单击【创建】即可完成参数设置，如图 3-23 所示。启动 pgAdmin4 可刷新看到新建的“postgis_yangtzeriver_supermap”数据库，如图 3-24 所示（注意：创建时请关闭 pgAdmin4 程序）。

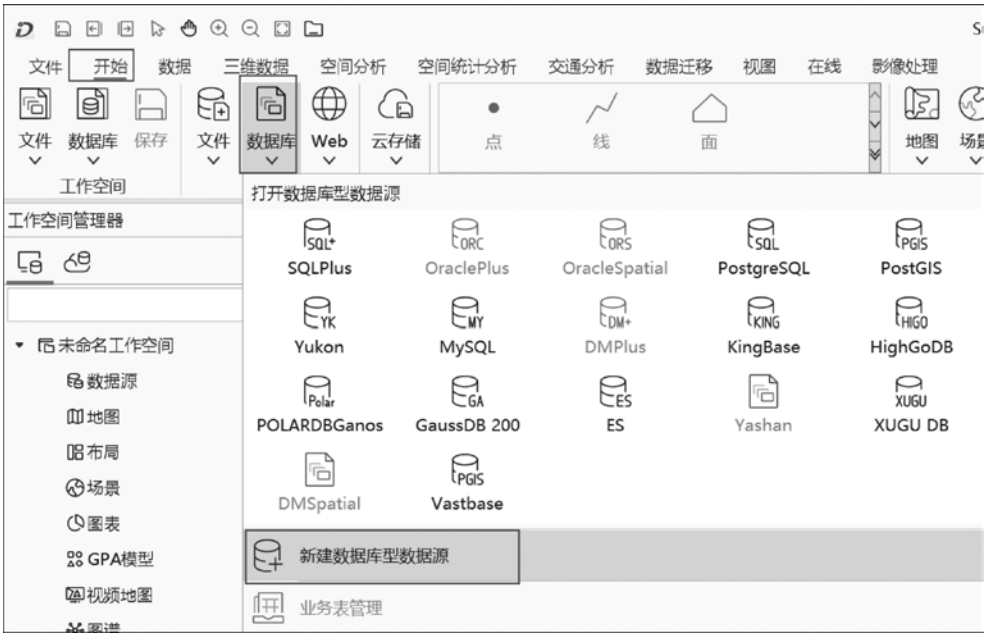


图 3-22 新建数据库型数据源

2）矢量数据导入数据库

在 SuperMap iDesktopX 中，右击刚刚创建的“localhost_postgis_yangtzeriver_supermap_public”数据源，单击【导入数据集...】（图 3-25），在【数据导入】对话框中单击【添加文件】按钮，选择 city.shp、province.shp、river.shp（图 3-26），单击【打开】即可看到导入数据进度条，完成后在“localhost_postgis_yangtzeriver_supermap_public”数据源中可看到导入的数据，如图 3-27 所示。在 pgAdmin4 中刷新数据库即可看到导入的数据。

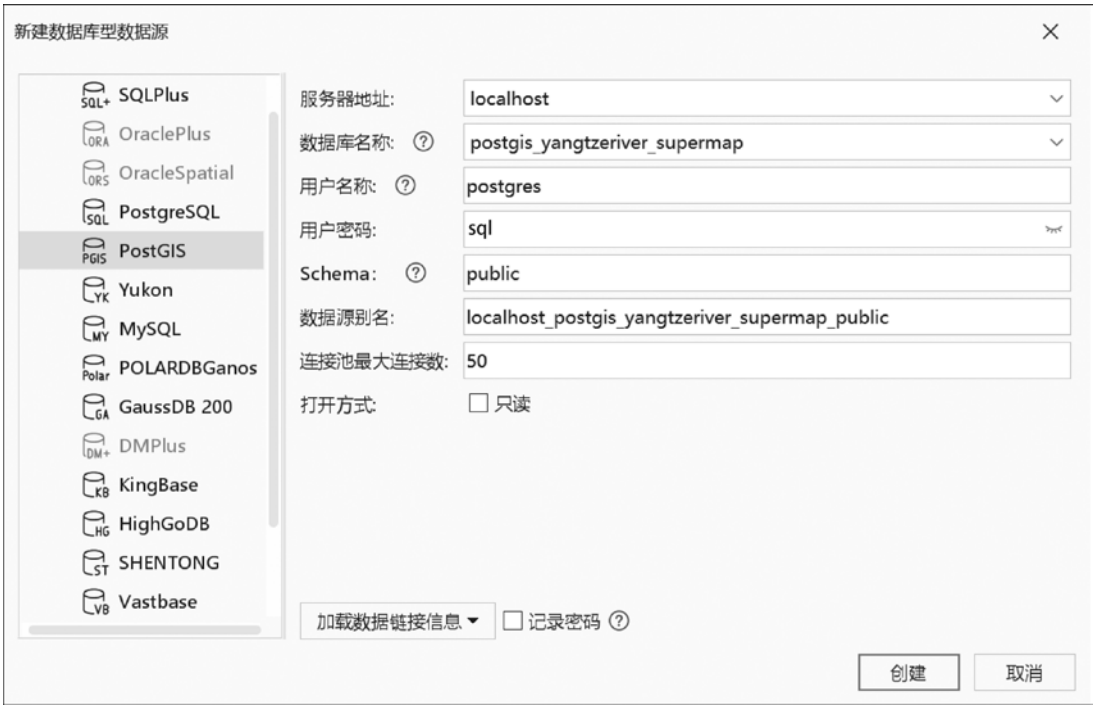


图 3-23　设置数据库型数据源参数

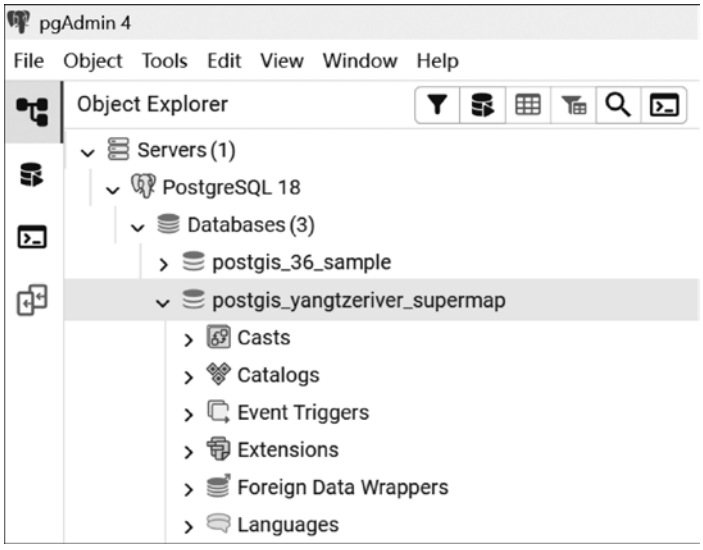


图 3-24　在 pgAdmin 中查看新建的数据库

图 3-25 导入数据集

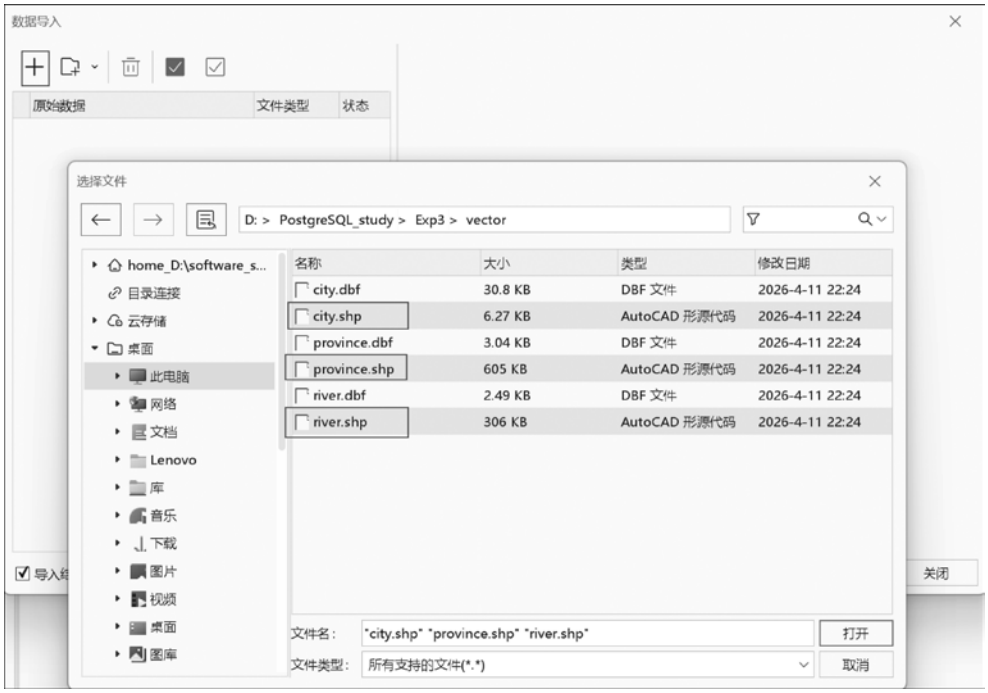


图 3-26 添加文件,选择待导入的图形文件

图 3-27　查看导入的数据

3）矢量数据的查看

选中上一步骤中导入的数据集，右击【添加到新地图】，如图 3-28 所示，即可在软件界面中看到 city. shp、province. shp、river. shp 三份空间数据，如图 3-29 所示。

图 3-28　添加到新地图

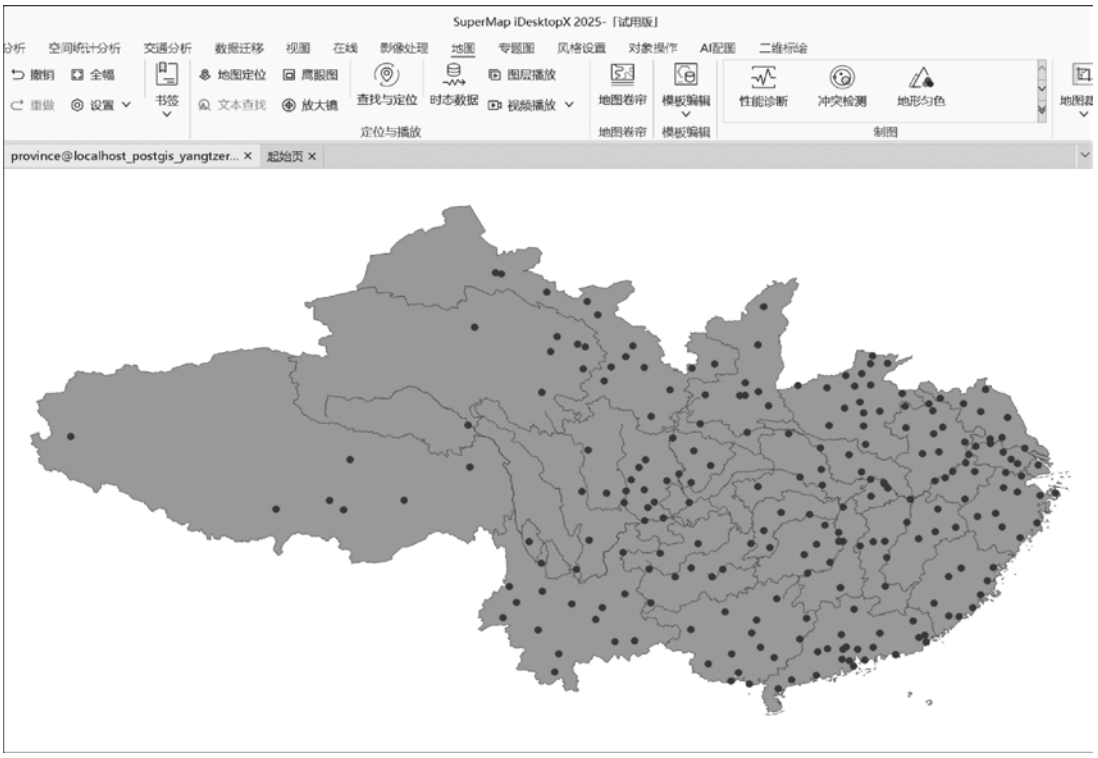


图 3-29　显示三份数据

4）矢量数据导出

选中导入的数据集，右击【导出数据集...】，如图 3-30 所示，设置导出目录即可导出矢量数据。

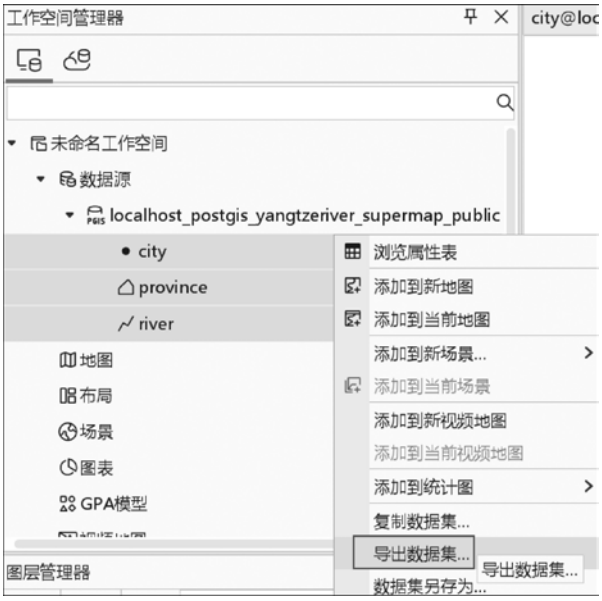


图 3-30　导出数据集

3.1.3　融合式体系结构的栅格数据导入与查看

本节以长江流域1km分辨率DEM栅格数据(yangtze_dem.tif)为例,介绍PostGIS栅格专用导入工具raster2pgsql.exe的使用方法,该工具为命令行工具,可实现栅格数据的切片、金字塔构建与PostGIS数据库入库,并借助QGIS进行查看,具体操作步骤如下。

1. 融合式体系结构的栅格数据导入

1) 定位工具目录

raster2pgsql.exe工具默认位于PostgreSQL安装目录的bin文件夹下,常规路径为C:\Program Files\PostgreSQL\18\bin,需先定位至该目录才能执行导入命令。

2) 打开命令提示符并执行导入指令

以管理员模式打开Windows命令提示符"cmd",如图3-31所示,通过cd命令将当前工作目录切换至上述bin目录,输入栅格数据导入指令,指令中需指定空间参考系、切片大小、金字塔级别、目标数据路径及数据库模式与表名,具体指令如下(路径需根据实际文件位置修改):

```
cd C:\Program Files\PostgreSQL\18\bin
raster2pgsql -I -C -s 4326 -t 128x128 -l 2,4 "C:\Users\XXX\Desktop\Exp3\raster\yangtze_dem.tif" yangtzeriver.dem | psql -h localhost -p 5432 -U postgres -d postgis_36_sample
```

上述指令中各参数的含义为:-I表示创建空间索引;-C表示使用COPY命令加速导入;-s 4326指定空间参考系为EPSG 4326;-t 128x128表示将栅格数据切分为128×128的瓦片;-l 2,4表示生成2级和4级栅格金字塔。后续依次为栅格文件路径、目标模式、表名,最后通过管道符连接psql完成数据库连接与数据写入。

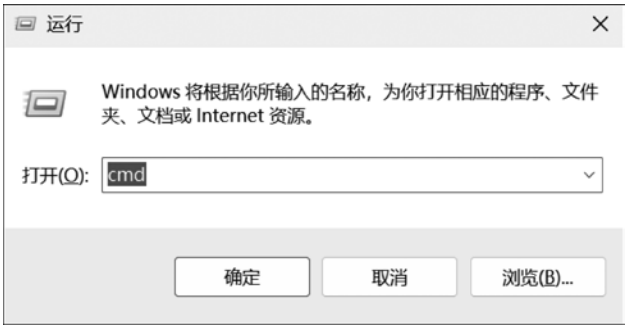


图3-31　打开Windows命令提示符(cmd)

raster2pgsql命令中各项参数的含义如表3-1所示。这里需要注意:命令内的路径需要根据实际进行修改。如编者的raster2pgsql.exe工具存储路径为"D:\software_setup\PostgreSQL\18\bin";待导入的栅格数据路径为"D:/supermap/yangtzeriver_data/raster/yangtze_dem.tif",如图3-32所示。

表 3-1 raster2pgsql 命令中的各项参数

参　数	说　明
-I	表示创建空间索引
-C	表示使用 COPY 命令导入数据
-s 4326	表示使用 EPSG 4326 投影进行转换，可以根据需要修改
-t TILE_SIZE	表示将栅格切分成瓦片(tiles)，TILE_SIZE 表示为 WIDTH×HEIGHT 的瓦片大小
-l LEVEL1，…	表示生成缩略图，后面的数字表示生成缩略图的级别
\path\to\raster\file. tif	表示栅格数据的路径和文件名
schema_name. table_name	表示要导入的模式名和表名

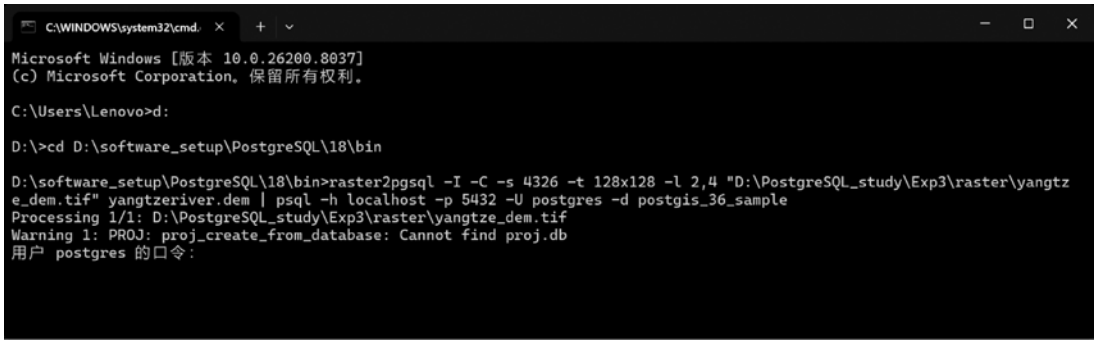


图 3-32 输入命令

3）执行导入并查看结果

输入指令后按回车键，根据提示输入 PostgreSQL 数据库密码，程序将启动栅格数据的导入工作，导入时间取决于栅格文件大小。若导入参数正确、文件无损坏，命令行窗口会显示“Processing”“addrasterconstraints”“COMMIT”等执行提示，无报错即表示数据导入成功，如图 3-33 所示。

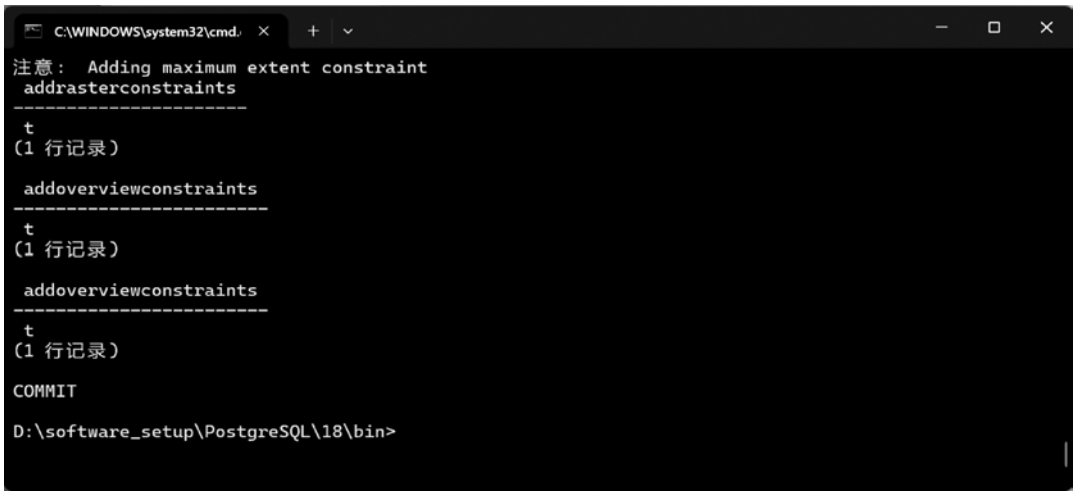


图 3-33 执行提示

栅格数据导入成功后，打开 pgAdmin 并进入 postgis_36_sample 数据库的 yangtzeriver 模式，可看到新增的 3 个栅格相关数据表：dem（栅格原始数据）、o_2_dem（2 级金字塔数据）、o_4_dem（4 级金字塔数据），完成栅格数据的 PostGIS 入库，如图 3-34 所示。

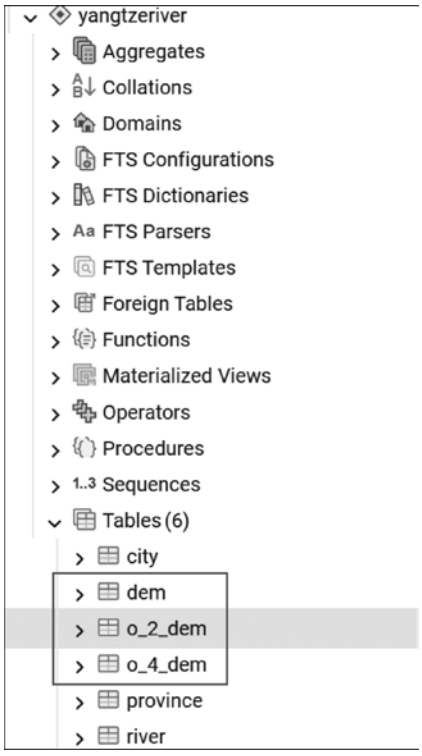


图 3-34　完成栅格数据的 PostGIS 入库

2. 使用 QGIS 查看栅格数据

在 QGIS 数据管理器工具中，右击导入的 DEM 数据，单击【添加到画布】，在 QGIS 软件界面中即可查看 DEM 数据，如图 3-35 所示。

3.1.4　寄生式体系结构下基于 SuperMap iDesktopX 的栅格数据导入查看

启动 SuperMap iDesktopX 软件，在【打开数据库型数据源】对话框，输入服务器地址、数据库名称、用户名称、用户密码、Schema 等信息，单击【打开】，连接数据库，如图 3-36 所示。

与导入矢量数据操作一致，右击数据源，在快捷菜单中选择【导入数据集...】，添加“yangtze_dem. tif”数据，编码类型选择“LZW”，数据集类型选择“栅格”，单击【导入】，如图 3-37 所示。双击数据源下的“yangtze_dem”栅格数据即可在软件主界面查看该数据，SuperMap iDesktopX 软件会自动设置默认颜色表，如图 3-38 所示。

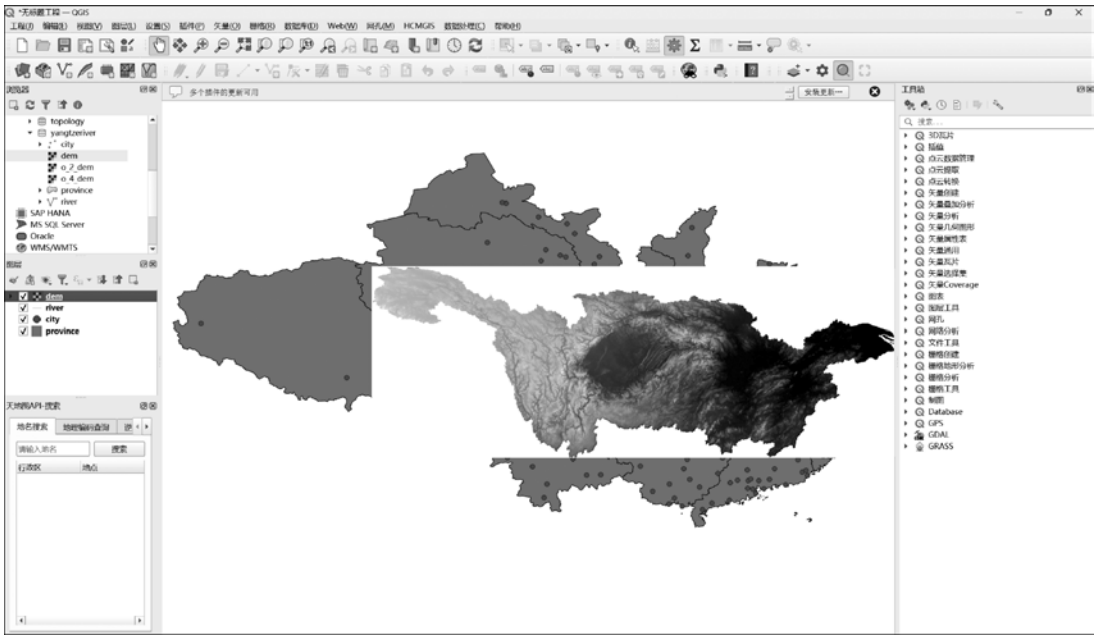


图 3-35　在 QGIS 软件界面中查看 DEM 数据

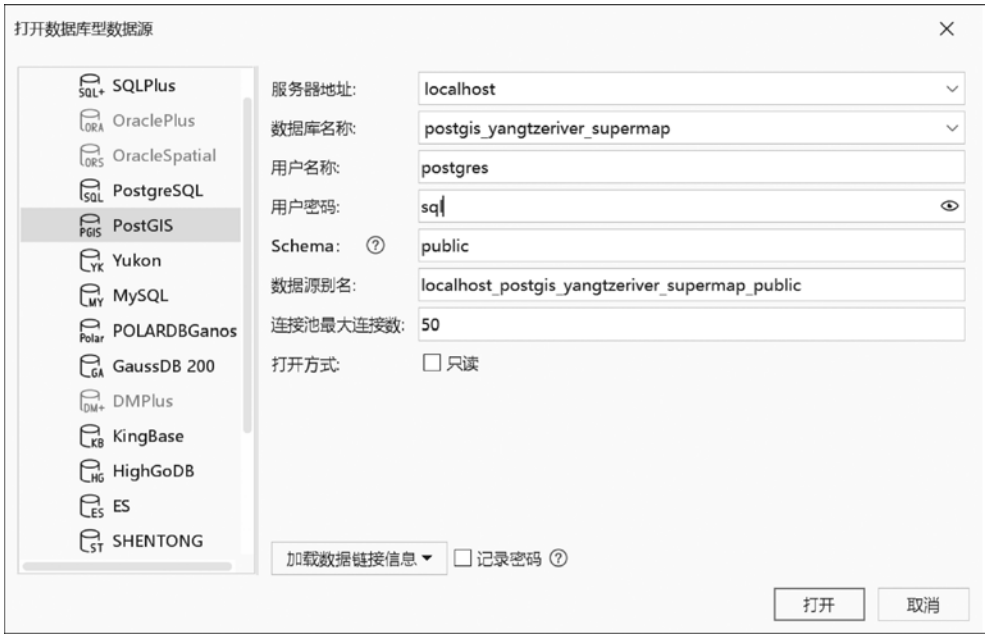


图 3-36　【打开数据库型数据源】对话框

图 3-37　导入栅格数据

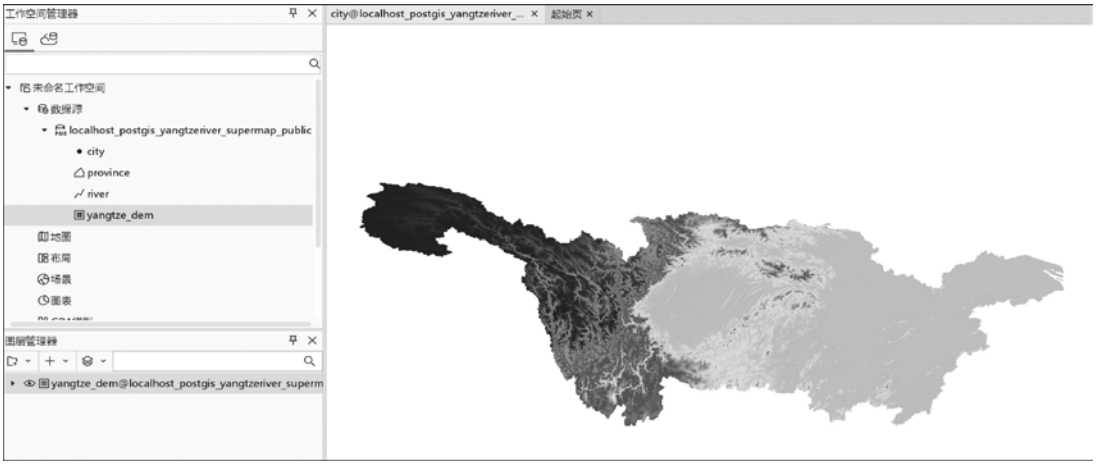


图 3-38　查看栅格数据

3.2　融合式体系结构下长江流域数据库 GSQL 查询分析实验

3.2.1　矢量空间查询

例 1：查询 province 表中 GDP 低于 1 万亿元的省级行政区。

```
1. SELECT pro.name FROM yangtzeriver.province pro WHERE pro.gdp<10000;
```

例 2：列出 GDP 超过 2 万亿元的省级行政区的省会和人口数。

```
SELECT ct.name, ct .population
FROM yangtzeriver.city ct, yangtzeriver.province pro
WHERE pro.gdp>20000 AND ct.capital='Y' AND pro.name=ct.province;
```

例 3：统计长江流域地级市的数量。

```
SELECT COUNT(*) FROM yangtzeriver.city;
```

例 4：统计 city 表中非省会城市的平均人口。

```
SELECT AVG(ct.population) FROM yangtzeriver.city ct WHERE ct.capital='N';
```

例 5：统计各省级行政区地级市的平均 GDP。

```
SELECT ct.province, AVG(ct.gdp) AS mean_gdp
FROM yangtzeriver.city ct
GROUP BY ct.province;
```

例 6：查找长江支流中最长的河流及其长度。

```
SELECT r.name ,r.shape_leng
FROM yangtzeriver.river r
WHERE r.mainstream='N'
ORDER BY r.shape_leng DESC
LIMIT 1;
```

例 7：列出与湖北省相邻的省级行政区名字。

```
SELECT prol.name AS "Neighbors of hubei"
FROM yangtzeriver.province prol, yangtzeriver.province pro2
WHERE ST_Touches(prol.geom,pro2.geom) AND pro2.name='湖北省';
```

例 8：列出 river 表中所有河流所流经的省级行政区。

```
SELECT r.name,pro.name
FROM yangtzeriver.river r, yangtzeriver.province pro
WHERE ST_Crosses(r.geom, pro.geom) OR ST_Within(r.geom, pro.geom);
```

例 9：列出 river 表中所有河流及距其最近的城市名称。

```
SELECT rl.name,cl.name FROM yangtzeriver.river rl, yangtzeriver.city cl
WHERE ST_Distance(rl.geom,cl.geom)<ALL
(SELECT ST_Distance(rl.geom,c2.geom)
FROM yangtzeriver.city c2
WHERE cl.name<>c2.name);
```

例 10：假设汉江仅能为方圆 300km 以内的城市供水，请列出能从汉江获取水的城市。

```
SELECT c.name,c.geom FROM yangtzeriver.city c, yangtzeriver.river r
WHERE ST_Within(c.geom,Geometry(
ST_Buffer(Geography(r.geom),300000)))AND r.name='汉江';
```

注意：此处将 r.geom 转换为 Geography 的目的是：由于研究区域较大，这里采用球面距离计算缓冲区；得到缓冲区后，将其转化为 Geometry 类型，再判断 c.geom 是否在缓冲区内。

例 11：列出 province 表中每个省份的名字、人口、面积。

```
1. SELECT pro.name,pro.population, ST_Area(pro.geom) AS Area
2. FROM yangtzeriver.province pro
3. ORDER BY Area DESC;
```

例 12：列出每个省份的 GDP 及其省会到赤道的距离。

```
1. SELECT pro.name,c.name,pro.gdp,
2. ST_Distance(ST_SetSrid(ST_ Point(ST_X(c.geom),0),4326),c.geom) AS Distance
3. FROM yangtzeriver.province pro, yangtzeriver.city c
4. WHERE c.province=pro.name AND c.capital='Y'
5. ORDER BY distance DESC;
```

注意：ST_Point 的作用是生成一个 X 与 c.geom 的 X 相同、Y 为 0 的点，即 c.geom 在赤道上的垂点，然后求该垂点与 c.geom 的距离。

例 13：列出 city 表中所有城市两两间的距离。

```
1. SELECT c1.name,c2.name, ST_Distance(Geography(c1.geom),Geography(c2.geom)) *
   0.001 AS "distance(km)"
2. FROM yangtzeriver.city c1, yangtzeriver.city c2
3. WHERE c1.name<>c2.name;
```

注意：ST_Distance()函数计算出来的是球面距离，单位是 m，需要进行单位转换，即乘以 0.001 转换成 km。

例 14：根据 river 表 geom 字段生成河流长度，验证该表中 length 的属性字段是否正确。

```
1. SELECT r.name, r.shape_leng,ST_Length(ST_GeographyFromText(ST_AsText(r.geom))) *
   0.001 AS "length(km)"
2. FROM yangtzeriver.river r;
```

注意：ST_GeographyFromText 用于将 Geometry 类型转换为 Geography 类型，其目的是计算河流在球面上的距离，由于球面坐标计算的长度单位是 m，所以需要乘以 0.001 转换为 km。根据计算结果可见，表中 length 字段下的属性值是正确的。

例 15：列出邻省最多的省级行政区的名称。

```
1. SELECT p1.name, COUNT(p2.name) AS "Number of neighbors"
2. FROM yangtzeriver.province p1, yangtzeriver.province p2
3. WHERE ST_Touches(p1.geom, p2.geom)
4. GROUP BY p1.name
5. ORDER BY COUNT(p2.Name) DESC
6. LIMIT 1;
```

例 16：计算河流在其所流经的各省级行政区境内的长度。

```
1. SELECT r.name,p.name,
2. ST_Length(ST_GeographyFromText(
3. ST_ASTEXT(ST_Intersection(r.geom,p.geom))))*0.001 AS "length(km)"
4. FROM yangtzeriver.river r, yangtzeriver.province p
5. WHERE ST_Crosses(r.geom,p.geom) OR ST_Within (r.geom,p.geom);
```

注意：ST_Intersection()函数生成各河流 Geometry 位于各省级行政区 Geometry 内的部分，即河流 Geometry 被各省级行政区 Geometry 切割后内部的那部分。

3.2.2 栅格空间查询

例 17：列出 DEM 数据表中各记录(瓦片)中心点处的纬度值。

```
1. SELECT ST_UpperLeftY(t1.rast)-0.5*ST_Height(t1.rast)/2*ST_ PixelHeight(t1.
   rast)
2. FROM yangtzeriver.dem t1;
```

注意：ST_UpperLeftY 返回各记录(瓦片)左上角点的纬度(单位：度(°))，ST_Height 则返回记录(瓦片)的高度(单位：像元)，ST_PixelHeight 返回像元的高度(单位：度(°))。

例 18：基于 DEM 数据表计算长江流域地区坡度。

由于 DEM 栅格数据表的坐标系统为地理坐标系，其水平距离的度量单位为度，而 DEM 数据在垂直方向上的高程单位为米(m)，在计算坡度前需要将单位统一。针对单位不匹配的问题，为了获得更加准确的坡度计算结果，有以下两种解决方案。

(1) 根据栅格中心估算地理坐标系(°)与投影坐标系(m)的缩放比，再计算坡度。

在计算坡度时，PostGIS 的 ST_Slope 函数提供了 scale 参数、栅格的地理坐标系(°)与投影坐标系(m)之间的缩放比。实际上，在纬度为 θ 的纬线方向上，经度每差 1°，其球面水平距离大约相差(111 120×cosθ)m，即纬线方向的 scale 值；在所有的经线方向上，纬度每差 1°，其球面垂直距离大约为 111 120m，即经线方向的 scale 值。由于 ST_Slope 函数仅提供了一个 scale 参数，无法精确区分纬线、经线两个方向的 scale 值，故这里粗略地将两个方向 scale 值求平均，作为 scale 参数输入 ST_Slope 函数，实现各栅格单元坡度的粗略估算。对应的 GSQL 语句如下：

```
1. CREATE TABLE yangtzeriver.slope_cal_scale AS
2. SELECT ST_Slope(
3. ST_Union(t1.rast), 1, '32BF', 'DEGREES',
4. 0.5*(111120+ cosd( --scale
5. ST_UpperLeftY(t1.rast)-0.5*ST_Height(t1.rast)*ST_ PixelHeight(t1.rast)
6. )*111120),
7. false) AS rast
8. FROM yangtzeriver.dem t1
9. GROUP BY t1.rid, t1.rast;
```

其中，cosd 用于计算 DEM 栅格表中各记录的中心点纬度的余弦值，第一个 0.5 用于计算各记录(瓦片)经线、纬线方向上 scale 的平均值。

(2) 将栅格数据转换到合适的投影坐标系后再计算坡度。

定义了投影坐标系后,数据的长度单位便转换成了 m,所以此时的 scale 设置为 1 即可。本例中采用的投影坐标系为：EPSG：4479,China Geodetic Coordinate System 2000。

```
1. CREATE TABLE yangtzeriver.slope_proj AS
2. SELECT ST_Slope(ST_Transform(ST_Union(t1.rast),4479),1,'32BF','DEGREES',1,
   FALSE) as rast
3. FROM yangtzeriver.dem t1;
```

经统计,第二种解决方案采用投影转换方式得到的坡度取值范围为[0,26.2646°],第一种解决方案粗略估计 scale 方式获得的坡度取值范围为[0,26.2578°],坡度误差会随着纬度的升高而逐渐增大。因此,第二种方案的结果应该更精确。

例 19：基于 DEM 数据表计算长江流域地区坡向。

```
1. CREATE TABLE yangtzeriver.aspect AS
2. SELECT ST_Aspect(ST_Union(t1.rast), 1, t1.rast) AS rast
3. FROM yangtzeriver.dem t1
4. GROUP BY t1.rid, t1.rast;
```

例 20：基于 DEM 数据表计算长江流域地区地表粗糙度。

```
1. CREATE TABLE yangtzeriver.roughness AS
2. SELECT ST_Roughness(ST_Union(t1.rast), 1, t1.rast) AS rast
3. FROM yangtzeriver.dem t1
4. GROUP BY t1.rid, t1.rast;
```

例 21：计算长江流域地区 DEM 数据分布直方图。

```
1. SELECT (stats).*
2. FROM (SELECT rid, ST_Histogram(rast) AS stats
3. FROM yangtzeriver.dem) AS foo;
```

例 22：计算长江流域地区 DEM 数据统计信息。

```
1. SELECT (stats).count,
2. round((stats).sum::numeric, 3) AS "sum",
3. round((stats).mean::numeric, 3) AS "mean",
4. round((stats).stddev::numeric, 3) AS "stddev",
5. round((stats).min::numeric, 3) AS "min",
6. round((stats).max::numeric, 3) AS "max"
7. FROM (SELECT ST_SummaryStatsAgg(rast, 1, TRUE, 1) AS stats
8. FROM yangtzeriver.dem) bar;
```

例 23：统计 rid=2 的 DEM 瓦片海拔的四分位数。

```
1. SELECT (pvq).*
2. FROM (SELECT ST_Quantile(rast) AS pvq
3. FROM yangtzeriver.dem WHERE rid=2) AS foo
4. ORDER BY (pvq).quantile;
```

3.2.3 栅格与矢量混合空间查询

例 24：裁剪出重庆市的 DEM 数据。

```
1. CREATE TABLE yangtzeriver.demcq AS
2. SELECT ST_Union(ST_Clip(rast, geom)) AS rast
3. FROM yangtzeriver.dem
4. CROSS JOIN (SELECT geom FROM yangtzeriver.province WHERE name='重庆市') AS geom
5. WHERE ST_Intersects(rast, geom);
```

注意：CROSS JOIN 表示对其前后的两张表做笛卡儿积。

例 25：生成各省份 DEM 数据统计信息表。

```
1. DROP TABLE IF EXISTS yangtzeriver.zonal_stats;
2. CREATE TABLE yangtzeriver.zonal_stats AS
3. SELECT name, geom,(ST_SummaryStats(ST_Union(ST_Clip(rast,geom)),1,TRUE)).*
4. FROM yangtzeriver.dem JOIN yangtzeriver.province on ST_Intersects(rast,geom)
5. GROUP BY province.name,geom;
```

注意：ST_Clip 用于裁剪出位于省级行政区内的栅格数据，ST_Union 用于合并位于省级行政区内的所有栅格记录。

注意：PostgreSQL 中，未加双引号的标识符会被自动转为小写，例如创建的 Yangtzeriver 模式会被识别为 yangtzeriver，导致找不到表。若模式名、数据名是混合大小写，SQL 中必须加双引号引用。

3.3 长江流域数据库局域网环境的远程访问

PostgreSQL 数据库默认仅允许本地(127.0.0.1 或 localhost)连接，无法直接被局域网内其他机器访问。下面介绍在局域网 C/S 架构下，如何实现客户端对远程 PostgreSQL 服务器的访问配置。

C/S 架构能充分发挥客户端与服务器端的硬件性能，通过合理分配业务处理任务，有效减少网络通信开销，提升系统运行效率。本书中将提供数据库服务的机器称为服务器(S 机)，发起访问请求的机器称为客户端(C 机)，具体参数配置与访问步骤详见 3.3.1 节。

3.3.1 搭建 S 机和 C 机的局域网络环境

搭建局域网需要相应的网络设备。本节介绍一种简便、快捷且低成本的局域网搭建方法，即利用 Windows 系统自带的移动热点功能组建临时局域网，供读者开展数据库远程访问实验使用。服务器端 S 机与客户端 C 机的网络环境构建步骤如下。

首先，右击系统托盘区的无线网络图标，选择【设置】选项，进入【网络和 Internet】设置界面，如图 3-39 所示。在该界面中选择【移动热点】选项，并启用移动热点功能。接下来，在服务器端 S 机的属性选项卡中，设置网络名称(SSID)与连接密码，供其他设备接入该移动热点。