浙江大学
ZHEJIANG UNIVERSITY

地理空间数据库期末复习

陶煜波
计算机科学与技术学院

空间数据

1. Reality

2. Information Model
   (conceptual model)

Objects
(discrete features)

Fields
(continuous features)

3. Representation or
   Data Model
   (logical model)

Vector, Raster, Networks, TIN, DEM, etc.

4. Databases
   (physical model)

Object-oriented, georelational geodatabases, etc.

5. File Structures
   (binary model)

Indexes, arrays, lists, sequential files, etc.

空间分析

Spatial Relationship	Description
Measurements 度量	What are the physical distances, lengths, and area? - patterns, distance, perimeter, area
Proximal 邻近	Are within a distance of? - near, far, absolute, relative
Topological 拓扑	Are spatially identical to? Identical means same shape, size on top of each other, same vertices, and same location - partially equivalent (intersect), within (contained/ adjacent) Are spatially adjacent to? - neighbours
Directional 方向	Bearings and compass directions - north, south, east, west

空间计算 (Spatial Computing)

- Spatial Computing
  - A set of ideas and technologies that tansforming our lives by understanding the physical world, knowing and communicating our relation to places in that work, and navigating through these places
  - Applications
    - A Uber driver knows precisely where customers are, nearby points of interest, and how to reach their destinations
    - Groups of friends can form impromptu events via "check-in" models used by Facebook and foursquare
    - Scientists use GPS to track endangered species to better understand behavior
    - Farmers use GPS for precision agriculture to increase crop yields while reducing costs
    - Google Earth is being used to teach children about their neighborhoods and the world
    - Augmented reality applications (e.g., Pokemon Go) are providing real-time place labeling in the physical world and providing people detailed information about major landmarks nearby

课程安排 – 理论 应用 实现

日期	课程内容	作业	日期	课程内容	作业
9.14	地理空间数据库概论 关系模型与关系代数		11.9	空间索引	Index
9.21	中秋节放假	SQL	11.16	空间查询处理与优化	*Quiz 4*
9.28	关系数据库标准语言	*Quiz 1*	11.23	空间网络模型	Network
10.9	几何对象模型	SSQL	11.30	pgRouting 视图与触发器	*Quiz 5*
10.12	PostGIS空间几何查询	*Quiz 2*	12.7	PostgreSQL函数与触发器	App
10.19	数据库设计 空间扩展E-R图	Design	12.14	事务处理	
10.26	关系数据库设计理论	*Quiz 3*	12.21	*Quiz 6*	
11.2	空间存储与索引		12.28	期末复习	

- 数据库系统原理
- 地理空间数据库

第一章 地理空间数据库概论

- 关系数据库基本概念
  - 概念数据模型 → 逻辑数据模型 → 物理数据模型
  - 逻辑数据模型
    - 层次、网状、关系、面向对象、对象关系
  - 关系模型
    - 内模式、模式、外模式
    - 映像关系 (逻辑独立性，物理独立性)
- 地理空间数据库基本概念
  - 矢量数据，栅格数据
  - 特点：空间特征，非结构化特征，空间关系特征，时态特征，多尺度特征
  - 地理空间数据库 = 关系数据库管理系统 + 空间扩展
    - PostgreSQL + PostGIS (空间数据类型、空间分析、空间索引)

第一章 地理空间数据库概论

- 空间数据管理技术的产生与发展
  - 文件系统
  - 文件与关系数据库混合管理系统
    - 空间数据-文件系统，属性数据-关系数据库
  - 空间数据引擎
    - 空间+属性-关系数据库，独立GIS数据引擎解析空间数据
  - 对象关系型数据库管理系统 (扩展SQL)
- 空间数据库标准
  - SFA SQL (几何对象模型、注记文本模型，PostGIS)
  - SQL/MM (几何对象模型、空间网路模型， Oracle Spatial)
- 空间计算
  - 空间相关性/关联性 (因果关系→相关关系)
  - 时空查询

第二章 关系模型与关系代数

- 数据结构
  - 关系：关系名、关系模式和关系实例
    - 基本关系、查询表、视图表
  - 关系模式：$R(A_1, A_2, \ldots, A_n)$
  - 逻辑结构：二维表
    - 属性、域、码、元组、分量
- 数据操作
  - 集合操作 (输入和输出都是集合)
  - 查询
    - 选择、投影、连接、除、并、交、差
  - 数据更新
    - 插入、删除、修改
  - 关系代数

关系模型

- 数据结构
- 数据操作
- 完整性约束 (关系的约束条件)
  - 实体完整性
    - 主码(primary key)，不能取NULL (unknown or undefined)
  - 参照完整性
    - 外码(foreign key)，取NULL或参照关系中的主码或Unique值
  - 用户定义完整性

sid	Name	GPA
101	Bob	3.2
123	Mary	3.8

Students

Primary Key:
Students: sid
Courses: cid
Enrolled: sid, cid

sid	cid	Grade
123	564	A

Enrolled

cid	cname	credits
564	564-2	4
308	417	2

Courses

关系代数

- 运算符
  - 集合运算符
    - 并∪，交∩，差−，笛卡尔积×
  - 专门的关系运算符
    - 选择σ，投影π，连接 ⋈，除÷
  - 算术比较符
    - > ≥ < ≤ = ≠
  - 逻辑运算符
    - ¬ ∧ ∨
  - 辅助操作
    - 重命名 ρ
- 关系代数 - set
  - 扩展关系代数 (SQL) - multiset, bag
    - 重复消除，分组与聚集，排序

第三章 SQL

CREATE TABLE <表名>

(<列名> <数据类型>[<列级完整性约束条件>]

[，<列名> <数据类型> [<列级完整性约束条件>]] …

[，<表级完整性约束条件>]);

INSERT INTO <表名> [(<属性列1>[，<属性列2 >] …)]

VALUES (<常量1> [，<常量2>] …);

UPDATE <表名>

SET <列名>=<表达式>[，<列名>=<表达式>]…

[WHERE <条件>];

DELETE

FROM <表名>

[WHERE <条件>];

SQL功能	动词
数据定义	CREATE, DROP, ALTER
数据查询	SELECT
数据操纵	INSERT, UPDATE, DELETE
数据控制	GRANT, REVOKE

约束条件与创建

- 约束条件 (Constraint，静态约束)
  - NOT NULL　　列
  - UNIQUE　　列或列集合
  - PRIMARY KEY　　列或列集合，自动获得UNIQUE
  - FOREIGN KEY　　列或列集合

    预防破坏表之间连接的动作, 防止非法数据插入外键列

    Restrict, Cascade, Set NULL (权限管理)
  - CHECK　　列或列集合
  - DEFAULT　　列

操作对象	操作方式		
	创建	删除	修改
基本表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	
触发器	CREATE TRIGGER	DROP TRIGGER	

数据查询

SELECT [ALL|DISTINCT]

<目标列表达式> [别名] [，<目标列表达式> [别名]] …

FROM <表名或视图名> [别名]

[，<表名或视图名> [别名]] …

[WHERE <条件表达式>]

[GROUP BY <列名1>[，<列名2>] …

[HAVING <条件表达式>]]

[ORDER BY <列名1> [ASC|DESC]

[，<列名2> [ASC|DESC]] …];

语义上的执行顺序：
1. FROM: 关系笛卡尔积
2. WHERE: 选择满足条件的行
3. GROUP BY: 根据属性分组
4. HAVING: 选择满足条件的组
5. ORDER BY: 结果排序
6. SELECT: 投影需要的列

- $\pi_{A1, A2, \ldots, An}(\sigma_{condition}(R_1 \times R_2 \times \ldots \times R_n))$

数据查询

- Aggregation聚集函数
  - COUNT, SUM, AVG, MAX, MIN
  - DISTINCT | ALL
- Where子句

name like ’%/%33' ESCAPE ’/’
11%33, Jone%33

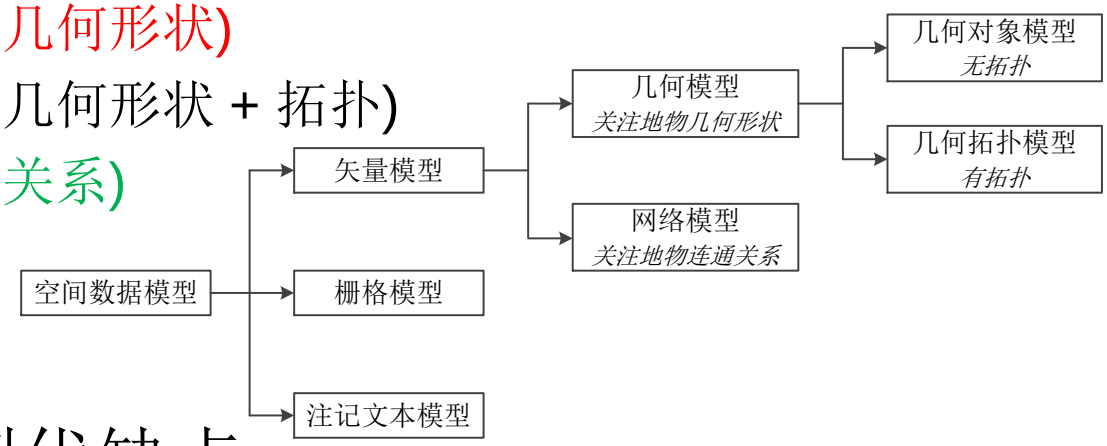
查 询 条 件	谓　词
比　较	=，>，<，>=，<=，<>
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，EXISTS，ALL, ANY, NOT …
字符匹配	LIKE，NOT LIKE, (%, _, ESCAPE)
空　值	IS NULL，IS NOT NULL
逻辑运算符	AND，OR，NOT

关系代数与SQL异同

关系代数	SQL
面向集合操作	面向multiset/bag操作, all, distinct
选择	Where 子句 (每行) / Having 子句 (每组，属性限制)
投影	Select 子句
连接	Natural Join, Inner Join [using (attributes)] (Left | Right | Full) Outer Join
除	无对应子句 (如何用SQL实现除法？)
并	union / union all
交	intersect
差	except
笛卡尔积	From子句
重命名	as
[分组]	Group by子句 (Select子句选择属性的限制)
[排序]	Order by子句, Asc, Desc
[空值]	NULL (三值逻辑，NULL = NULL的结果是NULL，Insert Rule，Selection Rule)

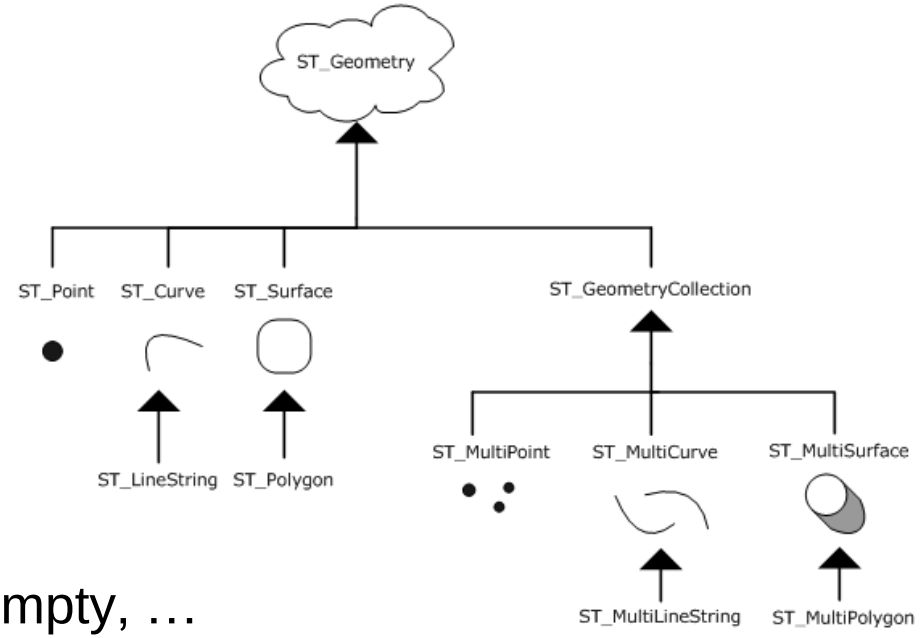
第四章 几何对象模型与查询

- 空间数据模型分类
  - 矢量模型
    - 几何对象模型 (地物几何形状)
    - 几何拓扑模型 (地物几何形状 + 拓扑)
    - 网络模型 (地物连通关系)
  - 栅格模型
  - 注记文本模型
- 矢量模型和栅格模型优缺点
- 概念模型 → 逻辑模型 → 物理模型



几何对象模型

- 对象关系数据库：数据+方法 (C++中的类)
- 几何对象层次关系
  - 坐标维数和几何维数
  - 边界、内部、外部
  - 九交矩阵
- 几何对象方法
  - 常规方法 (12种)
    - Dimension, Boundary, IsEmpty, …
  - 常规GIS分析方法 (7种)
    - Distance, Buffer, ConvexHull, Intersection, Union, Difference, SymDifference
  - 空间查询方法 (11种)
    - 包含于(within): 若$a \cap b = a$，且$\mathrm{I}(a) \cap E(b) = \emptyset$，则a包含于b内



空间拓扑关系

T = True
F = False
* = True / False, i.e., Don't care

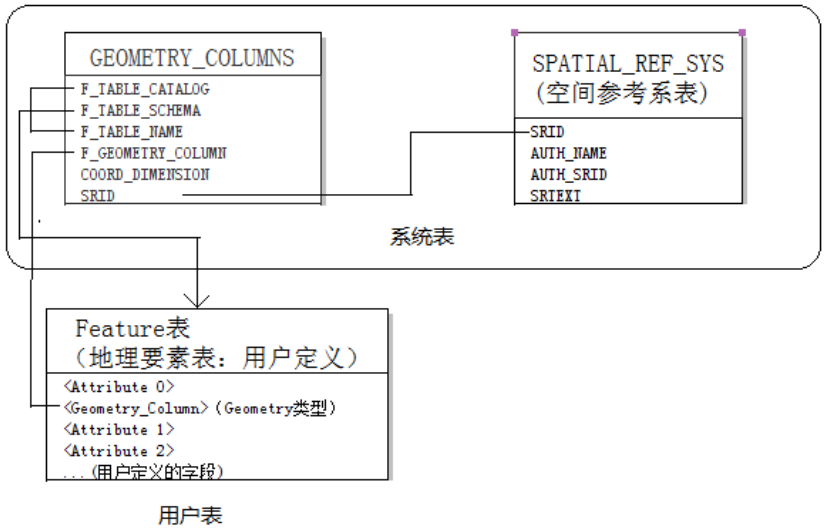
空间关系	定义	九交矩阵
Equals	$a \subseteq b$，$a \supseteq b$	TFFFTFFFT (同类：点/点, 线/线, 面/面)
Overlaps	Dim(I(a)) = Dim(I(b)) = Dim(I(a) ∩ $I(b)$)，$a \cap b \neq a$，$a \cap b \neq b$	T*T***T** (点/点, 面/面) 1*T***T** (线/线) (同类)
Disjoint	$a \cap b = \emptyset$	FF*FF**** (点线面所有组合)
Intersects	a.Intersects(b) ←→ !b.Disjoint(b)	? (点线面所有组合)
Within	$a \cap b = a$，$\mathrm{I}(a) \cap E(b) = \emptyset$	T*F**F*** (除点/点，点线面所有组合)
Contains	a.Contains(b) ←→ b.Within(a)	? (除点/点，点线面所有组合)
Touches	$\mathrm{I}(a) \cap I(b) = \emptyset$，$a \cap b \neq \emptyset$	FT*******, F**T*****, F***T**** (除点/点，点线面所有组合)
Crosses	$\mathrm{I}(a) \cap I(b) \neq \emptyset$ ， $a \cap b \neq a$，$a \cap b \neq b$	T*T****** (点/线, 点/面, 线/面) 0******** (线/线) (线/面，面/面，多点/线(面))

几何对象模型

- 空间查询方法 (11种)
  - Equals, Disjoint, Intersects, Touches, Crosses, Within, Contains, Overlaps, Relates
  - LocateAlong, LocateBetween
- 空间关系 (8种)
  - 相离(disjoint), 相交(intersects), 相等(equals)
    - a.Intersects(b) ↔ !a.Disjoint(b)
  - 交叠(overlaps), 包含于(within), 包含(contains)
    - a.Contains(b) ↔ b.Within(a)
  - 相接(touches), 穿越(crosses)

几何对象模型

- 逻辑模型
  - 基于预定义数据类型的实现
    - numeric和BLOB
  - 基于扩展几何类型的实现
    - Geometry类
- 表模式
  - 系统表
    - GEOMETRY_COLUMNS和SPATIAL_REF_SYS
  - 用户表
    - Feature和Geometry
- 物理模型
  - WKB和WKT



基于扩展Geometry类型的要素表模式

空间数据库

- 空间数据库 = 对象关系/关系数据库 + 空间扩展
  - Oracle + Oracle Spatial
  - SQL Server + SQL Server Spatial
  - PostgreSQL + PostGIS
  - MySQL + MySQL Spatial
  - SQLite + SQLite Spatialite
- PostGIS - 空间数据类型、空间函数和空间索引
  - 空间数据类型 (几何对象模型的Geometry)
    - geom geometry (Point/LineString/Polygon/Multixxx, 空间参考系)
    - ST_GeomFromText ('WKT表示', 空间参考系)
  - 空间数据分析 (几何对象模型的30个查询方法)
    - ST_XXX: ST_Distance, ST_Intersects, ST_DWithin, …
  - 空间数据索引 (PostGIS的GiST索引)

PostGIS

- WKT表示
  - POINT(10 10)
  - LINESTRING(10 10, 10 20, 20 20)
  - POLYGON((10 10, 10 20, 20 20, 20 15, 10 10))
- 单位-米
  - ST_Length(geom, false/true)
  - ST_Area(geom, false/true)
  - ST_Distance(geom1, geom2, false/true) < 10
  - ST_DWithin(geom1, geom2, 10, false/true)
- 空间数据查询 – 选择正确的函数解决问题，选择高效的函数提高效率
  - ST_Within和ST_DWithin的差异，ST_Distance和ST_DWithin的相互转换

第五章 空间扩展E/R图

1. Requirements Analysis | 2. Conceptual Design | 3. Logical, Physical, Security, etc.

- 需求分析阶段：全面了解与分析用户需求，需求分析做的是否充分与准确，决定了构建数据库系统的速度与质量
- Requirements analysis (技术人员和非技术人员)
  - What is going to be stored?
  - How is it going to be used?
  - What are we going to do with the data?
  - Who should access the data?

Database Design Process

1. Requirements Analysis | 2. Conceptual Design | 3. Logical, Physical, Security, etc.

- 概念结构设计阶段：是整个数据库设计的关键，通过对用户需求进行综合、归纳与抽象，形成一个独立于具体DBMS的概念模型
- Conceptual Design (E/R图)
  - A high-level description of the database
  - Sufficiently precise that technical people can understand it
  - But, not so precise that non-technical people can't participate

Database Design Process

1. Requirements Analysis
2. Conceptual Design
3. Logical, Physical, Security, etc.

- 逻辑结构设计阶段：将概念结构转换为某个DBMS所支持的数据模型，对其进行优化
- 物理结构设计阶段：为逻辑数据模型选取一个最适合应用环境的物理结构，使数据库的运行达到某种性能要求，如响应时间、处理频率、存储空间、维护代价等，保证数据库的安全性，如用户权限等

Database Design Process

1. Requirements Analysis
2. Conceptual Design
3. Logical, Physical, Security, etc.

- 数据库实施阶段：运用DBMS提供的数据语言、工具及宿主语言，根据逻辑设计和物理设计的结果建立数据库、编制与调试应用程序、组织数据入库、并进行试运行
- 数据库运行和维护阶段：数据库应用系统经过试运行后即可投入正式运行。在数据库系统运行过程中必须不断地对其进行评价、调整与修改

- 数据库设计过程是迭代设计过程，不断重复修正

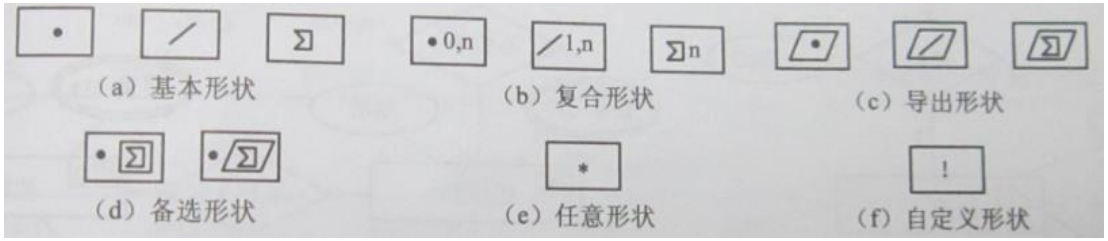
Database Design Process

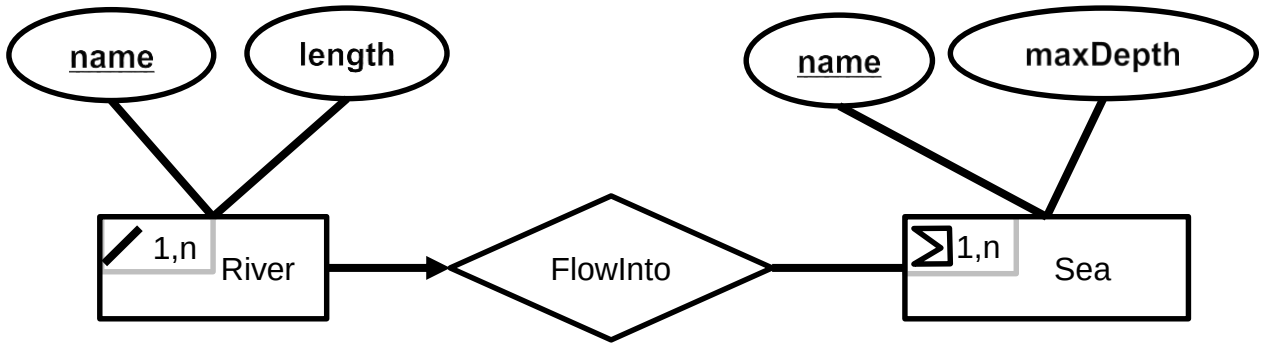
数据库设计一般要经过以下几个步骤：

- 需求分析阶段 → 数据流图和数据字典 (软工)
- 概念结构设计阶段 → E/R图或UML图
- 逻辑结构设计阶段 → 关系数据库模式 (规范化)
- 数据库物理设计阶段 → 存储方式、索引和用户权限
- 数据库实施阶段
- 数据库运行和维护阶段

E/R图

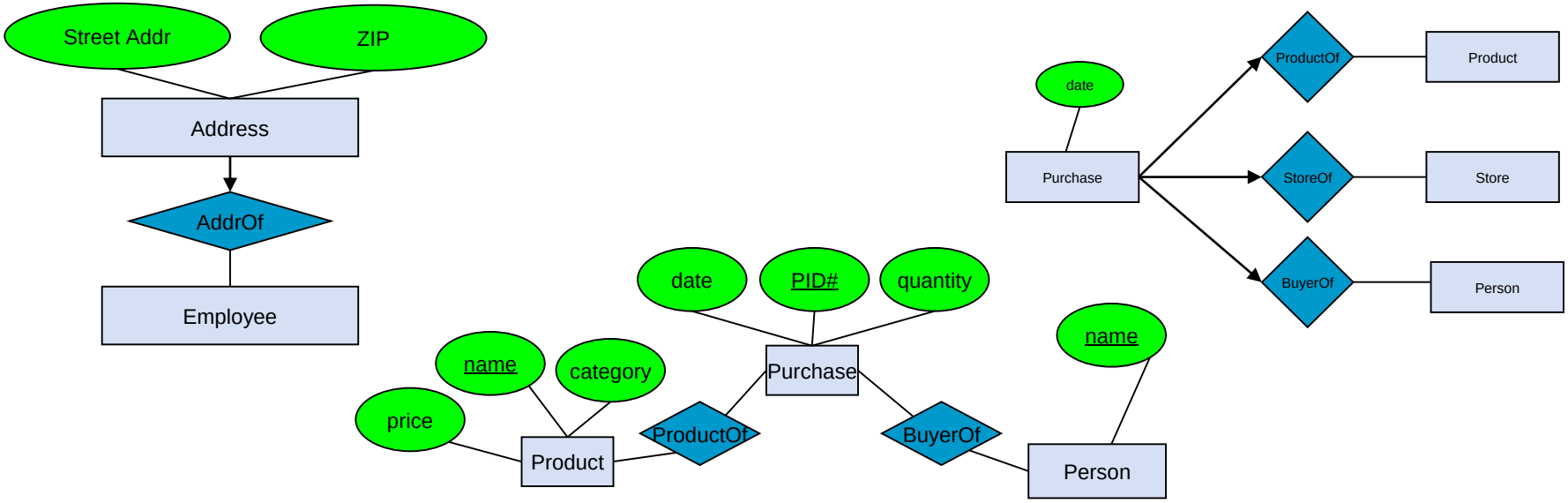
- Primitive units of E/R model
  - Entities sets: classes or types of objects
    - Attributes, Key
    - Pictograms
  - Relationships
    - A subset of cross-product
    - Attributes, Key
    - Multiplicity: 1:1, 1:N, M:N





E/R图

- Design consideration
  - Entity vs. Attributes
    - Multi-valued attributes, such as addresses, phones, and grades
  - Entity vs. Relationship
    - Multiple instances of each entity combination, such as purchase
  - Multi-way vs. New Entity + Binary relationships
    - Multiple combination, more-fine-grained constraints



从E/R图到关系模式

- Translation rules
  - Entity becomes Relation
  - Attributes become columns in the relation
  - Multi-valued attributes become a new relation
    - Includes foreign key to link to relation for the entity
  - Relationships (1:1, 1:N, N:1) become foreign keys
    - Foreign keys enforce unique constraint for 1:1
  - M:N Relationship become a relation
    - Containing foreign keys or relations from participating entities
- Constraints in E/R diagrams
  - Keys: Implicit constraints on uniqueness of entities
  - Single-value constraints (1:1, 1:N)
  - Referential integrity constraints: Referenced entities must

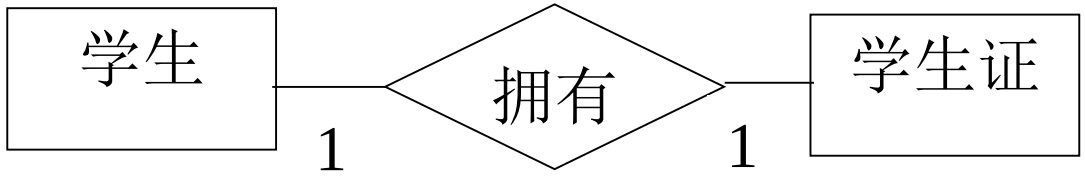
从E/R图到关系模式

- 实体 → 关系
  - 主码 → 主码
  - 属性 → 属性
  - 复合属性 → 若干原子属性
  - 多值属性 → 关系 或 转化为联系
  - 实体象形图 → Geometry类型
- 联系 (如何确定主码？)
  - 1:1 → 关系 或 与非强制性的实体合并
  - 1:n → 关系 或 与n端的实体合并
  - m:n → 关系

1:1 联系

- 方法一：
- Student (Sno, Sname, ….)
- Certificate (ID, visedate, …)
- Stu_Certificate (Sno, ID)

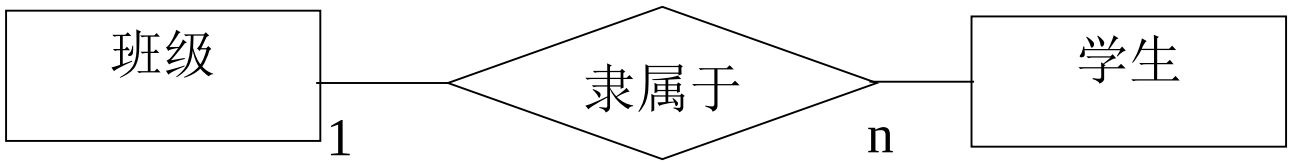
- 方法二：
- Student (Sno, Sname, …)
- Certificate (ID, visedate, …, Sno)



1:n 联系

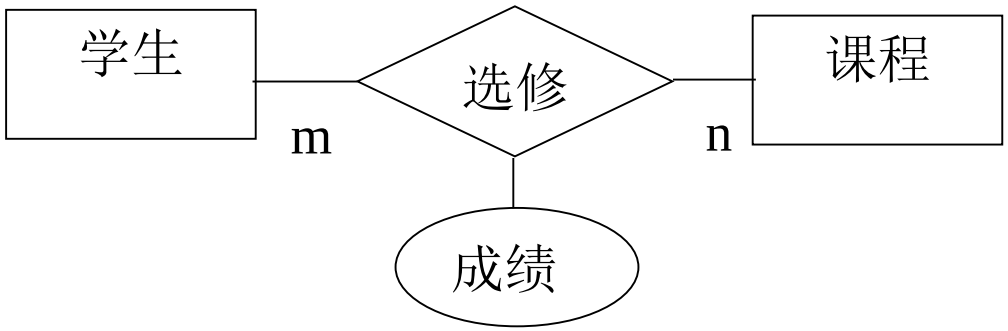
- 方法一：
- Student (Sno, Sname, ….)
- Class(ClassID, num, …)
- Class_Student (Sno, ClassID)

- 方法二：
- Student (Sno, Sname, …, ClassID)
- Class(ClassID, num, …)



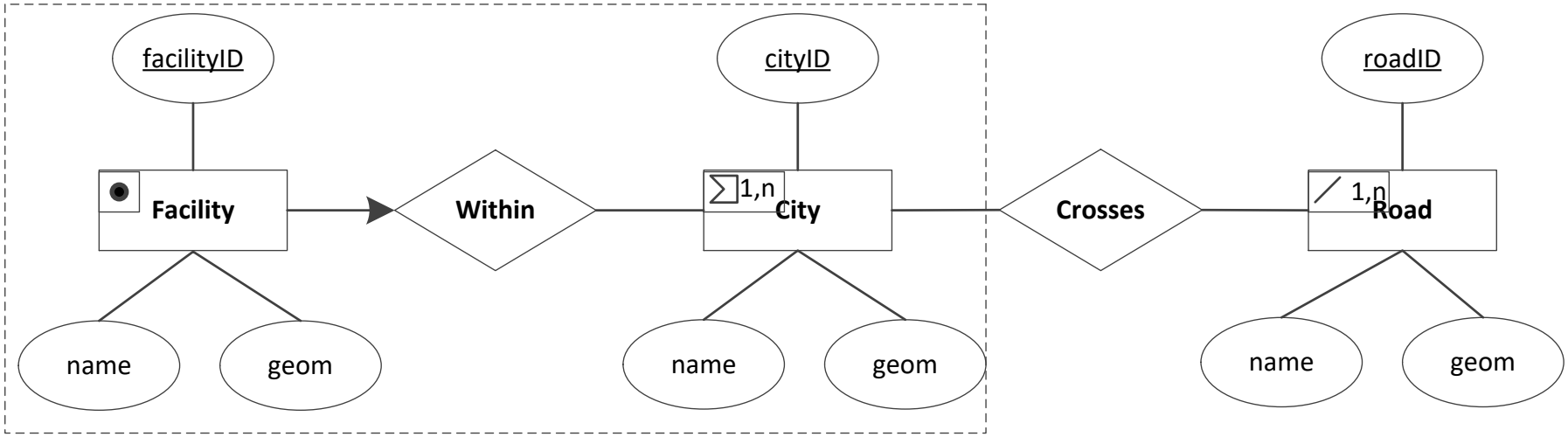
m:n 联系

- Student(sid, ...)
- Course(cid, ...)
- Enroll(sid, cid, grade, ...)



从E/R图到关系模式

- E/R图 → 关系模式



第六章 关系数据库设计理论

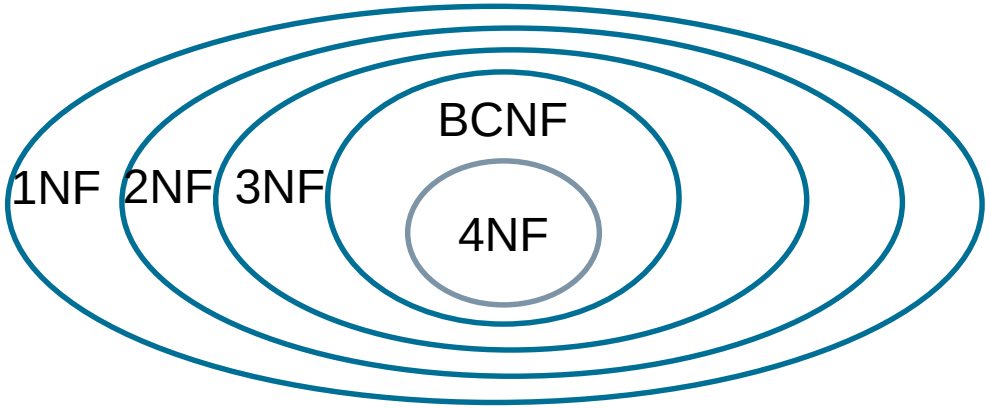
- 数据依赖
  - 关系R中，属性集A和属性集B存在函数依赖的定义：

$$\forall t, u \in R:$$
$$t[A_1, \dots A_n] = u[A_1, \dots A_n] => \ t[B_1, \dots B_m] = u[B_1, \dots B_m]$$

    - 计算属性的闭包判断是否为码(key)
    - 完全非平凡函数依赖
    - 函数依赖规则：splitting, combining, trivial-dependency, transitive
    - 指定关系的函数依赖集合：Minimal set of completely nontrivial FDs such that all FDs that hold on the relation follow from the dependencies in the set

数据依赖→设计异常→关系分解

- 数据依赖 → 设计异常
  - 数据冗余
  - 更新、插入、删除异常



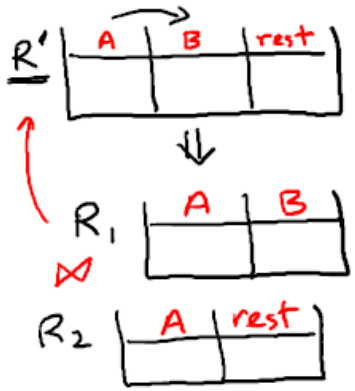
- 数据依赖 → 设计异常 → 关系分解
  - 要求：no anomalies, no lost information (lossless join property)
  - BC范式：Relation R with FDs is in BCNF if for each A→B, A is a key
  - 第4范式：Relation R with MVDs is in 4NF if for each nontrivial A→→B, A is a key
  - 设计考虑：保留函数依赖，考虑查询效率，避免过度分解
  - 空间数据的独特性：shape → other attributes(大部分情况)

BCNF分解

BCNF decomposition algorithm

Input: relation R + FDs for R

Output: decomposition of R into BCNF relations with "lossless join"



Compute keys for R (using FDs)

Repeat until all relations are in BCNF:

- Pick any R' with $A \rightarrow B$ that violates BCNF
- Decompose R' into $R_1(\bar{A}, \bar{B})$ and $R_2(\bar{A}, rest)$
- Compute FDs for R_1 and R_2
- Compute keys for R_1 and R_2

第七章 空间存储与索引

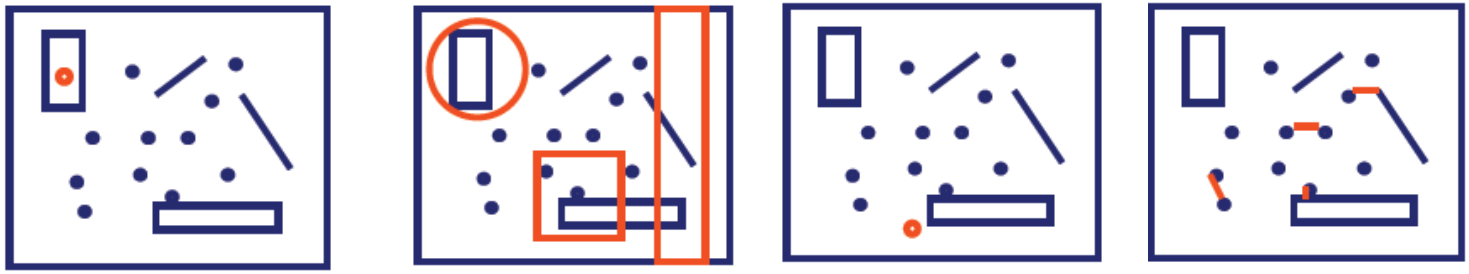
- Physical data models in relational database
  - Sequential scan is much faster than random reads (2%)
  - Total cost = I/O cost + CPU cost
  - Data file: heap (unordered), ordered, hashed, clustered
  - Data file vs. index file
  - Indices: B-Tree, Hash table, …
  - Primary key vs. index key, clustered index vs. unclustered index
- Query plan: Nested loop join / Merge join / Hash join
  - 查询规划：explain / explain analysis

空间扩展方法

- Two approaches to support spatial data and queries
  - Reusing relational physical data model concepts (重用关系数据库物理模型)
    - Space filling curves define a total order for points, such as Z-Curve, Hilbert Curve
    - This total order helps in using ordered files, search trees
  - New spatial techniques (新的空间技术)
    - Spatial indices, such as grids, quadtree, R-Tree
    - Provide better computational performance

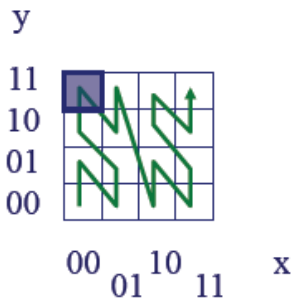
空间查询

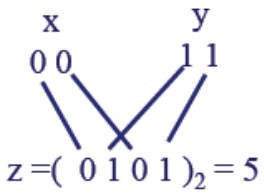
- Common Queries
  - Point query: Find all rectangles containing a given point
  - Range query: Find all objects within a query rectangle
  - Nearest neighbor: Find the point closest to a query point
  - Intersection query: Find all the rectangles intersecting a query rectangle (spatial join)



空间填充曲线

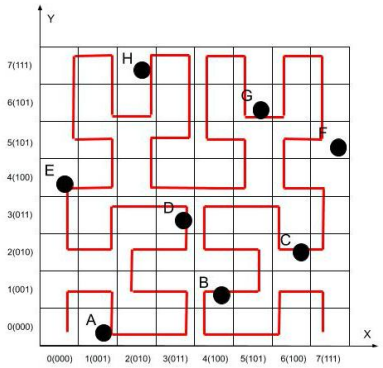
- Space-filling Curves (重用关系数据库的物理模型)
  - Z-Curve
  - Hilbert Curve
  - Sorted file + B+tree



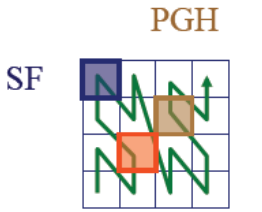


How about the reverse:

$(x,y) = g(z)$?



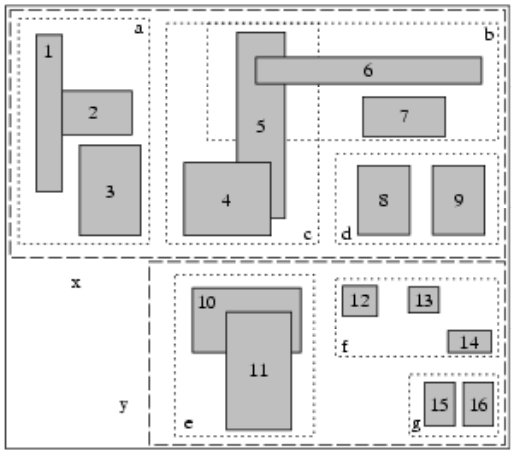
  - Point query
    - Calculate z/h value, search B+tree
  - Range query
    - Calculate z/h value, search the smallest value in B+tree, traverse
  - NN
    - Calculate z/h value, search B-tree, traverse, range query
  - Spatial join
    - Merge the list of (sorted) z/h-values, looking for the prefix property

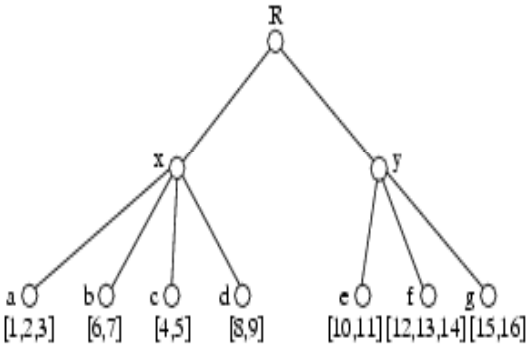


z	cname	etc
5	SF	
12	PGH	

空间索引

- R-Tree
  - Properties
    - Node overlapping
  - Insertion/Deletion
    - Increase in area
  - Point query
    - Traverse R-Tree (multiple children nodes)
  - Range query
    - Traverse R-Tree (multiple children nodes)
  - NN
    - Depth-first search, calculate the range, range query
  - Spatial join
    - Nested loop join, traverse R-Tree





索引创建

- 创建索引
  - Create [unique][clustered] Index <索引名> On <表名>(<属性列名>)
    - 具体数据结构取决于数据库系统实现
    - 如果了解数据属性特点，可以在创建索引时指定数据结构
  - Create Index <索引名> On <表名> Using GiST (<几何列名>)
    - 能对非空间数据属性构建空间索引？
- 空间索引只能创建在空间数据，尽可能创建在静态空间数据，基于几何要素的包围盒Envelope通过overlap判断，获得需要真正几何操作的候选集(filter step)
  - 能够利用空间索引的函数：Equals, Intersects, Touches, Crosses, Within, Contains, Overlaps, DWithin
    - 两个几何关系判断，仅需返回True/False，可以利用索引快速判断获得False，如果Envelope不相交，那两个几何也肯定不相交，返回False

索引选择

- 如何选择属性索引？
  - 看join，where，group by，order by使用的属性，根据属性的特点(唯一或非唯一)和用法(=，>或<，<>)创建索引
  - 当同一个关系使用多个属性的等值判断，可以创建多属性索引加速
- 空间函数和空间索引 – 减少不必要的两两空间处理
  - where子句调用空间函数时，可能会使用空间索引
    - 是否真的调用，取决于查询规划器得到的全表扫描和使用索引扫描的cost值
  - select子句调用空间函数时，通常不会使用空间索引
    - select子句中所有空间操作都是必要的
  - 索引是建在基表上，通过不会在查询结果上使用空间索引

查看空间索引

- 如何查看关系中已创建的索引？
  - PostgreSQL通常基于主键以sorted file进行存储，即主键本身是primary index
  - PostGIS shapefile loader缺省选项是对geometry创建空间索引
- 学会看查询规划，DBMS基于cost模型(与数据和SQL语句有关)预测使用或不使用索引所需时间，选择cost最小的查询规划进行真正的数据查询，由于是近似的cost，查询时间不一定最短

第八章 空间查询处理与优化

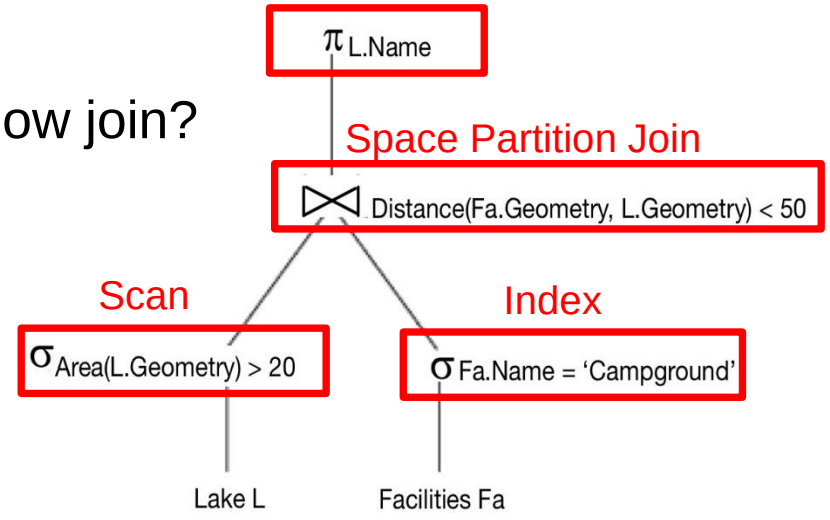
- 查询处理与优化步骤
  - 分析 → 检查 → 优化 (等价的关系代数表达式) → 执行
- 关键概念
  - Building blocks (操作类型)
    - Decompose SQL query into building blocks, e.g., select, join, sort
  - Strategies for each building block (每类操作的实现方式)
    - A few processing strategies for each building block
  - Query optimization (等价优化)
    - A fixed priority scheme, a simple cost model
- 空间数据库的QPO挑战
  - Building blocks: spatial select, spatial join, NN
  - Strategies: limited choice
  - Query optimization: both CPU and I/O intensive

空间查询

- 空间查询操作一般分为过滤和精炼两步
  - 空间索引用于过滤步，获得满足查询条件的对象候选集
  - 近似空间数据类型：MBR，近似空间操作：Overlap
- List of building blocks
  - Point Query
    - Linear search, binary search with Z-Curve, Indexed search with R-Tree
  - Range Query
    - Linear search, binary search (+scan) with Z-Curve, Indexed search with R-Tree
  - Nearest Neighbor
    - Two phase approach, single phase approach
  - Spatial Join
    - Nested loop, nested loop with one spatial index, space partitioning, tree matching

空间查询优化

- QPO process steps include
  - Creation of a query tree for the SQL query
  - Choice of strategies to process each node in query tree
  - Ordering the nodes for execution
- Key ideas for SDBMS include
  - Filter-Refine paradigm to reduce complexity
  - New building blocks and strategies for spatial queries
  - CPU cost is higher
    - Push down spatial selection below join?
    - Reorder join operations



代价模型

- $\sigma_{A=a}(R)$ (1 data block)
  - 返回的行数估计为T(R) * 1 / V(R, A)
  - Heap file，无索引：B(R)
  - 聚集索引(隐含sorted file)： B(R) * 1 / V(R, A)
  - 非聚集索引：T(R) * 1 / V(R, A)
- R ⋈ S (nested loop join, 2 data blocks)
  - 无索引
    - B(R) + B(R) * B(S); B(S) + B(S) * B(R)
  - 聚集索引
    - B(R) + T(R) * B(S) * 1 / V(S, B); B(S) + T(S) * B(R) * 1 / V(R, B)
  - 非聚集索引
    - B(R) + T(R) * T(S) * 1 / V(S, B); B(S) + T(S) * T(R) * 1 / V(R, B)

第九章 空间网络模型

- Location Based Services
  - Location: Where am I ?
  - Directory: What is around me?
  - Routes: How do I get there?
- Spatial Network Analysis
  - Route (A start-point, Destination(s))
  - Allocation (A set of service centers, A set of customers)
  - Site Selection (A set of customers, Number of new service centers)
- Spatial Network Examples
  - Road, Railway, River

Spatial Network Query Example

- River (Connect By)
  - List all direct and indirect tributaries of Mississippi river
  - List the direct tributaries of Colorado
  - Which rivers could be affected if there is a spill in North Platte river
- Railway (With Recursive)
  - List all stops which can be reached from A
  - List the routes numbers that connect A and B
- Road (pgRouting)
  - Find shortest path from a start-point to a destination
  - Find nearest hospital by driving distance
  - Find shortest route to deliver packages to a set of homes
  - Allocate customers to nearest service center

Data Models of Spatial Networks

- Conceptual Model
  - Information Model: Entity Relationship Diagrams
  - Mathematical Model: Graphs
    - Road (turns), River, Railway
- Logical Data Model
  - Abstract Data types - Transitive Closure
  - Custom Statements in SQL
    - SQL2 - CONNECT clause in SELECT statement
    - SQL3 - WITH RECURSIVE statement
- Physical Data Model
  - Storage-Structures, File-Structures
    - Adjacency matrix, adjacency list, normalized/denormalized tables
    - Minimize CRR
  - Algorithms for common operations

Query Processing for Spatial Networks

- Main memory
  - Connectivity: Breadth first search, depth first search
  - Shortest path: Dijkstra's algorithm, A*
- Disk-based
  - Connectivity strategies are in SQL3
  - Shortest path - Hierarchical routing algorithm
- pgRouting
  - pgr_createTopoloty
  - pgr_analyzeGraph
  - pgr_nodeNetwork
  - pgr_dijkstra(text edges_sql, bigint start_vid, bigint end_vid, boolean directed:=true)

第十章 数据库安全性与完整性

- 存取控制
  - 仅让用户看到或修改他们有权看到或修改的数据
  - Grant privs On R to users [With Grant Option]
  - Revoke privs On R From users [Cascade | Restrict]
  - Grant diagram
- 数据完整性 – 静态
  - Primary Key, Foreign Key, Unique, NOT NULL, Check
  - 如何定义和修改，何时做完整性检查
- 触发器 – 动态
  - When *event* occurs, check *condition*; if true, do *action*

数据完整性

- 数据完整性
  - Data-entry errors (inserts)
  - Correctness criteria (updates)
  - Enforce consistency
  - Tell system about data (store, query processing)
- 触发器
  - Move logic from applications to DBMS
  - To enforce constraints
    - Expressiveness
    - Constraint “repair” logic

触发器

- 触发器SQL Standard写法

  Create Trigger name

  Before | After | Instead Of events

  [referencing-variables]

  [For Each Row]

  When (condition)

  action

- 用Instead of触发器实现视图用户自定义修改

视图

- 视图作用
  - Hide some data from some users
  - Make some query easier / more natural
  - Modularity of database access
- Materialized Views
  - 除View的作用外，与索引类似，提高查询效率
- SQL Standard中updatable views的定义
  - Select (no Distinct) on single table T
  - Attributes not in view can be 'NULL' or have default value
  - Subqueries must not refer to T
  - No Group by or aggregation

第十一章 PostgreSQL服务器编程

- PostgreSQL扩展
  - 三层结构，PL/pgSQL
  - 自定义类型和操作符，定制排序方法和索引
- 函数
  - 函数结构
  - 条件表达式
  - 循环表达式
  - 返回集合
  - 错误处理与异常
  - 几何函数应用举例

```
create or replace function ST_NInteriorRings(geom geometry)
   returns integer
as $$
declare num integer = 0;
begin
   if ST_GeometryType(geom) = 'ST_Polygon' then
      num = ST_NumInteriorRings(geom);
   elsif ST_Geometype(geom) = 'ST_MultiPolygon' then
      for i in 1..ST_NumGeometries(geom) loop
         num = num + ST_NumInteriorRings(ST_GeometryN(geom, i));
      end loop;
   end if;
   return num;
end;
$$ language plpgsql;
```

PostgreSQL触发器

- 适用场景
  - 审计、日志、执行复杂约束、复制等
- 应用程序逻辑仅可能避免使用触发器
- INSERT, UPDATE, DELETE, TRUNCATE
  - FOR EACH ROW, OLD和NEW
  - FOR EACH STATEMENT, OLD和NEW未分配
  - TRUNCATE只能用于FOR EACH STATEMENT
  - TRUNCATE不会触发DELETE，抛出异常终止事务
  - 表执行BEFORE或AFTER操作
    - 对于BEFORE, INSERT, UPDATE, DELETE返回NULL处理
    - 对于AFTER, INSERT, UPDATE, DELETE抛出异常终止事务
  - 视图执行INSTEAD OF操作

第十二章 事务处理

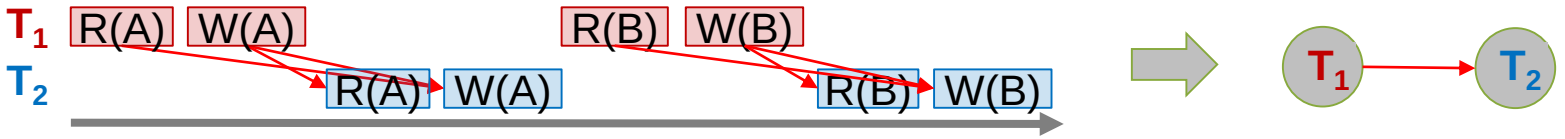
- A transaction (“TXN”) is a sequence of one or more *operations* (reads or writes) which reflects *a single real-world transition*
- Grouping user actions (reads & writes) into transactions helps with two goals:
  - Recovery & Durability: Keeping the DBMS data consistent and durable in the face of crashes, aborts, system shutdowns, etc.
  - Concurrency: Achieving better performance by parallelizing TXNs *without* creating anomalies
- Properties of Transactions
  - 原子性 (Atomicity)，一致性 (Consistency)，隔离性 (Isolation)，持久性 (Durability)

同步与调度

- Concurrency: Isolation & Consistency
- A **serial schedule** (串行调度) is one that does not interleave the actions of different transactions
- A **serializable schedule** (可串行化调度) is a schedule that is equivalent to ***some*** serial execution of the transactions
- Two actions **conflict** (冲突) if they are part of different TXNs, involve the same variable, and at least one of them is a write
  - Read-Write conflicts (RW)
  - Write-Read conflicts (WR)
  - Write-Write conflicts (WW)

同步与调度

- Anomalies with interleaved execution
  - Unrepeatable read
  - Dirty read / Reading uncommited data
  - Inconsistent read / Reading partial commits
  - Partially-lost update
- Schedule S is conflict serializable (冲突可串行化) if S is conflict equivalent to some serial schedule
  - Conflict graph

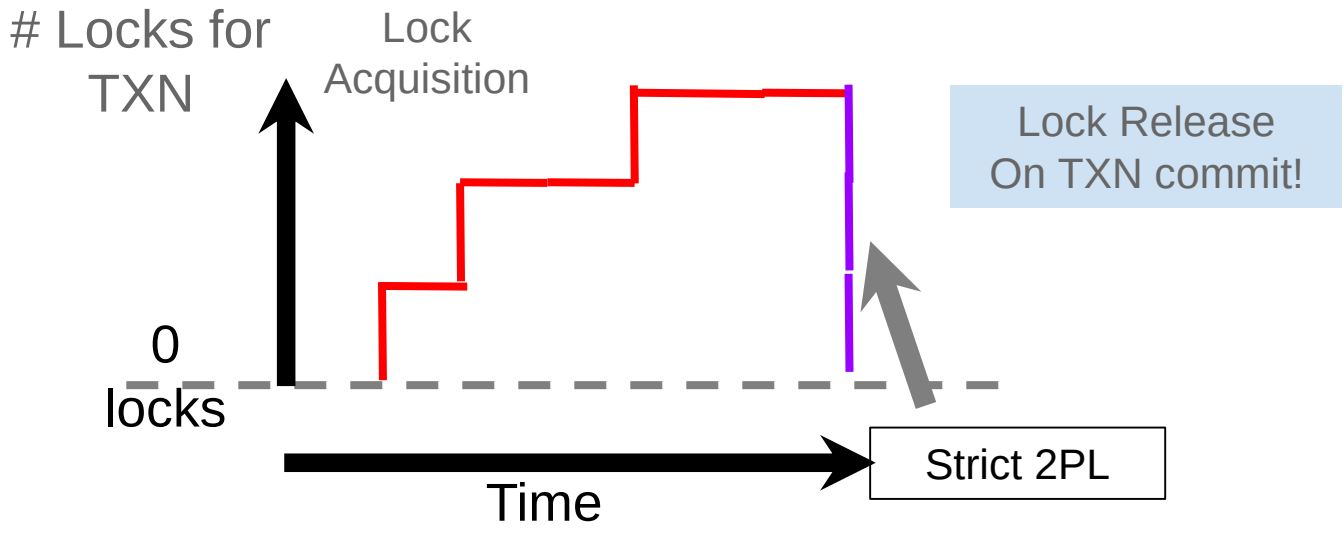


Two-Phase Locking

- Locking Scheduler vs. Multiversion Concurrency Control
  - An X (exclusive) lock on object before writing
  - An S (shared) lock on object before reading
- Fine granularity locking (e.g., tuples)
  - High concurrency, high overhead in managing locks
  - E.g., SQL Server
- Coarse grain locking (e.g., tables, entire database)
  - Many false conflicts, less overhead in managing locks
  - E.g., SQLite

T_1 \ T_2	X	S	
X	N	N	Y
S	N	Y	Y
	Y	Y	Y

Two-Phase Locking

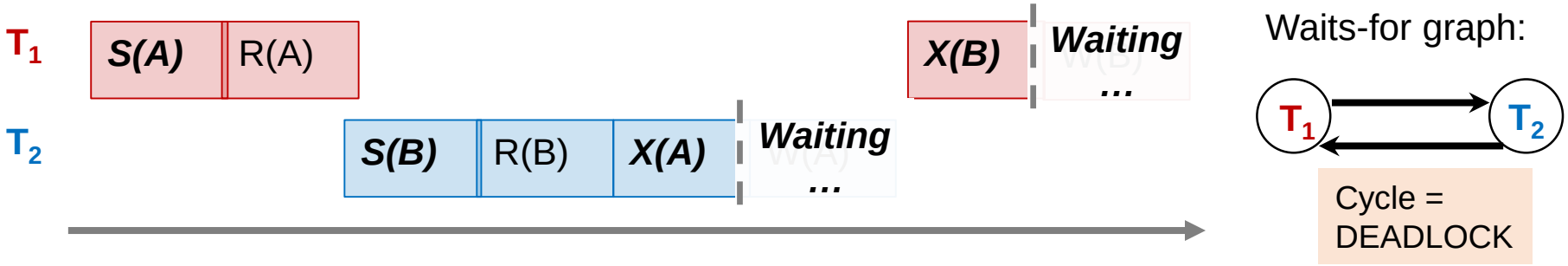


2PL: A transaction can not request additional locks once it releases any locks. Thus, there is a “growing phase” followed by a “shrinking phase”

Strict 2PL: Release locks only at COMMIT (COMMIT Record flushed) or ABORT

Two-Phase Locking

- **Deadlock**: Cycle of transactions waiting for locks to be released by each other
  - Waits-for graph



- Major differences between vendors
  - Lock on the entire database
    - SQLite
      - ➢ There is only one exclusive lock, thus, never deadlocks
  - Lock on individual records
    - SQL Server, DB2, etc
      - ➢ Checks periodically for deadlocks and aborts on TXN

Isolation Levels

	Dirty Reads (读尚未committed的数据)	Nonrepeatable reads (一个事务中对数据的两次读取数值不相同)	Phantoms (事务中能对关系增加新的元组)
Read Uncommitted	Y	Y	Y
Read Committed	N	Y	Y
Repeatable Read	N	N	Y
Serializable	N	N	N

- Read Uncommitted: 读数据前不对数据加S锁
- Read Committed: 读之前加S锁，读完释放
- Repeatable Read: 读之前加S锁，事务结束释放
- Serializable: 严格按照两段封锁协议对数据加锁

Isolation Levels

- Read Uncommitted
  - “Long duration” WRITE locks (Strict 2PL)
  - No READ locks (Read-only transactions are never delayed)
- Read Committed
  - “Long duration” WRITE locks (Strict 2PL)
  - “Short duration” READ locks (Only acquire lock while reading (not 2PL))
- Repeatable Read
  - “Lodng duration” WRITE locks (Strict 2PL)
  - “Long duration” READ locks (Strict 2PL)
- Serializable
  - “Long duration” WRITE locks (Strict 2PL)
  - “Long duration” READ locks (Strict 2PL)
  - Predicate locking (to deal with phantoms)

Write-Ahead Logging (WAL)

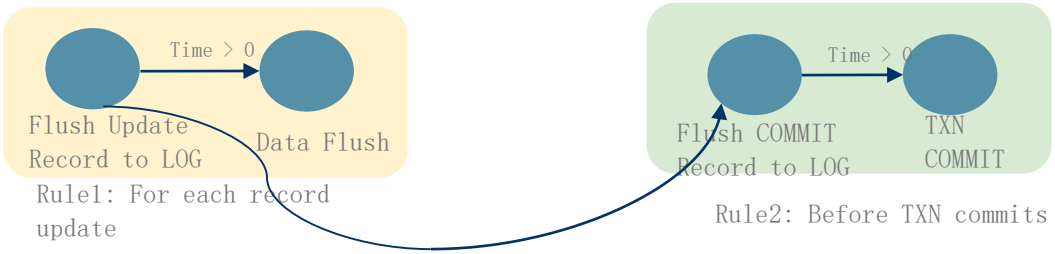
Algorithm: WAL

For each record update, write Update Record into LOG

Follow two Flush rules for LOG

- Rule1: Flush Update Record into LOG before corresponding data page goes to storage → **Durability**
- Rule2: Before TXN commits, → **Atomicity**
  - Flush all Update Records to LOG
  - Flush COMMIT Record to LOG

Transaction is committed *once COMMIT record is on stable storage*



OLTP和OLAP比较

- OLTP – Online Transaction Processing
  - Short transactions
  - Simple queries
  - Touch small portions of data
  - Frequent updates
- OLAP – Online Analytical Processing
  - Star schemas
  - Data cubes
  - With Cube and With Rollup (类似materialized view)
  - Special indexes and query processing technique

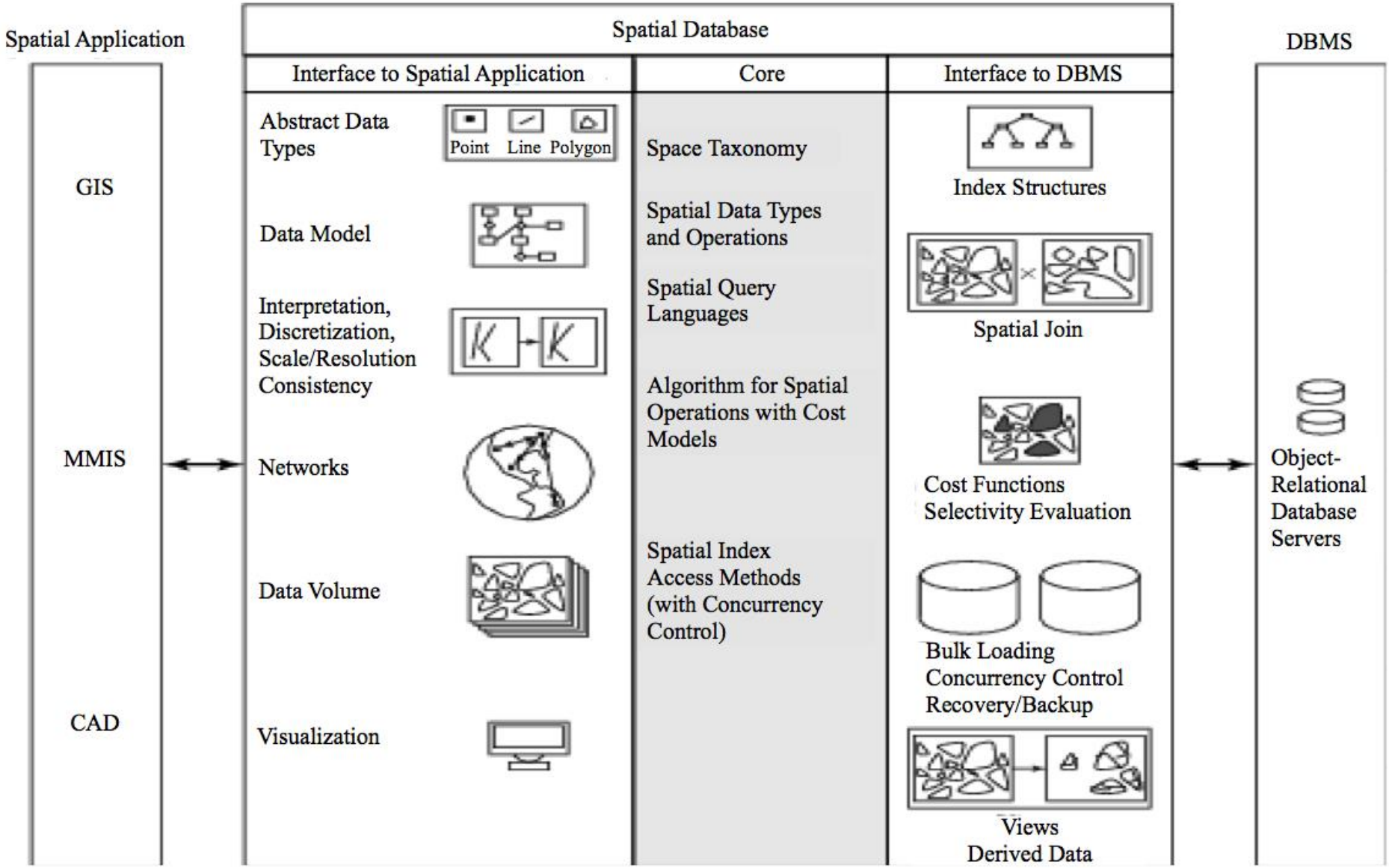
空间数据模型

- 空间数据模型
  - 几何对象模型 (PostGIS)
    - 基于预定义数据类型的实现 (numeric或BLOB)
    - 基于扩展几何类型的实现 (Geometry类)
  - 几何拓扑模型 (几何对象模型+空间网络模型)
  - 空间网络模型 (With Recursive, pgRouting)
    - 空间网络模型构建createTopology, analyzeGraph, nodeNetwork
    - 连通性分析connect by, with recursive, 最短路径分析pgr_dijkstra
  - 栅格模型 (课程：遥感数字图像处理)
  - 注记文本模型
    - 注记标签、注记文本、注记尺寸
    - 基于预定义数据类型和扩展几何类型的实现
- PostGIS的空间数据类型、GIS分析和索引
- pgRouting的空间网络构建与最短路径查询

课程内容：空间数据库设计

- 地理空间数据库概念，关系代数和SQL　第1-4周
- 几何对象模型与查询　第4-5周
- 空间网络模型与查询　第11-12周
- 空间数据库设计
  - 需求分析 (几何，属性，行为)
  - 概念设计 (扩展的E/R图)　第6-7周
  - 逻辑设计 (将扩展E/R图转换为关系，关系优化)
  - 物理设计 (空间数据库，存储方式，建立索引)　第8-10周
  - 实施　(建表，导入数据，SQL编程，试运行)
  - 运行维护 (转储和恢复，安全性和完整性控制)
    (性能监督、分析改进、重组与重构)　第10,12-14周
- OLAP / NoSQL

课程内容：空间数据库设计



课程内容：Location Based Services

- Location: Where am I?
  - Geo-code: Place Name → <latitude, longitude>
  - Reverse Geo-code: <latitude, longitude> → Place Name
- Directory: What is around me?
  - Where is the nearest Taxi? List all Banks within 1 mile
  - 几何对象模型：空间数据类型、空间关系判断和空间索引
  - 空间数据库设计：空间扩展E/R图、空间属性的函数依赖
  - 空间数据分析：空间索引与空间查询优化、OLAP
- Routes: How do I get there?
  - What is the shortest path to get there?
  - 空间网络模型：图模型、连通性、最短路径
  - 几何对象模型如何转换成空间网络模型 (只需了解基本概念)
  - PostgreSQL的空间函数和触发器编程

空间数据库作业

- 作业1：关系代数，公共自行车服务关系数据库构建、数据查询与分析 (SQL应用，最大/小值问题，分组求最大/小值问题)
- 作业2：美国湖泊、高速公路、城市和交通事故空间关系数据库构建和空间查询 (几何对象模型应用，空间关联，时空数据分析)
- 作业3：美食外卖空间数据库设计、数据库逆向设计和数据ETL (空间数据库设计与应用，空间扩展E/R图，函数依赖，BCNF)
- 作业4：代价模型、查询处理与优化、空间填充曲线与GiST空间索引 (几何对象模型的空间索引原理和应用，索引创建，查询计划)
- 作业5：美国航空、深圳地铁、杭州道路的空间网络连通性和最短导航路径查询，视图与触发器 (网络模型应用，可更新视图，instead of触发器)
- 作业6：基于位置服务的空间数据库设计 (空间数据库设计与应用)
- 作业6：基于纽约道路和自行车站点的四叉树与R-树空间索引创建与查询 (几何对象模型的空间数据类型与索引C++实现)

- 关系数据库
- 几何对象模型应用
- 空间网络模型应用
- 空间数据库设计

期末考试

- 基本概念与原理
  - 数据模型
  - 关系代数与SQL语句
  - 完整性约束
  - 空间数据管理技术
  - 空间数据库标准
  - 空间数据库设计
  - 空间数据模型与空间函数 (PostGIS)
  - 网络连通性与最短路径 (pgRouting)
  - 空间查询处理与优化
  - OLTP

期末考试

- 实际应用 (关键：基于做过的练习，举一反三)
  - SQL
  - 空间扩展E/R图，关系转换与规范化
  - 关系创建，完整性约束
  - 空间数据插入、更新、删除
  - 空间函数，空间几何查询和网络查询
  - 用户权限、视图、索引
  - PostgreSQL函数与触发器

- 需要记住30个常用的PostGIS空间函数和pgRouting最短距离查询函数
  - 考试可能需要使用其他给出说明的空间函数

期末考试

- 复习内容
  - 课件
  - 测试
  - 作业和练习
    - 我们已经做了海(河流，湖泊，船)，陆(国家，城市，道路，地铁，出租车、自行车)，空(机场，航班)各类应用
    - 区分哪些是静态空间数据，哪些是动态空间数据
- 不考察内容
  - Jupyter Notebook, QGIS, OpenStreetMap, Python, C++等作业内容

期末考试

- 简答题举例
  - 空间数据管理技术的发展阶段
  - 空间数据库的设计流程
  - 空间数据特点，是否能作为关系的主码
  - R-Tree索引的区域查询与NNQuery查询
  - 空间查询处理，空间索引在其中的作用
  - 空间数据库与关系数据库的异同
    - 数据模型，数据库设计，索引，查询优化，……
  - 学完地理空间数据库课程后，对空间数据和空间计算有哪些新的体会和认识
  - 美国航天局(NASA)在2015年9月28日宣布，在火星表面发现了有液态水活动的“强有力”证据。为进一步探索火星，计划构建基于PostGIS的火星信息系统。从空间数据类型、空间分析和空间索引三个方面，分析PostGIS哪些功能不用修改，哪些功能需要修改。为进一步探索火星，在改造后的PostGIS中，构建哪些关系存储已有或未来计划采集的火星表面信息，为采集这些信息，需要哪些基础设施？

Thank You!

- 期末闭卷考试
  - 1月10号 10:30-12:30
  - 可带一张A4纸，占最终考评的50%
  - 平时成绩 = Min((作业成绩 + 测试成绩 + 附加题) / 4, 100)
  - 最终成绩 = (平时成绩 + 期末成绩) / 2
- 预祝考试顺利！