

1、举例说明什么是空间数据、非空间数据？如何理解空间查询和非空间查询的区别？常用的空间数据库管

理方式有哪几种及其各自特点。

数据：是指客观事务的属性、数量、位置及其相互关系等的符号描述。空间数据：是对现实世界中空间对象（事物）的描述，其实质是指以地球表面空间位置为参照，用来描述空间实体的位置、形状、大小及其分布特征等诸多方面信息的数据。河流的泛洪区，卫星影像数据、气象气候数据等都可以是空间数据

店名称 店员人数，去年的销售量，电话号码等是非空间数据

空间查询是对空间数据的查询或命令

人工管理阶段

文件管理阶段 缺点：

1) 程序依赖于数据文件的存储结构，数据文件修改时，应用程序也随之改变。

2) 以文件形式共享，当多个程序共享一数据文件时，文件的修改，需得到所有应用的许可。不能达到真正的共享，即数据项、记录项的共享。

常用：

文件与数据库系统混合管理阶段 优点：由于一部分建立在标准的 **RDBMS** 上，存储和检索数据比较有效、可靠。

缺点：1) 由于使用了两个子系统，它们各自有自己的规则，查询操作难以优化，存储在 **RDBMS** 外的数据有时会丢失数据项的语义。

2) 数据完整性的约束条件可能遭破坏，如在几何空间数据系统中目标实体仍存在，但在 **RDBMS** 中却已删除。

3) 几何数据采用图形文件管理，功能较弱，特别是在数据的安全性、一致性、完整性、并发控制方面，比商用数据库要逊色得多

全关系型空间数据库管理系统

属性数据、几何数据同时采用关系式数据库进行管理

空间数据和属性数据不必进行烦琐的连接，数据存取较快

属性间接存取，效率比 **DBMS** 的直接存取慢，特别是涉及空间查询、对象嵌套等复杂的空间操作

**GIS** 软件：**System9**，**Small World**、**GeoView** 等

本质：**GIS** 软件商在标准 **DBMS** 顶层开发一个能容纳、管理空间数据的系统功能。

对象关系数据库管理系统

优点：在核心 **DBMS** 中进行数据类型的直接操作很方便、有效，并且用户还可以开发自己的空间存取算法。

缺点：用户须在 **DBMS** 环境中实施自己的数据类型，对有些应用相当困难。

面向对象的数据库系统。

采用面向对象方法建立的数据库系统；

对问题领域进行自然的分割，以更接近人类通常思维的方式建立问题领域的模型。

目前面向对象数据库管理系统还不够成熟，价格昂贵，在空间数据管理领域还不太适用；

基于对象关系的空间数据库管理系统可能成为空间数据管理的主流

2、什么是 GIS，什么是 SDBMS？请阐述二者的区别和联系。

**GIS** 是一个利用空间分析功能进行可视化和空间数据分析的软件。它的主要功能有：搜索、定位分析、地形分析、流分析、分布、空间分析 / 统计、度量 **GIS** 可以利用 **SDBMS** 来存储、搜索、查询、分享大量的空间数据集

改：地理信息系统 是以地理空间数据库为基础，在计算机软硬件的支持下，运用系统工程和信息科学的理论，科学管理和综合分析具有空间内涵的地理数据，以提供管理、决策等所需信息的技术系统。简单的说，地理信息系统就是综合处理和分析地理空间数据的一种技术系统。

2、**SDBMS** 是一个软件模块。它可以           、利用一个底层的数据库管理系统           、支持多种空间数据模型、相应的空间抽象数据类型（**ADT**）以及一种能够调用这些 **ADT** 的查询语言           、支持空间索引、高效的空  
间操作算法以及用于查询优化的特定领域规则           3、区别与联系：           、利用 **GIS** 可以对某些对象和图层进行  
操作，而利用 **SDBMS** 则可以对更多的对象集和图层进行更加简单的操作           、**SDBMS** 可以  
在 **GIS** 不能使用的某些领域进行使用，例如基因组学、天文学、多媒体信息系统等           、**GIS** 可  
以作为 **SDBMS** 的前端，利用一个高效的 **SDBMS** 可以大大提高 **GIS** 的效率和生产率。

改：联系：**GIS** 可作为 **SDBMS** 的前端工具，一个高效的空间数据库系统是实现 **GIS** 高效查询和分析的前  
提条件。

区别：**GIS** 和 **SDBMS** 的主要不同侧重点：

**GIS** 是一个侧重于空间数据可视化和分析的软件，           **GIS** 常用分析功能：

搜索	专题搜索、按区域搜索
定位分析	缓冲区、叠置分析
地形分析	坡度/坡向、排水网系
流分析	连接性、最短路径
分布	变化检测、接近、最近邻接
空间分布/统计	自相关、相似性检索、拓朴
度量	距离、周长、形状、方向

**GIS** 使用 **SDBMS** 存储、检索、查询、共享大型空间数据集

**SDBMS** 重点关注：

高效存储、查询和共享大型空间数据集

提供尽量简单的查询方法

通过空间索引和查询优化方法加快大型空间数据集的查询反应时间

**SDBMS** 有可能用于非 **GIS** 领域的其它方面：如天文、气象、生物等

3、用传统数据库系统管理空间数据，存在哪些局限？

只支持简单的数据类型，如：数字、字符串、日期。实现上述的多段线表达非常复杂

答：(1)传统数据库系统管理的是不连续的、相关性较小的数字和字符；而地理信息数据是连续的，并  
且具有很强的空间相关性。

(2)传统数据库系统管理的实体类型较少，并且实体类型之间通常只有简单、固定的空间关系；而地理  
空间数据的实体类型繁多，实体类型之间存在着复杂的空间关系，并且还能产生新的关系           (如拓扑关系)。

(3)传统数据库系统存贮的数据通常为等长记录的数据；而地理空间数据通常由于不同空间目标的坐标  
串长度不定，具有变长记录，并且数据项也可能很大，很复杂。

(4)传统数据库系统只操纵和查询文字和数字信息；           而空间数据库中需要有大量的空间数据操作和查询，  
如相邻、连通、包含、叠加等。

或者：总结标准 DBMS存储空间数据的局限性

空间数据记录是变长的（如点数的可变性），而一般的数据库都只允许把记录的长度设定为固定；  
在存储和维护空间数据拓扑关系方面存在着严重缺陷；  
一般都难以实现对空间数据的关联、连通、包含、叠加等基本操作；  
不能支持复杂的图形功能；  
单个地理实体的表达需要多个文件、多条记录，一般的 DBMS也难以支持；  
难以保证具有高度内部联系的 GIS 数据记录需要的复杂的安全维护。

4、什么是 SDBMS ? SDBMS 的三层体系结构是什么 ?

一个 **SDBMS** (空间数据库管理系统) 是一个软件模块, 它利用一个底层数据库管理系统 (如 **ORDBMS**、**OODBMS** );

**SDBMS** 支持多种空间数据类型、 相应的空间抽象数据类型 ( **ADT** ) 以及一种能够调用这些 **ADT** 的查询语言

**SDBMS** 支持空间索引、高效的空间操作算法以及用于查询优化的特定领域规则

**SDBMS** 包括: 空间数据模型、查询语言、文件组织、查询优化等。下图表示了基于对象关系模型上的一个空间数据库应用的三层体系结构。

**SDBMS** 三层体系结构

顶层为空间应用, 如 **GIS**、**MMIS** (多媒体信息系统) , 或者 **CAD**。该层不直接与 **OR-DBMS** 打交道, 需要一个中间层与 **OR-DBMS** 交互。

中间层: 空间数据库 ( **SDB** ), 中间层是封装大多数空间领域知识的地方, 不“插”入到 **OR-DBMS** 中。又称空间数据刀片、空间数据暗盒、空间数据引擎。

最后一层 ;**DBMS**

5、数据库模式有哪些 ?

物理模式 (物理层设计) 内模式、逻辑模式 (通常简称为“模式” ) 子模式 (外模式) 通常, 数据库管理系统支持一个物理模式、一个逻辑模式和多个子模式。

6、什么是数据模型? 概念模型有哪些? 逻辑模型有哪些? 每一种模式的原理是什么?

数据模型是数据库系统中关于数据内容和数据之间联系的逻辑组织的形式表示。每一个具体的数据库都由一个相应的数据模型来定义。 (数据库的概念描述, 是数据库系统中用于提供信息表示和操作手段的形式构架。)

概念模型: 按用户的观点从现实应用中抽象出事物以及事物之间的联系

结构数据模型: 从计算机实现的观点来对数据建模

概念模型:

实体 - 联系模型 ( **ER** )

现实世界被划分为若干实体 ( **entity** ), 由属性 ( **attribute** ) 来描述性质, 通过联系 ( **relationship** ) 互相关联

面向对象模型

逻辑数据模型:

层次模型

用树结构表示实体之间联系的模型叫层次模型

树由节点和连线组成

节点代表实体型

连线表示两实体型间的一对多联系

网状模型

网状数据模型是一个满足下列条件的有向图:

1、可以有一个以上的节点无父节点。

2、至少有一个节点有多于一个的父节点 (排除树结构) 。

关系模型

用二维表来表示实体及其相互联系

面向对象模型

为了有效地描述复杂的事物或现象，需要在更高层次上综合利用和管理多种数据结构和数据模型，并用面向对象的方法进行统一的抽象。

7、 数据库设计的三个步骤有哪些？每一步有些什么内容？

答、首先，采用高层次的概念数据模型来组织所有与应用相关的可用信息；

然后，逻辑建模阶段，与概念数据模型在商用 **DBMS** 上的具体实现有关

最后，数据库设计的第三个步骤是物理设计的建模，它解决数据库营养在计算机中具体实现是方方面面的细节。

改：概念模型

按用户的观点从现实应用中抽象出事物以及事物之间的联系

逻辑建模

建立概念和联系的逻辑结构

逻辑结构设计的步骤：

- 1) 将概念结构转化为一般的关系、网状、层次模型、面向对象模型
- 2) 对数据模型进行优化
- 3) 设计用户子模式

物理设计建模

对逻辑结构进行具体实现方面的安排和考虑

存储组织、索引、内存管理 .....

8、 ER 模型的作用， ER 图包括哪些要素，如何表达多值属性？

答： **ER** 图可以以一种避开计算机隐喻的方式来表达这个微型世界，从而把应用中的概念与实现细节分离开来。

**ER** 图包括实体（物理上或概念上独立存在的事物或对象）、属性和联系。实体用属性来刻画性质，实体之间通过练习相互作用和关联。属性可以是单值或多值。 **ER** 图中实体用矩形表示，属性表示为椭圆，联系为菱形。码属性加下划线，多值属性用双椭圆。

9、对于空间数据， ER 模型方法的不足之处？为表达空间概念，扩展 ER 模型主要增加了哪些要素？举例说明用象形符号扩展 ER 图，对于空间数据建模有何好处？

. **ER** 图在空间建模中的不足：

场模型无法用 ER 模型进行自然映射——因为：ER 模型的最初设计隐含了基于对象模型的假设。

传统 ER 模型中，实体之间的关系由应用来导出；而空间建模中，空间对象之间总会有内在联系。

建模空间对象所使用的实体类型与“地图”比例尺有关。有时是点、线，有时是多边形。

扩展 E-R 模型：

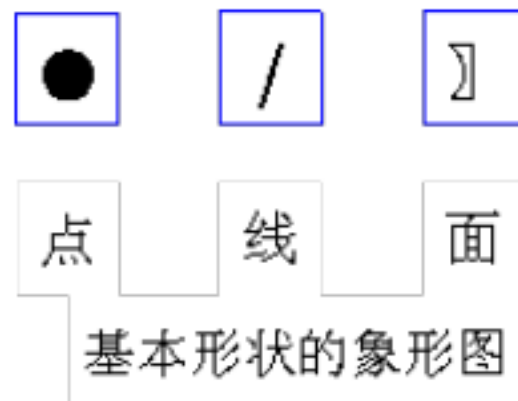
## 1) 实体象形图：

象形图：象形图是一种将对象插在方框内的微缩图表示，这些微缩图用来扩展 ER 图，并插到实体矩形框中的适当位置。

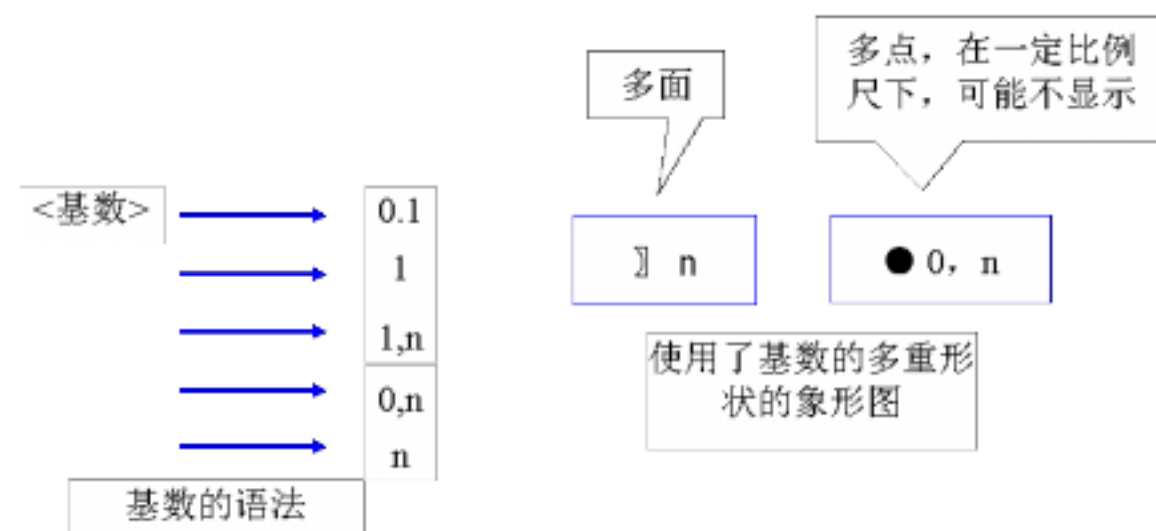
形状：形状是象形图的基本图形元素，它代表着空间数据模型中的元素。

一个模型元素可以是基本形状、复合形状、导出形状或备选形状。

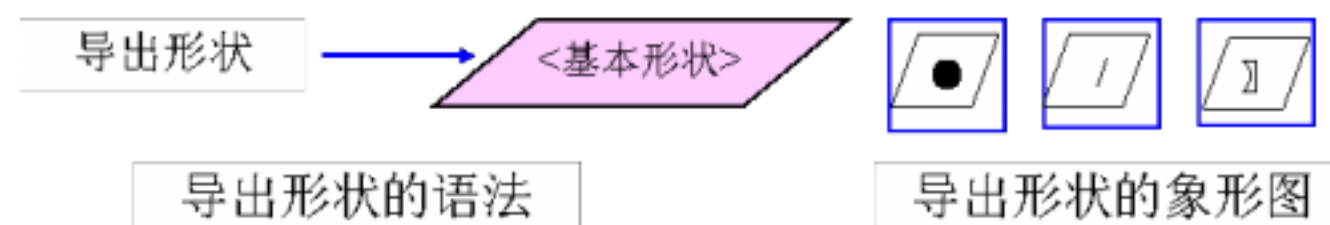
基本形状



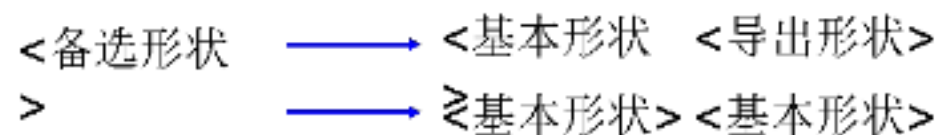
复合形状：为了处理那些不能用某个基本形状表示的对象，我们定义了一组聚合的形状，并用基数来量化这些复合形状



导出形状：如果一个对象的形状是由其他对象的形状导出的，那么就用斜体形式来表示这个象形图。



备选形状：备选形状可以用于表示某种条件下的同一个对象。例如，根据比例尺，一条河流可以表示成一个多边形或一条线。

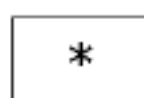


备选形状的语法



任意可能的形状

任意形状：对于形状的组合，我们用通配符（\*）表示，它表示各种形状。例如，一个灌溉网是由泵站（点）、水渠（线）以及水库（多边形）所组成的。



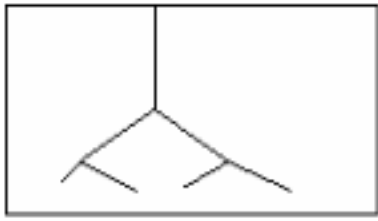
任意可能的形状



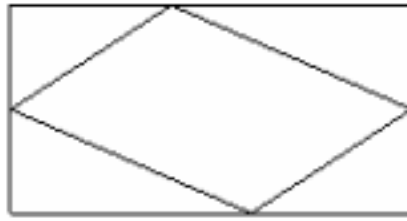
用户自定义形状

2) 联系象形图。

联系象形图用来构建实体间联系的模型。例如， **part-of** 用于构建道路与路网之间联系的模型，或是用于把森林划分成林分的建模。



Part of(网络)



Part of(分区)

好处：用象形符号扩展 ER 图，以便专门处理空间数据类型。 这将减少 ER 图以及所产生的关系模式的复杂度，同时改进空间建模的质量。空间联系 (例如 Road-Crosses-River)就可以从 ER 图中省略，用隐式的方式表示。关系模式中的表达多值空间属性的关系和 M：N 空间联系也就不需要了

10、举例说明如何将 ER 图映射成关系模型？

1.实体映射成单独关系

2.对于基数为 1 1 的联系转换为一个独立的关系模式，也可与任一端对应的关系模式合并。将任一实体的码属性作为其他关系的一个外码。 如 **Manager-Forest**

3.对于基数为 M 1 的联系，可以转换为一个独立的关系模式，也可以与 M 端对应的关系模式合并。 将“1”侧关系的主码作为“M”侧关系的外码，转换来的关系的主码为 M 侧的码。如 **Forest-FireStation**

4.对于基数 M:N 的联系，则每一个 M:N 的联系被映射成一个新关系，其主码由参与的实体对主码组成，联系的属性映射成关系的属性，如 **Facility-River**

5.对于多值属性，创建一个具有两列的新关系， 一列对应多值属性，另一列对应实体的码。多值属性和实体码一起构成新的关系的主码。如 **Forest-stand** 的几何属性 **polygonid**，新表为 **Fstand-Geom**。

6.多值属性 **Elevation** 也需要一个新表，表中由 **ForestName**、**Elevation** 和 **Pointid** 共同构成主码。

7. 具有相同码的关系模式可合并

11、常用的空间信息模型有哪些？它们分别由哪些内容组成？ 采用什么样的数据结构？ 基于每种空间信息模型有哪些操作？

两种常用空间信息模型：

场模型 ( **Field base model**)，采用栅格模型

对象模型 (**Object based model**)，采用矢量结构

场模型用于表示具有连续的空间变化的情况，形状不定的现象。对象模型用于表示具有固定形状的空间实体/概念描述空间上离散的空间对象。

场模型的 3 个组成部分：空间框架、场函数、场操作。

场操作分类：

( 1 ) 局部操作

对于局部操作，空间框架内一给定位置的新场取值只依赖于同一位置场的输入值。

( 2 ) 聚焦操作

指定位置的结果场的值依赖于同一位置的一个假定领域上的场的值

设  $E(x,y)$  是 **state-park** 的高程场， $E$  给出了空间框架  $F$  在位置  $(x,y)$  的高程值，计算高程场的梯度  $\nabla E(x,y)$ ，就是一个聚焦操作，梯度值依赖于  $(x,y)$  的邻域场  $(x_1, y_1)$  的高程。

( 3 ) 区域操作

与聚集运算符或积分运算有关。如在森林的例子中求某种树种的平均高度。

对象模型的组成部分：对象类型、对象属性和操作、对象关系。

空间对象的操作：面向集合的、拓扑的方位的、度量空间的、欧氏空间的

12、什么是范式理论？理解并简述函数依赖、部分函数依赖、部分函数依赖、传递函数依赖 的涵义。

范式是符合某一种级别的关系模式的集合。

设  $R(U)$  是一个属性集  $U$  上的关系模式， $X$  和  $Y$  是  $U$  的子集。若对于  $R(U)$  的任意一个可能的关系  $r$ ， $r$  中不可能存在两个元组在  $X$  上的属性值相等，而在  $Y$  上的属性值不等，则称“ $X$ 函数确定  $Y$ ”或“ $Y$ 函数依赖于  $X$ ”，记作  $X \rightarrow Y$ 。 $X$  称为这个函数依赖的决定属性集 (Determinant)。 $Y=f(x)$

函数依赖不是指关系模式  $R$  的某个或某些关系实例满足的约束条件，而是指  $R$  的所有关系实例均要满足的约束条件。

在关系模式  $R(U)$  中，如果  $X \rightarrow Y$ ，并且对于  $X$  的任何一个真子集  $X'$ ，都有  $X' \not\rightarrow Y$ ，则称  $Y$  完全函数依赖于  $X$ ，记作  $X \twoheadrightarrow Y$ 。若  $X \rightarrow Y$ ，但  $Y$  不完全函数依赖于  $X$ ，则称  $Y$  部分函数依赖于  $X$ ，记作  $X \rightharpoonup Y$ 。

在关系模式  $R(U)$  中，如果  $X \rightarrow Y$ ， $Y \rightarrow Z$ ，且  $Y \not\rightarrow X$ ， $Y \not\rightarrow Z$ ，则称  $Z$  传递函数依赖于  $X$ 。

注：如果  $Y \rightarrow X$ ，即  $X \rightarrow Y$ ，则  $Z$  直接依赖于  $X$ 。

13、结合实例，简述 1~4NF 的涵义，并能判别属于第几范式，及如何转换成更高级别的范式。

各种范式之间存在联系：

$1NF \supset 2NF \supset 3NF \supset BCNF \supset 4NF \supset 5NF$

某一关系模式  $R$  为第  $n$  范式，可简记为  $R$   $nNF$ 。

**1NF** 的定义：如果一个关系模式  $R$  的所有属性都是不可分的基本数据项，则  $R$  **1NF**。第一范式是对关系模式的最起码的要求。不满足第一范式的数据库模式不能称为关系数据库。

但是满足第一范式的关系模式并不一定是一个好的关系模式。

**2NF** 的定义：定义 5.6 若关系模式  $R$  **1NF**，并且每一个非主属性都完全函数依赖于  $R$  的码，则  $R$  **2NF**。（所有非主属性完全依赖每个候选关键字。）

例：**SLC(Sno, Sdept, Sloc, Cno, Grade)** **1NF**

**SLC(Sno, Sdept, Sloc, Cno, Grade)** **2NF**      **SC ( Sno, Cno, Grade )** **2NF**  
**SL ( Sno, Sdept, Sloc)** **2NF**

（**sloc**为学生住处，**sdept**为选课）

订单号	商品号	商品名	商品描述	单价	供应商号	供应商名	供应商电话
000001	200	A	..... 2.00	234560	XXXXXX	.....	
000001	201	B	..... 1.00	234560	XXXXXX	.....	
000001	202	C	..... 10.00	234560	XXXXXX	.....	
000001	203	D	..... 20.00	234560	XXXXXX	.....	
000001	204	E	..... 5.00	234560	XXXXXX	.....	
-----							
000002	200	A	..... 2.00	234561	YYYYYY	.....	
000002	201	B	..... 1.00	234561	YYYYYY	.....	
000002	202	C	..... 10.00	234561	YYYYYY	.....	
000002	204	E	..... 5.00	234561	YYYYYY	.....	

000003	202	C	.....	10.00	234560	XXXXXX	.....
000003	203	D	.....	20.00	234560	XXXXXX	.....
000003	204	E	.....	5.00	234560	XXXXXX	.....

（ 订单号   商品号   商品名   商品描述   单价   供应商号   供应商名   供应商电话 ）

其中：   主码（ 订单号    , 商品号    ）

          商品号    (商品名   ,商品描述   , 单价 )

          因为 “商品号 ”在表中是主键的一部分    ,    所以 “商品名   商品描述   单价 ”对于 “商品号 ”存在部分函数依赖 .

          将存在部分依赖关系的列拿出来新生成一个新的表            **Product**,  
          而原来的 **Order** 表中去掉了一些列 ,形成一个新的 **Order** 表,

**Order** 表： 订单号   商品号   供应商号   供应商名   供应商电话    ...

**Product** 表： 商品号   商品名   商品描述   单价    ...

采用投影分解法将一个    **1NF** 的关系分解为多个    **2NF** 的关系，可以在一定程度上减轻原        **1NF** 关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。

将一个 **1NF** 关系分解为多个    **2NF** 的关系，并不能完全消除关系模式中的各种异常情况和数据冗余。

**SL ( Sno ,   Sdept ,   Sloc )        2NF**

例： **2NF** 关系模式 **SL(Sno, Sdept, Sloc)**中

函数依赖：

**Sno   Sdept**  
          **Sdept   Sloc**  
          **Sno   Sloc**

**Sloc** 传递函数依赖于    **Sno**, 即 **SL** 中存在非主属性对码的传递函数依赖。

          解决方法

          采用投影分解法，把    **SL** 分解为两个关系模式，以消除传递函数依赖：

**SD ( Sno ,   Sdept )**  
          **DL ( Sdept ,   Sloc )**

**SD** 的码为 **Sno** ,   **DL** 的码为 **Sdept**

定义 **3FN**: 关系模式 **R<U , F>** 中若不存在这样的码    **X**、属性组 **Y** 及非主属性 **Z ( Z ⊆ Y )**，使得 **X   Y , Y    X , Y   Z** , 成立，则称 **R<U , F>**        **3NF**。( 所有非主属性都不传递函数依赖每个候选关键字或一个或多个属性    (列)依赖于非主键的属性    (列).    )

例，    **SL(Sno, Sdept, Sloc)**        **2NF**

**SL(Sno, Sdept, Sloc)**        **3NF**

**SD ( Sno ,   Sdept )        3NF**

**DL ( Sdept ,   Sloc )        3NF**

若 **R   3NF** , 则 **R** 的每一个非主属性既不部分函数依赖于候选码也不传递函数依赖于候选码。

如果 **R   3NF** , 则 **R** 也是 **2NF**。

采用投影分解法将一个    **2NF** 的关系分解为多个    **3NF** 的关系，可以在一定程度上解决原        **2NF** 关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。

          将一个 **2NF** 关系分解为多个    **3NF** 的关系后，并不能完全消除关系模式中的各种异常情况和数据冗余。

学生关系表 **Student**(学号 , 姓名 , 年龄 , 所在学院 , 学院地点 , 学院电话 ) ,

关键字：    “学号 ” ,

(学号 )        (姓名 , 年龄 , 所在学院 , 学院地点 , 学院电话 )



(学号) (所在学院) (学院地点, 学院电话):即存在非关键字段 “学院地点”、“学院电话”对关键字段 “学号”的传递函数依赖。

这个关系是符合 2NF 的,但是不符合 3NF ,  
它也会存在数据冗余、更新异常、插入异常和删除异常的情况 ,  
把学生关系表分为如下两个表 :

学生 : (学号, 姓名, 年龄, 所在学院 );  
学院 : (学院, 地点, 电话)。

这样的关系是符合 3NF

假设仓库管理关系表为 StorehouseManage(仓库 ID, 存储物品 ID, 管理员 ID, 数量) ,且一个管理员只在一个仓库工作 ; 一个仓库可以存储多种物品。判断该关系模式所属范式

这个数据库表中存在如下决定关系 :

(仓库 ID, 存储物品 ID) (管理员 ID, 数量)  
(管理员 ID, 存储物品 ID) (仓库 ID, 数量)

所以, (仓库 ID, 存储物品 ID) 和(管理员 ID, 存储物品 ID) 都是 StorehouseManage的候选关键字 ,  
表中的唯一非关键字段为数量 , 它是符合第三范式的。

范式的判断 :

- 1、确定候选键,找出主属性和非主属性
- 2、确定非主属性和候选键之间是否存在函数依赖,若存在部分函数依赖,则关系模式属于 1NF, 若存在传递函数依赖,则关系模式属于 2NF, 若消除了部分函数依赖和传递函数依赖,则关系模式属于

3NF

候选键的确定 :

- 1、可以按照候选键的定义求解,即关系模式 R ( U , F ) 中的一个或一组属性 X , 若属性集 U 完全依赖于 X , 则 X 为关系模式 R 的候选键。也就是说根据语义分析得到的 F , 如果 X 可以确定每一个属性, 那么 X 就是候选键。

4NF 定义: 关系模式 R ( U , F ) 1NF , 如果对于 R 的每个非平凡多值依赖 X ↔ Y ( Y 不包含于 X ) , X 都含有候选码, 则 R 4NF 。



4NF 限制关系模式的属性之间不允许有非平凡且非函数依赖的多值依赖。



如果一个关系模式是 4NF , 则必为 BCNF

课程 C	教师 T	参考书 B
数学	邓海	高数
数学	邓海	数学分析
数学	邓海	微分方程
数学	陈红	高数
数学	陈红	数学分析
数学	陈红	微分方程
物理	李东	普通物理
物理	李东	光学
...	...	...

关系模式：**TEACH(C,T,B)**，**C**表示课程，**T**表示教师，**B**表示参考书。假设某一门课由多个教师讲授，一门课使用相同的一套参考书。

关系模式存在以下依赖：

数学 【邓海,陈红】【高数,数学分析,微分方程】

物理 【李东,张强,刘明】【普通物理学,光学】

该关系模式码为（**C，T，B**），为全码。满足**BCNF**，但仍存在四种异常。

为什么呢？

对**TEACH（C，T，B）**处理，去掉多值依赖。

分解两个关系模式：

**CT（C，T）** **4NF**

**CB（C，B）** **4NF**

14、什么是拓扑关系，举例说明拓扑与非拓扑特性、拓扑与非拓扑操作。

拓扑关系

答：是指满足拓扑几何学原理的各空间数据间的相互关系。即用结点、弧段和多边形所表示的实体之间的邻接关联和包含等关系。

拓扑特性：弹性变形后临近物体之间的拓扑关系没有发生改变

非拓扑特性：弹性变形后临近物体之间的拓扑关系发生了改变

拓扑操作与非拓扑操作

常见的拓扑属性

endpoint(point, arc)	点是弧的端点
simple-nonself-intersection(arc)	非自交的弧
on-boundary(point, region)	点在区域的边界上
inside(point, region)	点在区域内部
outside(point, region)	点在区域之外
open(region)	区域是开域（不包括边界）
close(region)	区域是闭域（包括边界）
connected(region)	区域是连通域（区域上任 2 点，都有路径相连）
inside(point, loop)	点在环中
crosses(arc, region)	弧穿过区域
touches(region, region)	区域与区域相邻
touches(arc, region)	弧与区域相邻
overlap(region, region)	区域与区域重叠

常见的非拓扑属性

Euclidean-distance(point, point)	2 点间的欧氏距离
direction(point, point)	点在点的东面
length(arc)	弧的长度（单位向量长度为 1 个单位）
perimeter(area)	区域的周长（单位正方形的周长为 4 个单位）
area(region)	区域的面积（单位正方形的面积为 1 个平方单位）

拓扑信息：研究空间相关的事物本身或者事物之间的在空间坐标变换下的不变质

事物本身的内外关系

事物之间的相离、相接、相交

事物之间相连的布局

几何信息

描述了事物在空间中的位置及所占据的范围

将地球表面以投影方式转换为平面

通过平面几何来抽象描述和研究事物的位置和范围

用图形和符号的方式来描绘这些空间相关的事物

属性信息

与位置范围无关的其它信息

描述了事物本身的内在性质和外在表现

事物之间的非位置关系

??? 用于空间对象之间拓扑关系的操作测试

8 个

<b>Equal</b>	相等 —— 若 2 个几何体的内部和边界在空间上都相等，则返回真
<b>Disjoint</b>	相离 —— 若 2 个几何体的内部和边界都不相交，则返回真
<b>Intersect</b>	交叠 —— 若 2 个几何体相交，则返回真
<b>Touch</b>	相接 —— 若 2 个面仅边界相交，而内部不相交，则返回真
<b>Cross</b>	横过 —— 若一条线和面的内部相交，则返回真
<b>Within</b>	在内部 —— 若给定的几何体的内部不与另一个几何体的外部相交，则返回真
<b>Contains</b>	包含 —— 若给定的几何体包含另一个几何体，则返回真
<b>Overlap</b>	覆盖 / 被覆盖 —— 若 2 个几何体的内部有非空交集，则返回真

15. OGIS 提出的关于空间几何体的基本构件有哪些？

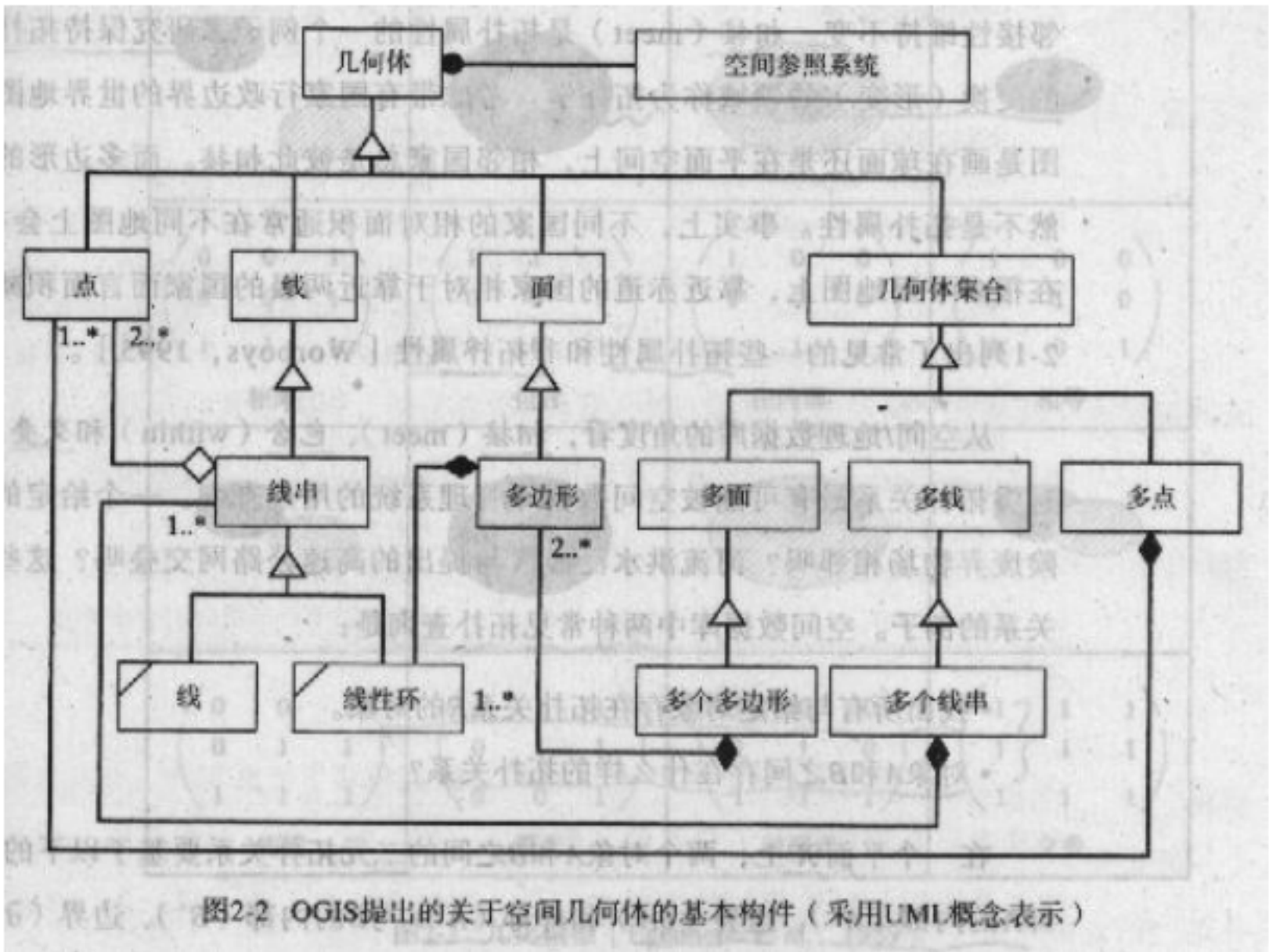


图2-2 OGIS提出的关于空间几何体的基本构件（采用UML概念表示）

16、OGIS 支持的空间操作有哪些？

OGIS 类中操作分 3 类

用于所有几何类型的基本操作

6 个

<b>SpatialReference( )</b>	返回几何体的基本坐标系统
<b>Envelope( )</b>	返回包含几何体的最小外接矩形
<b>Export( )</b>	返回以其他形式表示的几何体
<b>IsEmpty( )</b>	若几何体为空集，则返回真
<b>IsSimple( )</b>	若几何体为简单的（不自交的），则返回真
<b>Boundary( )</b>	返回几何体的边界
用于空间对象之间拓扑关系的操作测试	
8 个	

<b>Equal</b>	相等 —— 若 2 个几何体的内部和边界在空间上都相等，则返回真
<b>Disjoint</b>	相离 —— 若 2 个几何体的内部和边界都不相交，则返回真
<b>Intersect</b>	交叠 —— 若 2 个几何体相交，则返回真
<b>Touch</b>	相接 —— 若 2 个面仅边界相交，而内部不相交，则返回真
<b>Cross</b>	横过 —— 若一条线和面的内部相交，则返回真
<b>Within</b>	在内部 —— 若给定的几何体的内部不与另一个几何体的外部相交，则返回真
<b>Contains</b>	包含 —— 若给定的几何体包含另一个几何体，则返回真
<b>Overlap</b>	覆盖 / 被覆盖 —— 若 2 个几何体的内部有非空交集，则返回真
用于空间分析的一般操作	
7 个	


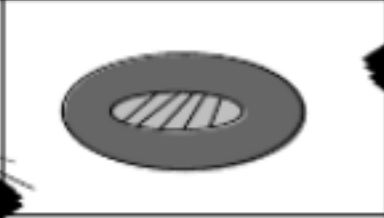
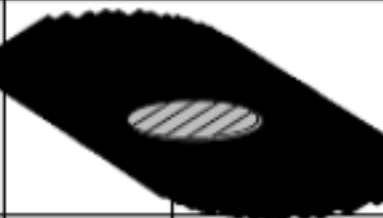
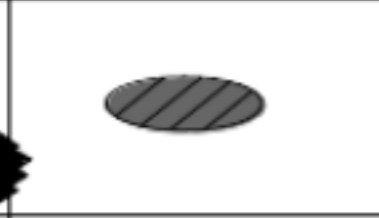

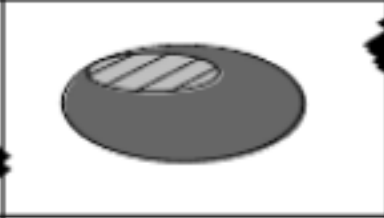


<b>Distance</b>	求距离 —— 返回 2 个几何体之间的最短距离
<b>Buffer</b>	求缓冲区 —— 返回到给定几何体距离小于等于指定值的几何体的点的集合
<b>ConvexHull</b>	求最小闭包 —— 返回几何体的最小闭包
<b>Intersection</b>	集合交 —— 返回 2 个几何体的交集构成的几何体
<b>Union</b>	集合并 —— 返回 2 个几何体的并集构成的几何体
<b>Difference</b>	集合差 —— 返回几何体与给定几何体不相交的部分
<b>SymmDiff</b>	返回 2 个几何体与对方互不相交的部分

17. 说明九交模型表达拓扑关系的原理。

在一个平面上。两个对象 **A**、**B** 之间的二元拓扑关系主要基于以下的相交情况，即分别是 **A** 和 **B** 的内部、边界、外部。值六部分可以构成九交模型。

考虑取值有空 **(0)** 和非空 **(1)**，可以确定有 **29=512** 种二元拓扑关系。 对于 **R2** 嵌在中的二维区域，有八个关系是可实现的，并且它们彼此互斥且完全覆盖。 ：相离、相接、交叠、相等、包含、在内部、覆盖、被覆盖。

$$\Gamma_9(A, B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

			
$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ disjoint	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ contains	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ inside	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ equal
			
$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ meet	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ covers	$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ coveredBy	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ overlap

18. 简述关系模式中的三种完整性。

答：码约束：每个关系必须要有一个主码；

实体完整性约束：主码不能为空；

参照完整性约束：外码的属性值要么是另一个关系的主码，要么为空值。

19. 主码、外码的概念。

从候选码中选择一个唯一地标识一个元组候选码作为码。 若候选码多于一个，则选定其中的一个做为主

码（ **Primary Key** ）。

外码：关系模式 **R** 中属性或属性组 **X** 并非 **R** 的码，但 **X** 是另一个关系模式的码，则称 **X** 是 **R** 的外部码，简称外码

20、UML 的作用？了解 UML 的主要符号。

**UML** 是用于面向对象软件设计的概念层建模的新兴标准之一，它是一种标准化语言，用于在概念层对结构化模式和动作行为进行建模。

符号：类——等价于 **ER** 图中实体，可象形图扩展；属性——数据成员；方法——成员函数；关系——类之间的联系。 3 种关系：聚合关系，泛化关系，关联关系。

改： **UMLCD** 符号

类 —— 等价于 **ER** 图中实体， 可象形图扩展

属性 —— 数据成员： + —— 公有的； - —— 私有的； # —— 受保护的

方法 —— 成员函数

关系 —— 类之间的联系，类似于 **ER** 图中联系

3 种关系：聚合关系 —— 整体 -部分关系，一个类作为另一个类的一部分 —— 强聚合



一个类作为多个类的一部分 —— 弱聚合

泛化关系 —— **generalization**，几个子类抽象出一个父类

关联关系 —— 不同类的对象之间的联系。涉及 **n** 个类 —— **n** 元关联

21、比较 ER 与 UML。

答：1) 没有方法的类就是实体； 2) 属性在两个里都一样； 3) **UML** 中没有主键和完整性约束； 4) **ER** 中没有方法； 5) **ER** 中关系的内容更丰富； 6) **ER** 图中的实体与数据集有关，但 **UML** 的类几乎和数据集无关。

改：

### ■ 比较**ER**与**UML**



22、请列举 SQL 所包含哪几个部分？每个部分的功能是什么？对每种功能列举相关的操作符（语句）。

数据定义语言 (**DDL**)，例如：**CREATE**、**DROP**、**ALTER** 等语句。

数据操作语言 (**DML**)，例如：**INSERT**（插入）、**UPDATE**（修改）、**DELETE**（删除）语句。 数据查询

语言 (**DQL**)，例如：**SELECT** 语句。

数据控制语言 (**DCL**)，例如：**GRANT**、**REVOKE**、**COMMIT**、**ROLLBACK** 等语句。

23、SQL 有哪些版本，每个版本有什么特点？（参照 PPT）空间数据类型和操作被允许加入到 SQL 的哪个版本中 (SQL3)。

SQL 版本：**SQL2/SQL92**、**SQL3/SQL99**

**SQL-86**

**SQL-89**：“具有完整性增强的数据库语言 **SQL**”，增加了对完整性约束的支持

**SQL-92**：“数据库语言 **SQL**”，是 **SQL-89** 的超集，增加了许多新特性，如新的数据类型，更丰富的数据操作，更强的完整性、安全性支持等。

**SQL-3**：新的标准，增加了对面向对象模型的支持

## 24、SELECT

FROM

WHERE

ORDER BY

GROUP BY,

HAVING

自己编写 SQL语句实现：

- (1) 查询员工信息表 employee 中每个员工的所有信息
- (2) 查询员工信息表 employee 中员工的姓名和年龄
- (3) 在员工信息表 employee 中按照员工年龄降序查询数据
- (4) 在员工信息表 employee 中年龄在 20~26 岁的员工姓名
- (5) 在员工信息表 employee 中查询姓赵的员工信息
- (6) 在员工信息表 employee 中，求所有员工业绩的总合
- (7) 在员工信息表 employee 中，查询业绩最高的员工信息
- (8) 查询员工信息表 employee 中员工的数量
- (9) 在员工信息表 employee 中，按照员工部门对记录进行分组
- (10) 在员工信息表中，统计各部门员工的总业绩
- (11) 在员工信息表中，按照部门进行分组并对计算部门员工的平均年龄，再查询平均年龄小于 22 的信息。

**答：**1) SELECT \* from employee

(2) select 员工姓名, 员工年龄 from employee

(3) select \* from employee

order by 员工年龄 desc

(4) select 员工姓名 from employee

where 员工年龄 between 20 and 26

(5) select \* from employee

where 员工姓名 like '赵%'

(6) select sum(员工业绩) as 员工业绩总和 from employee

(7) select \* from employee

where 员工业绩 =( select MAX(员工业绩) from employee )

(8) select COUNT(员工编号) as 员工数量 from employee

(9) select COUNT(\*) as 部门数, 所在部门 from employee

group by 所在部门

(10) select 所在部门, SUM( 员工业绩 ) as 总业绩 from employee

group by 所在部门

( 11 ) select 所在部门, AVG( 员工年龄 ) as 平均年龄 from employee

group by 所在部门

having AVG( 员工年龄 ) < 22

( 12 ) PPT中增加、删除、更新表中数据的例句

( 13 ) PPT中创建、修改、删除表结构的例句

( 14 ) PPT中授予、收回权限的例句

25、读懂书中关系代数查询、每一个空间查询例句。或者给出查询目的，要求写语句。

( 1 ) 查询：列出 Country 表中所有与美国相邻的国家名字

( 2 ) 查询：列出 River 表中河流流经的国家名字

( 3 ) 查询：对于 River 表中列出的河流，在 City 表中找到距其最近的城市

( 4 ) 查询：列出距劳伦斯河方圆 300km的城市

( 5 ) 查询：列出 Country 表中每个国家的名字、人口和国土面积

( 6 ) 查询：求出河流在流经的各国家境内的长度

( 7 ) 查询：列出每个国家的 GDP及其首都到赤道的距离

( 8 ) 查询：按邻国多少列出所有国家

( 9 ) 查询：列出只有 1 个邻国的国家

( 10 ) 查询：哪个国家的邻国最多

(11) 用 SQL语言查询：圣劳伦斯河发源地国家的首都的名字是什么，该城市的人口是多少？

(12) 用集合并运算列出所有符合下列条件的国家：它们要么在北美州，要么是河流发源地的国家

(13) 用关系代数列出所有位于北美洲但不是河流发源地的国家。

(14) 用关系代数列出要么是首都城市，要么人口 >2 百万的城市

(15) 用关系代数列出 GDP超过 20 亿美元的国家的首都和人口数

( 16 ) 用事务实现，对数据表 table\_1 进行插入记录的工作，当遇到错误时回滚到插入数据前的状态

( 17 ) 用事务实现，阻止其他用户对数据表进行修改，但可以查询

( 18 ) 用带锁的方式创建事务，阻止其他用户对数据表 table\_1 进行访问

25. ( 1 ) 查询：列出 Country 表中所有与美国相邻的国家名字

**SELECT C1.Name AS “ Neighbors of USA ”**

**FROM Country C1, Country C2**

**WHERE Touch(C1.Shape, C2.Shape) = 1 AND** ( 拓扑相接 )

**C2.Name = ‘ USA’**

（ 2 ）查询：列出 River 表中河流流经的国家名字

```
SELECT      R.Name, C.Name
FROM        River R, Country C
WHERE       Cross(R.Shape, C.Shape) = 1           （ 横过 ）
```

（ 3 ）查询：对于 River 表中列出的河流，在 City 表中找到距其最近的城市

```
SELECT      C1.Name, R1.Name
FROM        City C1, River R1
WHERE       Distance(C1.Shape, R1.Shape) <        （ 求距离 ）
            ALL(SELECT      Distance(C2.Shape, R1.Shape)
FROM          City C2
WHERE         C1.Name <> C2.Name)
```

（ 4 ）查询：列出距劳伦斯河方圆 300km 的城市

```
SELECT      Ci.Name
FROM        City Ci, River R
WHERE       Overlap(Ci.Shape, Buffer(R.Shape, 300)) = 1    AND   （ 被覆盖缓冲区 ）
            R.Name = ' St. Lawrence
```

（ 5 ）查询：列出 Country 表中每个国家的名字、人口和国土面积

```
SELECT      C.Name, C.Pop, Area(C.Shape) AS      “ Area ” （ 求面积，仅适用于多边形、多个多边形，
若为经纬度坐标，则需中间变换，对求距离、长度一样 ）
FROM        Country C
```

（ 6 ）查询：求出河流在流经的各国家境内的长度

```
SELECT      R.Name, C.Name, Length(Intersection(R.Shape, C.Shape) AS      “ Length ” （ 求长度，线串
与多边形的交集为线串 ）
FROM        River R, Country C
WHERE       Cross(R.Shape, C.Shape) = 1           （ 河流流经的国家 ）
```

（ 7 ）查询：列出每个国家的 GDP 及其首都到赤道的距离

```
SELECT      Co.Name, Co.GDP, Distance(Point(0, Ci.Shape.y), Ci.Shape) AS      “ Distance ” （ 求距离，
赤道上与城市经度相同的点 ）
FROM        Country Co, City Ci
WHERE       Co.Name = Ci.Country    AND
            Ci.Capital = ' Y ’
```

（ 8 ）查询：按邻国多少列出所有国家

```
SELECT      Co.Name, Count(Co1.Name)                （ 计数 ）
FROM        Country Co, Country Co1
WHERE       Touch(Co.Shape, Co1.Shape) =1           （ 相邻 ）
GROUP BY Co.Name                                    （ 按国家分组 ）
ORDER BY Count(Co1.Name)                            （ 按计数排序 ）
```

（ 9 ）查询：列出只有 1 个邻国的国家

```
SELECT      Co.Name
FROM        Country Co, Country Co1
WHERE       Touch(Co.Shape, Co1.Shape) =1           （ 2 国相邻 ）
GROUP BY Co.Name                                    （ 按国家分组 ）
HAVING      Count(Co1.Name) = 1                    （ 计数为 1 ）
```

```
SELECT      Co.Name
FROM        Country Co
WHERE       Co.Name IN      ( 满足 1 个邻国条件 )
                        (SELECT      Co.Name
                          FROM        Country Co, Country Co1
                          WHERE       Touch(Co.Shape, Co1.Shape)=1
                          GROUP BY    Co.Name
                          HAVING      Count(*) = 1)                ( 计数为 1 )
```

(10 ) 查询：哪个国家的邻国最多

CREATE VIEW Neighbor AS ( 创建视图 **Neighbor** )  
( 复杂查询 第一个查询 —— 计算各  
国邻国数 )

```
SELECT      Co.Name, Count(Co1.Name) AS  “ Num_neighbors ” ( 国家计数做新属性 )
FROM        Country Co, Country Co1
WHERE       Touch(Co.Shape, Co1.Shape) =1                ( 2 国相邻 )
GROUP BY    Co.Name                                     ( 按国家分组 )
```

( 第二个查询 —— 从视图 **Neighbor** 中选出邻国数最大的国家 )

```
SELECT      Co.Name, Num_neighbors
FROM        Neighbor
WHERE       Num_neighbors = (SELECT Max(Num_neighbors)      ( 求最大值 )
                             FROM Neighbor)
```

(11)查询：圣劳伦斯河发源地国家的首都的名字是什么，该城市的人口是多少？

```
SELECT      Ci.Name,Ci.Pop
FROM        City Ci, Country Co, River R
WHERE       R.Origin=Co.Country  AND
            Co.Name=Ci.Country  AND
            R.Name= St.Lawrence 'AND
            Ci.Capital= 'Y '
```

(12) 用集合并运算列出所有符合下列条件的国家：它们要么在北美州，要么是河流发源地的国家

- 1 ) R= Name ( Cont = NAM(Country))
- 2 ) S= origin (River)
- 3) R S 结果表见 P69

(13) 列出所有位于北美洲但不是河流发源地的国家。

- 1 ) R= Name ( Cont = NAM(Country))
- 2 ) S= origin (River)
- 3) R-S 结果表见 P69

26. 简述事务的概念及特征。

事务的定义：事务是数据库中执行的一个工作单位，它是由用户定义的一组操作序列组成。这些操作“要么全做，要么都不做”。

事务的特征 —— ACID 原则



- ( 1 ) 原子性 ( **Atomicity** ) :指的是整体性，全部操作的不可再分，要么不执行，要么全部执行。
- ( 2 ) 一致性 ( **Consistency** ) : 事务执行的结果必须是使数据库从一个一致性状态变到另一个一致状态。
- ( 3 ) 隔离性 ( **Isolation** ) : 一个事务的执行不能被其他事务干扰。
- ( 4 ) 持久性 ( **Durability** ) : 也称永久性，指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。

27. 什么是并发操作？并发操作可能带来哪些问题？

数据库的重要特征是支持数据共享，允许多个用户程序并行地存取数据库中的数据。这样，多个用户或多个事务可能同时对同一数据进行操作，称为并发操作。

如果系统对并发操作不加以控制，就会存取或存储不正确的数据，破坏数据的完整性。

并发操作带来的三个问题： 1 ) 丢失修改 2 ) 污读 3 ) 不可重读

28. 简述污读、不可重读、活锁、死锁的概念。

污读：当事务 T1 在读取数据对象时另一个事务 T2 同时对其进行修改，导致事务 T1 读取的数据不正确。

不可重读：当事务 T1 首先读取数据对象，事务 T2 修改数据对象的值导致事务 T1 再次读取数据对象时与前一次读取的对象不一致。

活锁：当某个事务请求对某一数据的排他性封锁时，由于其他事务一直优先得到对该数据的封锁与操作而使这个事务一直处于等待状态，这种状态形成活锁。

死锁：指的是多个事务因封锁冲突（竞争资源）而永远等待下去的情形。也就是说，同时处于等待状态的事务间，每个事务的执行都以另一个事务释放锁为前提，结果造成任何一个事务都无法得到执行的现象。

29. 简述锁的类型及其作用？

锁的类型有：排他锁（写锁， X 锁 **exclusive lock**）作用：可以防止并发事务对资源进行访问

共享锁（读锁， S 锁 **share lock**）作用：允许并行事务读取同一种资源， 这时的事务不能修改访问的数据。

30. 简述一级、二级、三级封锁协议的内容和区别。

一级封锁协议：内容：事务 T 在修改数据对象前必须对其加 X 锁，直到事务结束才释放。可以解决“丢失修改”问题！

二级封锁协议：内容：在一级封锁协议的基础上，另外加上事务 T 在读取数据对象 R 前必须对其加 S 锁，读完后立即释放。可以解决“污读”问题！

三级封锁协议：内容：对于二级封锁协议当中的读锁，直到事务 T 结束才释放。可以解决“不可重读”问题！

31. 简述解决活锁、死锁的方法。

活锁问题：当某个事务请求对某一数据的排他性封锁时，由于其他事务一直优先得到对该数据的封锁与操作而使这个事务一直处于等待状态，这种状态形成活锁最简单的方法就是先锁。如何才能避免活锁呢？来先服务的策略。按照请求封锁的次序对事务排队，一旦记录上的锁释放，就使申请队列中的第一个事务获得锁。

预防死锁的方法： 1 ) 一次封锁法 2 ) 顺序封锁法

死锁的诊断与解除

1 ) 超时法：当某事务的等待时间超过了规定的时限，就认为发生了死锁。

2 ) 等待图法：用一个有向图表示事务等待的情况。

32. view( 视图 ) 的含义和创建语句。

含义：视图是用来描述导出数据或查询结果简化复杂网状查询的表

**CREATE VIEW**

<视图名> [( <列名> [, <列名> ] ... )]

**AS** <子查询>

**[WITH CHECK OPTION]** ；表示对视图进行 **update,insert** 和 **delete** 操作时要保证更新、插入或删除的行满足视图定义中的谓词条件（即子查询中的条件表达式）。

33.计算机存储设备的种类？优缺点分别是什么？

寄存器（**register**）：与运算部件直接连接，速度最快，极少（几十个）

高速缓冲存储器（**cache memory**）：在 **CPU** 中，速度极快，容量小（几十 **K~2M**）

主存储器（**main memory**）：速度很快（纳秒级），一般容量在几十 **M** ~ 几个 **G**

随机访问：访问任何存储单元，时间相同；易失性：断电丢失。

快闪存储器（**flash memory**）：速度受到存储介质和接口限制；随机访问，非易失性，断电不丢失

磁盘存储器（**disk memory**）：同上，但是机械装置，速度更慢

光盘存储器（**CDROM/CDR/CDRW/DVD**）：只读，可写一次，可重复读写；机械装置，随机访问，速度更低

磁带存储器（**tape**）：速度最低，容量价格比最高（至几百 **G**）

34、磁盘存储相关概念：磁道 **track**、扇区 **sector**、柱面 **cylinder**？页面的概念？

答：磁道：圆心磁盘片上向边缘延伸的同心圆

扇区：每个磁道中被分成若干等份的区域

柱面：是磁盘上具有相同磁道的集合

页面：又称磁盘块。是磁盘与主存之间的最小传输单位

35、访问磁盘扇区数据的过程，哪个过程花费的时间最多？

全部存取时间： $ta = ts + tl + tt$

**ts** 寻道时间 —— 磁头到达特定磁道的时间（平均 4 ~ 10 毫秒）

**tl** 延迟时间 —— 磁盘块旋转到磁头下方的时间（平均 2 ~ 5 毫秒）

**tt** 传输时间 —— 磁头读/写块中数据的时间

一般： $ts > tl > tt$

36、域 (filed)、记录 (record)、文件 (file) 的概念，

1、数据项：是可以定义数据的最小单位，也叫元素、基本项、字段等。数据项与现实世界实体的属性相对应，数据项有一定的取值范围，称为域。

2、记录：由若干相关联的数据项组成。

3、文件：文件是一给定类型的（逻辑）记录的全部具体值的集合。

37、页面的概念

页面的概念：磁盘与主存之间的最小传输单位。一个文件可能跨越多个页面。一个页面是槽的集合，一个

槽包含一条记录

38、什么是聚类、内部聚类、外部聚类、全局聚类？

聚类：以某种搜索码值的顺序安排记录的物理存储

空间聚类：空间数据库中 —— 空间上相邻的、查询上关联的对象 —— 存储在一起

内部聚类 ( internal clustering ) —— 为了加快对单个对象的访问, 一个对象的全部表示都存放在一个磁盘页面中

本地聚类 ( local clustering ) —— 为了加快对多个对象的访问, 一组空间对象被分配在同一磁盘页面中, 一般: 依据数据空间中对象的位置 /近似

全局聚类 ( global clustering ) —— 一组空间邻接的对象存储在多个物理上邻接的磁盘页面中, 这些页面可由一条读命令访问

39、使用空间填充曲线组织空间数据的意义?

答: 1) 空间中的位置排序 2) 据中使用传统的有效搜索

40. 结合实例, 简述 Hilbert、Z 曲线编码原则。

**Z 曲线:** 1) 读入空间对象点的  $x, y$  坐标 —— 二进制表示

2) 对二进制的  $x, y$  坐标的每一位, 隔行扫描, 形成一个由 0, 1 组成的字符串

3) 计算该二进制

字符串的十进制数值,

该十进制数 —— **Z 值**

4) 按 **Z 值**, 由小到大,

连线 —— **Z 曲线**

**Hilbert 曲线算法**

1) 读入对象点的  $x, y$  坐标 —— 二进制表示,  $n$  位

2) 对二进制的  $x, y$  坐标的每一位, 隔行扫描, 形成一个由 0, 1 组成的字符串 (图 a))

3) 将该字符串自左至右分成 2 位长的串  $s_i, i = 1, 2, \dots, n$

4) 给每个 2 位长的串规定一个十进制数  $d_i$ , 如: 规定 “00”为 0, “10”为 3, “11”为 2 (图 b))

对于上步合并后的数组, 对左第一位值, 若:

$j = 0$  —— 把后面所有的 1 变成 3, 3 变成 1

$j = 3$  —— 把后面所有的 0 变成 2, 2 变成 0 (图 c))

计算变换后的二进制串的十进制数, 按数值大小, 由小到大连线 —— **Hilbert 曲线** (图 d))

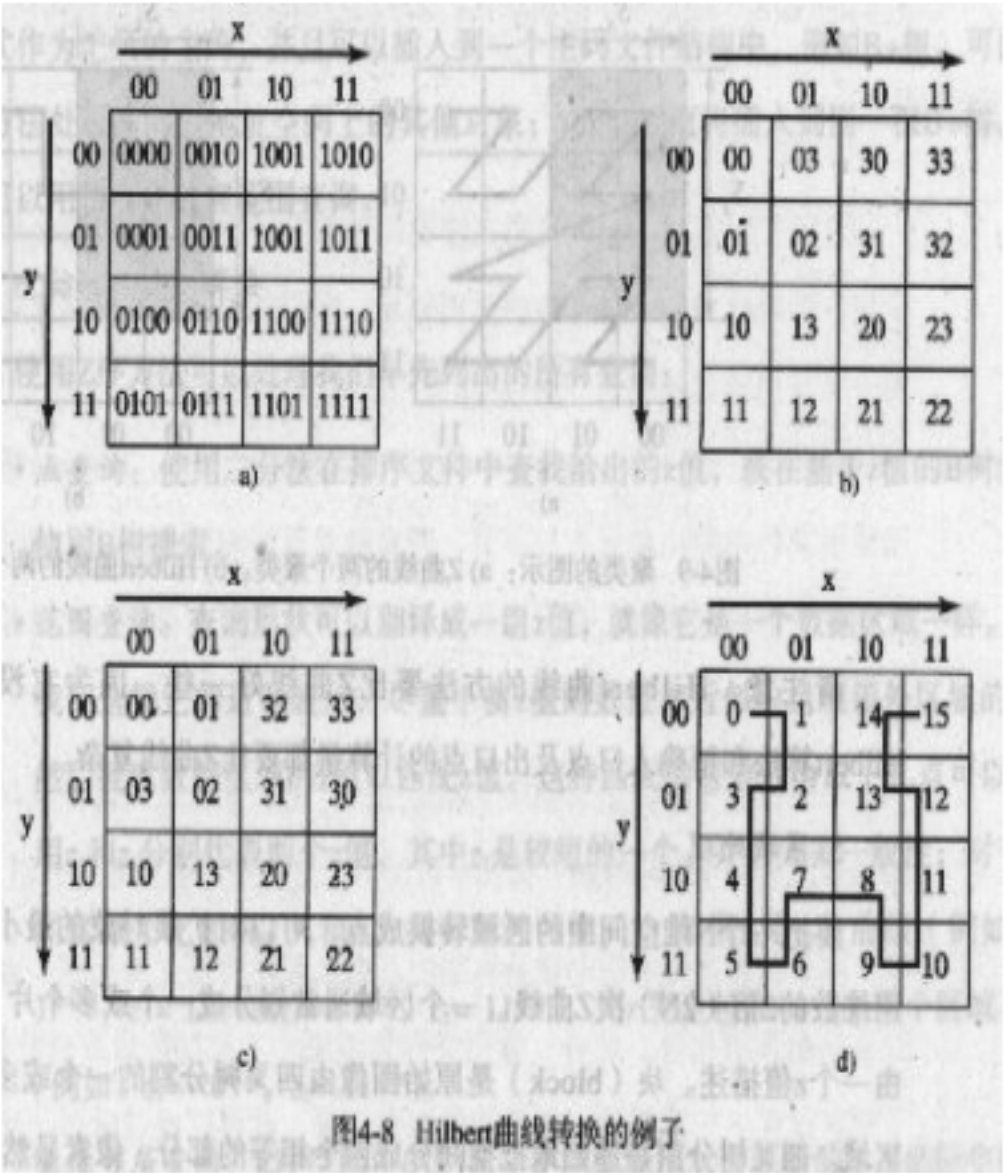


图4-8 Hilbert曲线转换的例子

41、什么是索引？索引文件的内容。主索引和二级索引。

索引文件是用来提高数据文件查询效率的辅助文件。记录的只有码值和数据文件中的页面地址。索引记录被排序，数据文件本身可以是不按关键码排序。

主索引，如果数据文件的记录是按照主码排列的，那么索引就只需要保存数据文件的每个磁盘页面第一个主码域值。每个索引记录一个数据页面。

二级索引：堆数据文件，一个索引记录一个数据。

一个磁盘最多只有一个主索引，因为主索引决定了数据在磁盘上的存储顺序。

42、什么是空间索引？有哪些空间索引方法？阐述格网索引、四叉树索引、R 树索引的基本思想。

答：空间索引结构用一组桶（通常对应二级存储的页面）来组织对象。

空间索引呢就是依据空间对象的位置和形状或空间对象之间的某种空间关系按一定的顺序排列的一种数据结构，其中包含空间对象的概要信息，如对象的标识、外接矩形及指向空间对象实体的指针。

方法：1)在系统中加入专门的外部空间数据结构，为空间属性提供如同B 树之于线性属性的功能。

2)使用空间填充曲线（如Z 序、Hilbert 曲线）将空间对象映射到一维空间，以便空间对象存储在标准的一维索引（例如B 树）中。

43、网格文件包含哪两部分内容？建立格网索引的思路和步骤？了解R 树索引和R+树索引的思想？

由二部分组成：

网格目录 —— 目录中每一项指向一个数据桶

线性比例的一维数组 —— 标示网格目录的索引，包含对象（记录）的块/桶。（如图中的每个桶的号码）

**R 树的特性：**

对于空间中的 **M** 个对象,每个页面 **m** 个键：

每个叶结点，包含 **m~M** 条索引记录（ **m<=M/2** ），除非它是根结点

一个叶结点上的每条索引，记录项（ **I**，元组标示符）。**I**—— **MBR**，在空间上包含了所指元组表达的 **k** 维数据对象；元组标示符 —— 对应 **MBR** 的空间对象的元组的唯一标示符

每个非叶结点，都有 **m~M** 个子结点，除非它是根结点

对于非叶结点中的每个项（ **I**，子结点指针）。**I**—— 子结点指针指向的、更低层次上结点项中所有矩形的 **MBR**

根结点，至少有 **2** 个子结点，除非它是叶结点

所有叶结点出现在同一层上

所有 **MBR** 的边与全局坐标系的轴平行

**R+** 树

—— 空间对象的 **MBR** 可能被非叶结点的 **MBR** 分割

**R+** 树的特点：

对于中间结点，每个项（ **I**，**child-pointer** ），当且仅当 **R** 被 **I** 覆盖时，以 **child-pointer** 指向的结点为根的子树，包括一个矩形 **R**。当 **I** 为一个叶结点的矩形时，**R** 只与 **I** 交叠

对于中间结点，任何 **2** 个结点（ **I1**，**child-pointer1** ）和（ **I2**，**child-pointer2** ），**I1** 与 **I2** 之间的交叠为 **0**—— 中间结点的所有矩形不相交

根至少有 **2** 个结点，除非它是叶结点

所有叶结点在同一层上

**44、什么是查询优化器？查询优化器所承担的主要任务是什么？**

答：查询优化器是数据库软件中的一个模块，它用于产生不同计算计划并确定适当的执行策略。主要任务：逻辑转换、动态规划。

**45. 查询语言与查询树之间的互换由什么执行？**

答：由语法分析器执行

**46. 对查询树进行逻辑转换的目的和一般方法是什么？**

答：方法：将非空间的选择和投影操作下推 目的：减少连接操作所涉及的关系大小，从而减少计算代价。



47、简述 SQL sever 2008 的安全机制。

第一层是 SQLServer 服务器级别的安全性，这一级别的安全性建立在控制服务器登录账号和密码的基础上，即必须具有正确的服务器登录账号和密码才能连接到 SQL Server 服务器。

第二层安全性是数据库级别的安全性，用户提供正确的服务器登录账号和密码通过第一层的 SQLServer 服务器的安全性检查之后，将接受第二层的安全性检查，即是否具有访问某个数据库的权利。

第三层安全性是数据库对象级别的安全性，用户通过了前两层的安全性验证之后，在对具体的数据库安全对象（表，视图，存储过程等）进行操作时，将接受权限检查，即用户要想访问数据库里的对象时，必须事先被赋予相应的访问权限，否则系统将拒绝访问。

48、什么是存储过程？什么是触发器？二者之间有哪些联系与区别？

存储过程：一组为了完成特定功能的 SQL 语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。在 SQL Server 中存储过程分为两类：即系统提供的存储过程和用户自定义的存储过程。

触发器（Trigger）是 SQL Server 提供的除约束之外的另一种保证数据完整性的方法，它可以实现约束所不能实现的更复杂的完整性要求。DML触发器可以分为两种：After 触发器和 Instead of 触发器。触发器是一种特殊的存储过程，它不允许带参数，不能由用户直接通过名称调用，而是由用户的某一动作自动触发。

49、三代地理数据信息及优缺点？

### 1、CAD 数据模型

优点：

缺点：不能存储足够多的属性信息，地图图层和注记标注是基本的属性描述

### 2、Coverage 数据模型

优点：**Coverage** 数据模型的优势是用户可以自定义要素表格；不仅可以添加字段并且还可以建立与外部数据表格的关联。

缺点：**Coverage** 数据模型有个重大缺陷——要素是以统一的行为聚集的点、线和面的集合。也就是说，表示道路的线的行为和表示溪流的线的行为是一模一样的——显然，这并不是我们所需求的。

### 3、Geodatabase数据模型

优点：与过去的数据库模型相比，其最大的特点是 **geodatabase** 更加智能化，每个要素不再仅仅是一条有几何字段的记录，而是一个拥有属性和行为的对象，是一个基于面向对象模型的关系数据库（对象——关系数据库）

缺点：

50、Geodatabase 的概念？包含 Geodatabase 的内容包含哪些？

Geodatabase 是一种采用标准关系数据库技术来表现地理信息的数据模型。Geodatabase 支持在标准的数据库管理系统（DBMS）表中存储和管理地理信息。Geodatabase 支持多种 DBMS 结构和多用户访问，且大小可伸缩。

（1）要素类（Feature Class）：是具有同样几何类型和属性的要素集合；——矢量图层。

（2）对象类（Object）：是 Geodatabase 数据模型中存储数据库表；——表

- ( 3 ) 要素数据集 ( Feature Database ) : 是有相同空间参考的要素类的集合。
- ( 4 ) 表 : 同纯关系数据库中的表。
- ( 5 ) 子类 ( subtype ) : 在要素类内部可以划分若干个次一级的组 , 每个组是一个子类。每个子类有其自己的完整性规则和 GIS 行为。
- ( 6 ) 关系 ( relationship ) 是一种表 ( 或要素类 ) 与表 ( 或要素类 ) 之间的关系 , 通过建立这些关系类 , 可以改善数据库查询机制 , 提高数据查询检索效率。
- (7) 拓扑关系 ( topology ) : 拓扑关系将参与拓扑的各个要素类集成在一个拓扑图中作为一个拓扑单元来管理 , 规定同一个要素类中的各个要素如何与其他要素共享几何 , 或者不同要素类之间如何共享几何。
- ( 8 ) 几何网络 : 各个要素类作为一个整体参与到几何网络的构造 , Geodatabase 通过拓扑关联保证参与到几何网格中的各个要素类的空间几何的连通性。几何网络将导致保证网络连通性的行为。
- ( 9 ) 栅格数据集 : 影像
- ( 10 ) 元数据 : 对数据中各个数据元素的描述。

51、ADO.NET的两个组件是什么？ .NET Framework 数据提供程序的四个对象及其用途。

两个组件： DataSet 和 .NET 数据提供者。

SqlConnection 、 sqlCommand SqlDataReader 、 sqlDataAdapter 。

- 1)Connection: 建立与特定数据源的连接。
- 2 ) Command对数据源执行数据库命令 , 用于返回数据 , 修改数据 , 运行存储过程以及发送和检索参数信息等。
- 3 ) DataReader: 从数据源种读取只进且只读的数据流。
- 4 ) DataAdapter : 执行 SQL命令并用数据源填充 DataSet 。DataAdapter 提供连接 DataSet 对象和数据源的桥梁。 DataAdapter 使用 Command对象在数据源中执行 SQL命令 , 以便将数据加载到 DataSet 中 , 并使 DataSet 中的数据更改与数据源保持一致。

52、以长沙市或实习作业的地形图为例 , 以书上的例子为参考 , 建立一个空间数据库模式 , 要求通过分析建立空间扩展的 E-R 图 ,然后把实体和联系映射到关系数据库中 ,每个模式要求达到第三范式。

选择题	填空题	名词解释	简答或应用题	写程序题	论述题
-----	-----	------	--------	------	-----

10 题	6 题 20 空	4 题	3 题	2 大题 7 小题	1 题
10 分	20 分	12 分	23 分	20 分	15 分