第六讲

空间数据索引技术与空间查询语言

- 空间数据存储技术
- 空间数据索引技术
- 空间查询语言

概述

回忆数据库设计的三个步骤：

- ◆概念模型
- ◆逻辑模型
- ◆物理模型

对这三个模型，可以用交通工具来类比：

概念模型：

逻辑模型：

物理模型：

数据库物理模型

什么是数据库物理模型？

使用计算硬件、操作系统软件，以一种高效而稳健的方式来实现逻辑模型的文件组织。

通过对理解物理模型的理解，能够有助于我们改善查询效率。例如，如果我在查询时感觉速度很慢，这时可以考虑建立索引以提高查询速度。

关系数据库物理模型

- **关系数据库的物理模型：➢**
- 采用外存储设备——磁盘来保存数据
- 使用有序的文件结构
- 采用辅助的层次树状索引来加快数据的访问
- 数据类型：数字、字符串、日期、货币等；
- **所有数据类型都是线性可排序的**
- 操作：在一维线性排序数据上进行检索插入、删除等

空间数据库物理模型

- 从空间数据库中获取数据的有效方法，是通过索引(Index)。索引和数据库的物理模式密切相关。
- 在关系数据库中，广泛采用B＋树索引，即使是从一百亿条记录的数据库集合中查找一条与给定条件匹配的记录，也可在几分之一秒内完成。
- 传统的B＋树索引难以达到快速查找空间目标的目的，必须要研究专门的空间索引技术。空间索引一般是采用近似的方法，如空间对象的外接矩形、空间格网等

空间数据库物理模型

常用空间查询：

- 点查询： 找出所有包含给定点的矩形
- 范围查询： 找出所有位于给定矩形内的点
- 最近邻居： 找出距查询点最近的点
- 距离扫描：按给定点距离的增序列出所有点
- 相交查询：找出所有与给定矩形相交的矩形
- 包含查询：找出所有完全包含在给定矩形中的矩形
- 空间连接查询：找出所有相互交叠的矩形

空间数据存储：磁盘文件

计算机中存储架构

- 计算机的主要组件：
- **CPU**
- 输入输出设备,如鼠标、键盘、显示器、打印机等
- 通讯部件，如内部总线、网卡、调制解调器
- 存储架构

存储设备类型

- 主存——速度很快（纳秒级），但关电源后存储内容丢失

空间数据存储：磁盘文件

计算机中存储架构

- 第二存储设备——速度慢（微秒级），但存储内容不受电源的影响
- 第三存储设备——速度更慢，存储内容不受电源影响，往往存储容量非常大。

数据库管理数据的方式：

- 在第二存储设备上存储数据
- 用主存储设备改进性能
- 用第三存储设备备份数据

空间数据存储：磁盘文件
计算机中存储架构

- 传统**DBMS**、应用程序和**SDBMS**在**CPU**和**I/O**相对代价特征(Input /Output)

	CPU代价	I/O代价
DBMS	低	高
C程序	高	低
SDBMS	高	高

磁盘的几何结构

- **磁盘概念**
- 盘片：物理圆形磁片
- 磁道：盘片上的同心圆
- 柱面：所有盘片上的直径相同的磁道
- 扇区：组成页面的最小单位
- 页面：磁盘和主存间的最小传输单位，一般为扇区的整数倍

磁盘的几何结构

磁盘驱动器概念

- ➢磁头读写数据
- ➢每个盘片有自己的磁头
- ➢磁头机构同时在径向上移动磁头
- ➢盘片沿一轴心高速旋转

磁盘的几何结构

- **存取时间**
- 寻道时间：移动磁头到相应磁道上的时间**ts**
- 延迟时间：要读取页面旋转到磁头下的时间**tl**
- 传输时间：磁头读或写相应数据的时间**tt**

总时间：**ta=ts+tl+tt**

这几个时间通常满足下列不等式：

ts>tl>tt

tt一般在初始时固定，但ts和tl可能通过优化策略缩短。

磁盘的几何结构

- *Western Digital Cavier AC36400*磁盘驱动器参数

格式化容量	6448MB
柱面数	13328
每磁道扇区数	63
每个扇区字节数	512
盘片数	3个双面
寻道时间	9.5ms
延迟时间	5.5ms

磁盘的几何结构

影响磁盘读写数据效率的因素：

- 扇区大小
- 扇区越大传输大数据速度越快
- 但扇区越大，存储空间和浪费越多
- 数据在盘片上的存放位置
- 在盘片中间位置比盘片内或外沿的读写速度要快
- 对于大数据块，尽量存放在同一柱面上
- 原因：减少了平均寻道时间

磁盘的软件结构形式

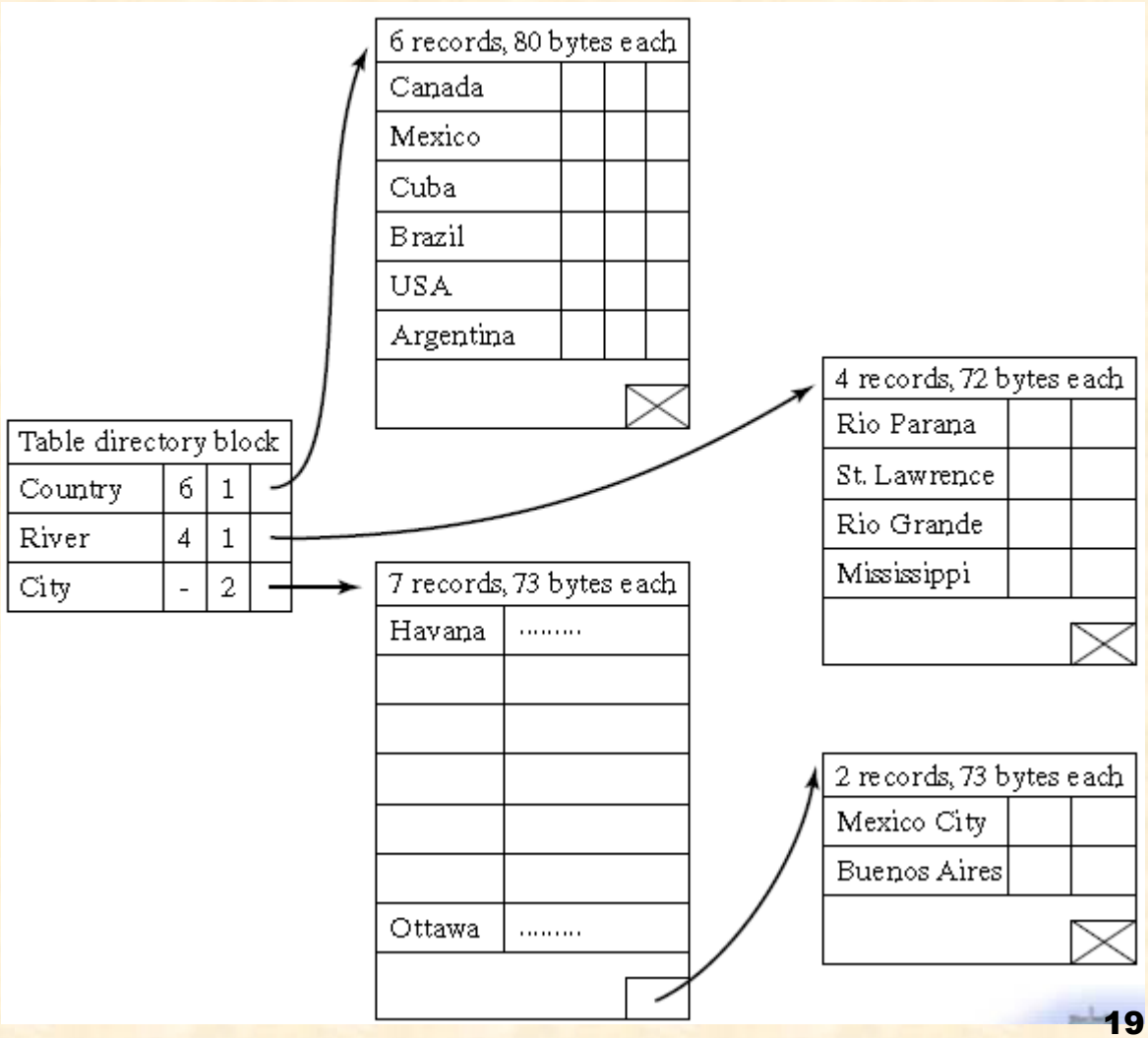
- 从软件的角度，数据在磁盘上以域、记录、文件这种层次结构的形式存放的。
- 域表示一个关系表或实体属性
- 记录则对应于关系表的一行，即一个实体，是属性的集合
- 文件，类似于整个关系表，是记录的集合。

记录、文件到磁盘的映射

- 在**Country**表中，记录的大小：
- **size(name)+size(cont)+size(pop)+size(GDP)+size(life-exp)+size(shape) ≤30+30+4+4+4+8=80**
- 这里假定整型分配**4**个字节，指针分配**8**个字节，为简化期间，将**varchar(N)**直接分配**N**个字节。类似的，**River**每条记录占用**72**个字节，**City**每条记录占用**73**个字节。

记录、文件到磁盘的映射

通常一条记录比扇区小，多个记录存放在一个扇区中，一个文件包含多个扇区。如右图，Country和River各占一个扇区，City占用两个扇区。这里假定一个扇区为512字节。



记录、文件到磁盘的映射

- **二进制大对象（BLOB）**：在关系数据库中将复杂数据类型转换成BLOB存储。如Oracle中的LONG RAW数据类型。
- 对复杂类型管理
- 支持事务操作
- 不支持查询操作

缓冲区管理

使用缓冲区管理的目的：

- 计算机读取内存的速度要要比读取磁盘快的多
- 思想：将需要重复读取的数据，读入内存，即缓冲区，后续读取该数据时直接在内存中读取。
  - ✓显著提高查询时间
  - ✓减少磁盘的物理读取

缓冲管理器是DBMS中的一个模块，它的主要功能：

- 哪一个扇区留在内存中？
- 如果内存缓冲池已满时，哪个缓冲区被移走？
- 什么时候将扇区在单用户查询前放于内存中？

常用算法有LRU算法。

缓冲区管理

LRU算法

所谓的LRU(Least recently used)算法的基本概念是:当内存的剩余的可用空间不够时,缓冲区尽可能的先保留使用者最常使用的数据,换句话说就是优先清除”较不常使用的数据”,并释放其空间。之所以“较不常使用的数据”要用引号是因为这里判断所谓的较不常使用的标准是人为的、不严格的。所谓的MRU(Most recentlyused)算法的意义正好和LRU算法相反

缓冲区管理

- 以Oracle9i为例，使用者查询数据A,初始的时候LRU List中没有数据A,于是Oracle 9i到磁盘读取A,然后放到LRU List的MRU端,使用者再从LRU List中读取数据A,同理对于B,C…当LRU List满了以后,如果使用者查询N,此时N不在LRU List中而且LRU List中已经没有freebuffer了,此时Oracle 9i就开始从LRU端淘汰A以腾出空间存放N。

文件结构

文件结构：

一种组织文件中记录顺序的方法，以便于对文件的各种操作，常用文件结构包括：

- 无序（堆）文件
- 顺序文件
- 散列文件
- 聚类文件（用于空间数据存储）

常用文件操作：

查找：关键字→相匹配记录值

查找下一个： 返回当前记录值相邻的下一个记录

插入：在不改变文件结构的情况下增加一条新记录

最近邻居：一个空间对象的最近相邻对象

文件结构

常用文件结构

无序文件（或堆－heap）：非常方便插入，常用于日志文件，但查找比较费时

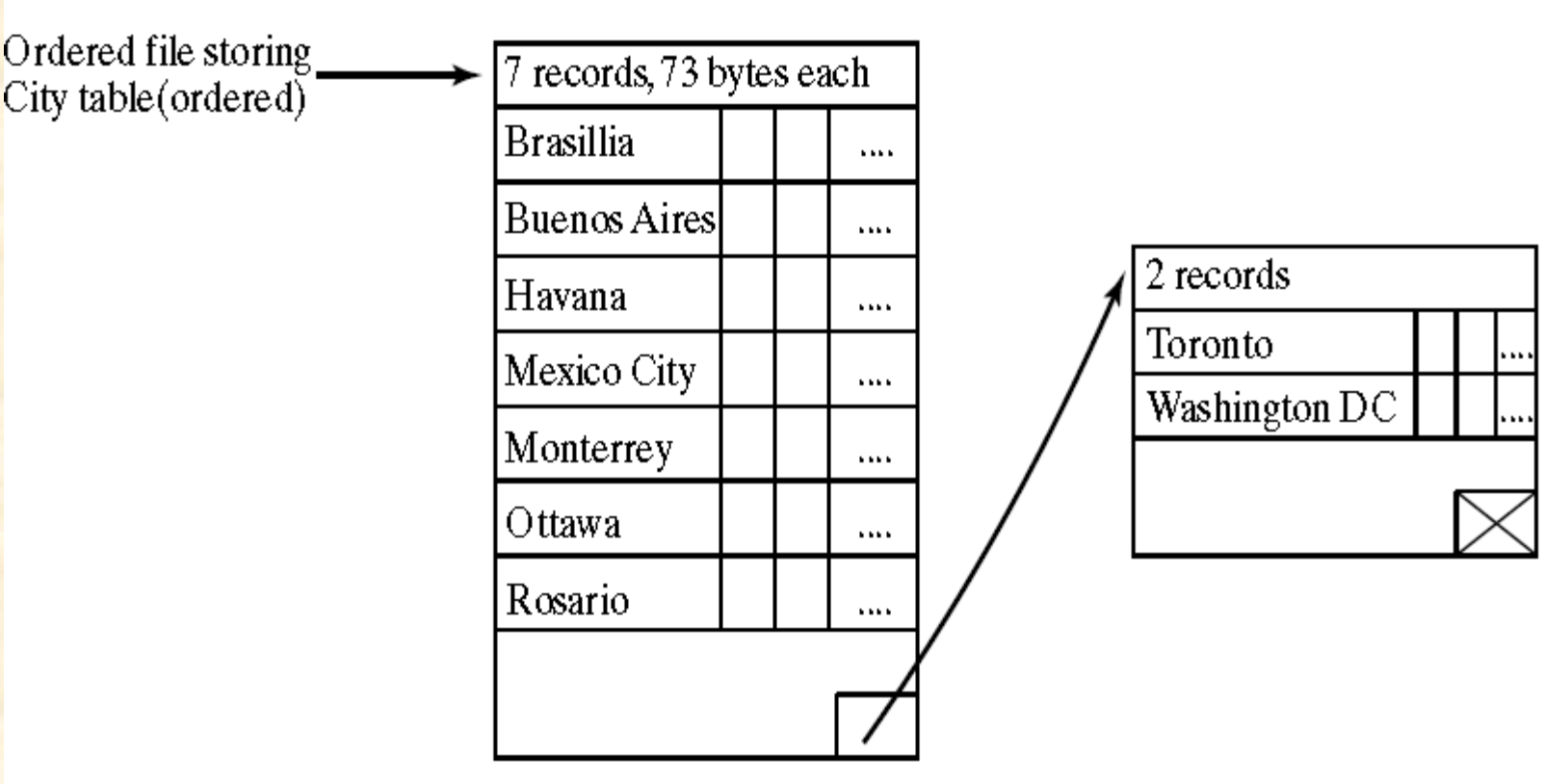
- 记录存放没有特殊顺序
- 插入可以通过简单的将记录置于文件最后扇区完成
- 查找、查找邻近操作需要扫描整个文件

文件结构

顺序文件结构：

- 记录按某一属性值排序存放
- 查找邻近记录可以简单拾取物理上的下一个记录完成
- 查找、插入和删除通过二分查找，也非常有高效
- 空间上的最近邻查找需要检索一定范围内的记录

文件结构


Ordered file storing
City table(ordered)
7 records, 73 bytes each
Brasillia
Buenos Aires
Havana
Mexico City
Monterrey
Ottawa
Rosario
....
2 records
Toronto
Washington DC

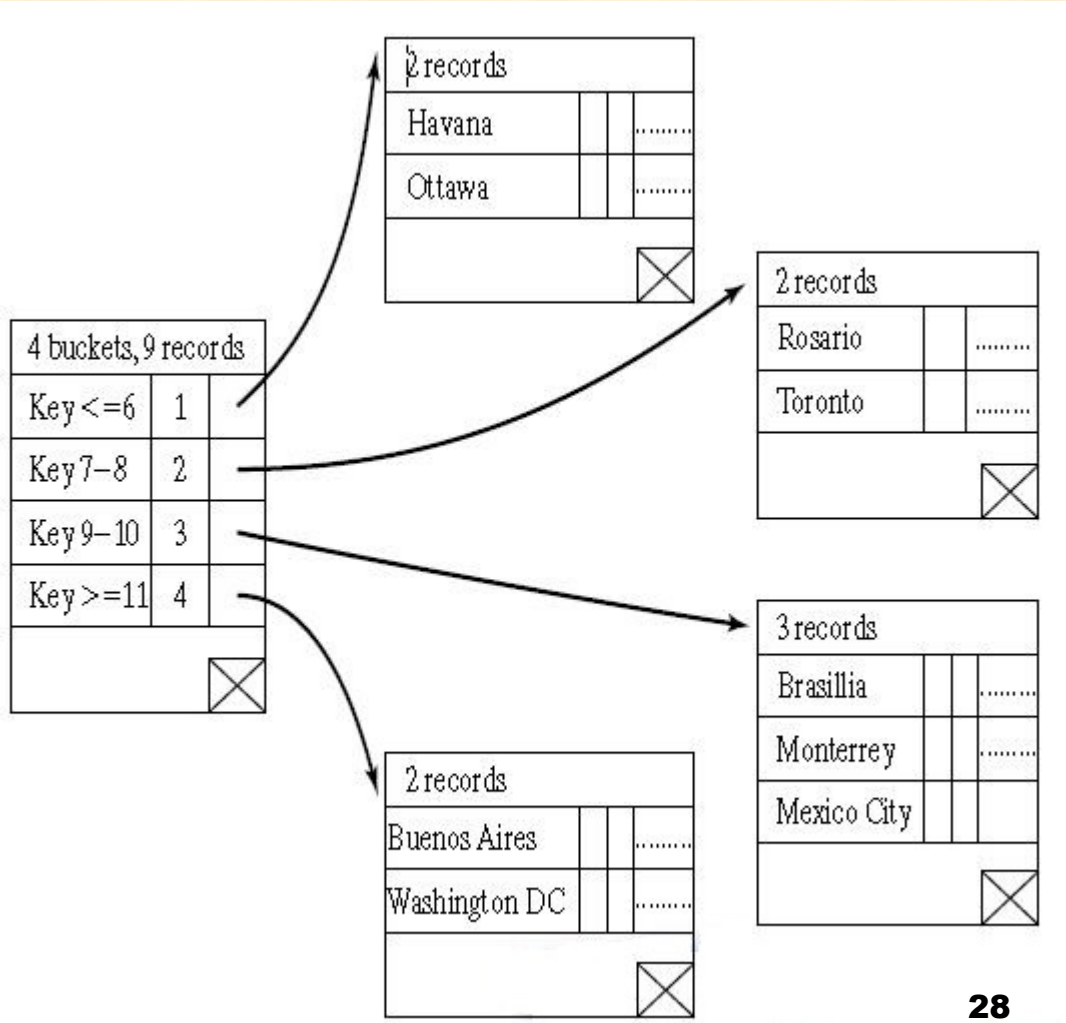
文件结构

散列文件结构：

- 采用简单的算法将主码映射到散列单元，主码à散列单元(桶）
- 散列单元à扇区地址

操作：

- 查找、插入、删除非常高效
- 查找邻近目标和查找下一个较慢



空间数据文件结构—聚类

聚类的目的就是降低响应大查询的寻道时间（ts)和等待时间（tl），对于空间数据库，即要求“**空间上相邻和查询上有关联性的对象在物理上存储在一起**”。空间数据库支持有三种聚类：

内部聚类：一个对象的全部表示存储在同一个磁盘页面中。

本地聚类：一组空间对象（或近似）被分组到同一个页面上。

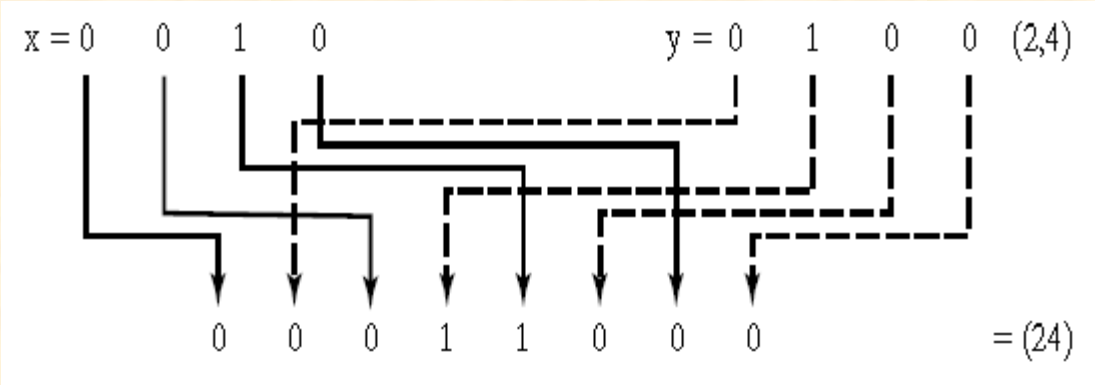
全局聚类：一组空间对象存储到多个邻接的页面上。

空间数据文件结构—聚类

- 问题：
- 空间数据所处的多维空间中没有天然的顺序
- 存储磁盘从逻辑上说是一维的设备
- 空间聚类技术就是要寻找一个从高维空间向一维空间的映射方法，空间上邻近的元素，映射为直线上接近的点，而且一一对应为达到这一目的，人们提出了很多种算法，其中较有代表性的有：Z曲线和Hilbert曲线

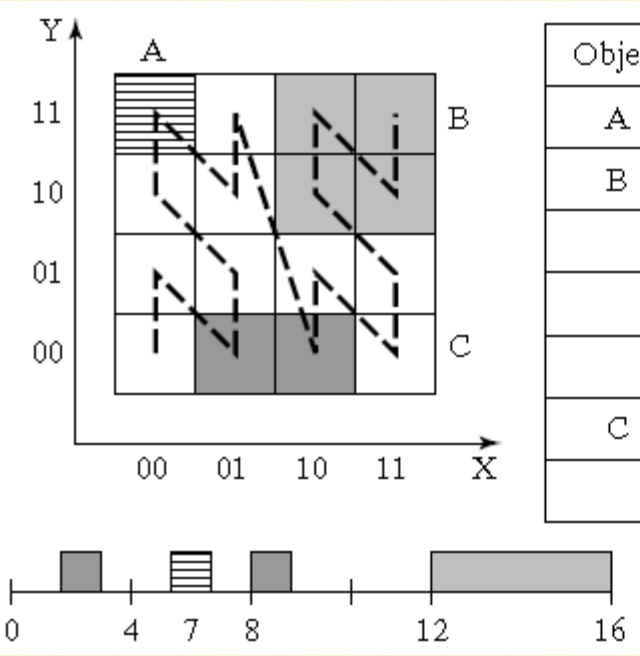
空间数据文件结构—聚类

- **Z**曲线：
- ➢读入**x**、**y**坐标的二进制值
- ➢隔行扫描二进制数字的比特到一个字符串
- ➢计算结果二进制串的十进制数



空间数据文件结构—聚类

A、B、C对象的Z曲线



Object	Points	x	y	interleave	z-value
A	1	00	11	0101	5
B	1				
	2				
	3				
	4				
C	1	01	00	0010	2
	2	10	00	1000	8

空间数据文件结构—聚类

Hilbert曲线

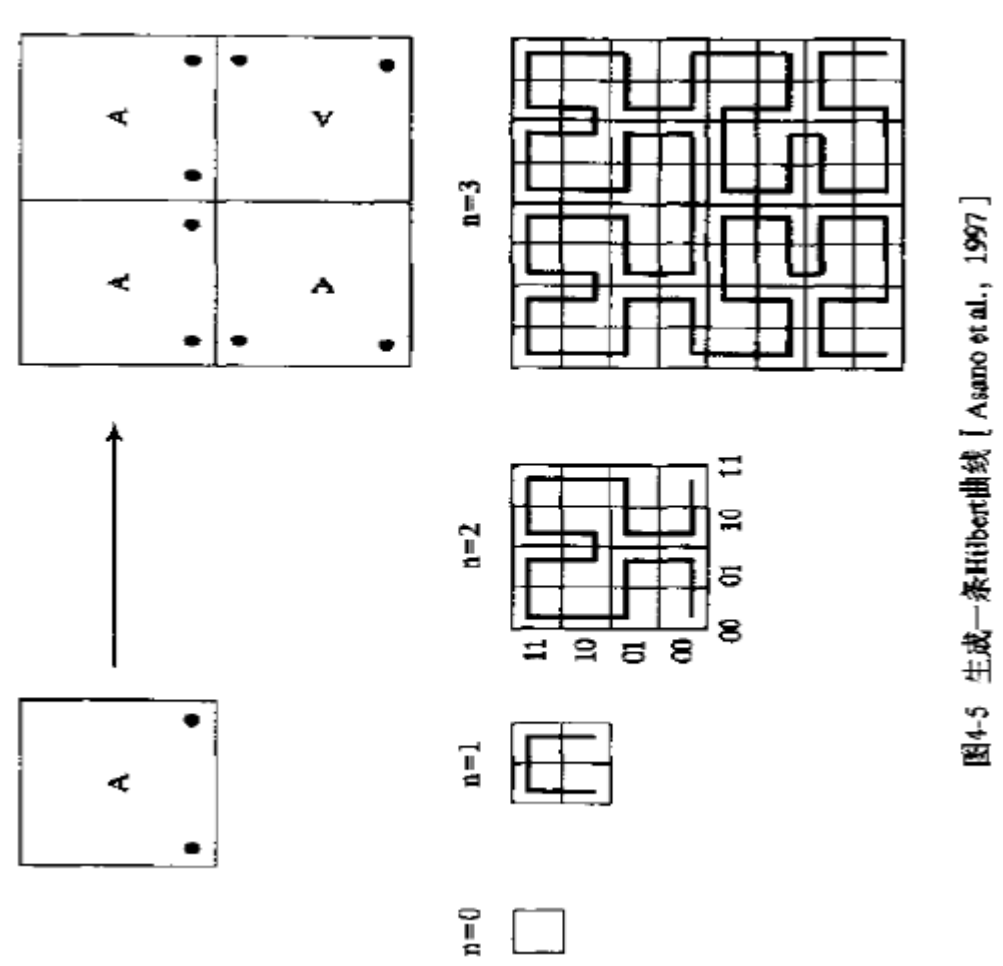


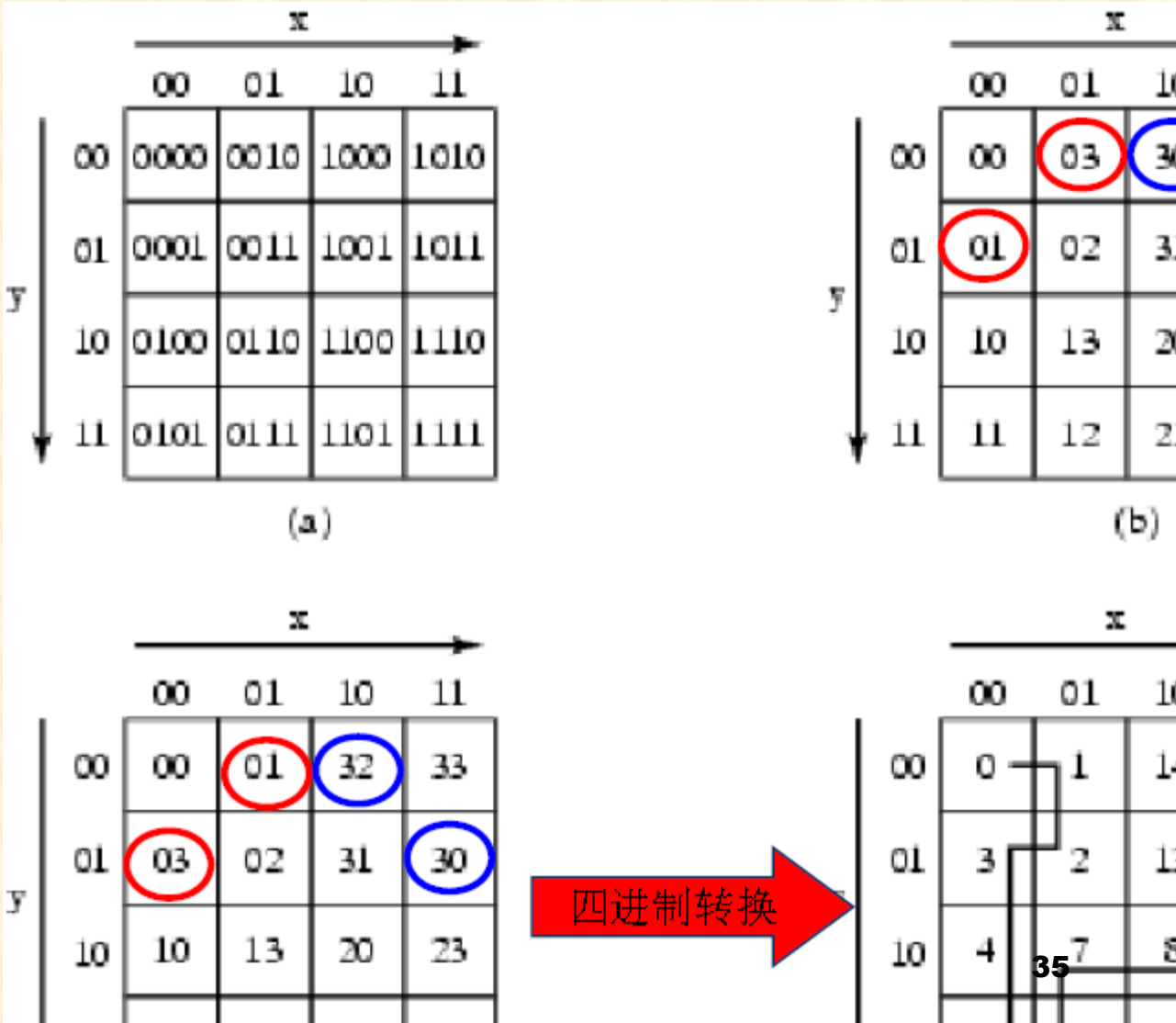
图4-5 生成一条Hilbert曲线[Asano et al., 1997]

空间数据文件结构—聚类

- **Hilbert曲线：**
  - 读入x、y坐标的二进制值
  - 将二进制形式的X 、Y 按位相互交叉, 构成一个长度为2n的二进制串S
  - 将字符串自左至右分成2个比特长的串Si，i=1,2，……，n
  - 规定每2个比特的十进制值di，如“00”等于0，“01”等于1，**“11”等于2，“10”等于3**
  - 对于数组中每个数字j，如果j＝0，把后面的所有1变成3,3变成1；若j=3，则把后面的所有0变成2,2变成0
  - 将数组中每个值按上步换成二进制，自左至右连接所有串，并计算其十进制值。

空间数据文件结构—聚类

- Hilbert曲线



空间数据文件结构—聚类

磁盘访问的度量：

假定每个点都对应一个网格单元，曲线为每个单元指定一个整数值。对一个给定查询，这里采用聚类在给定查询代表的子空间中每个网格点的散列平均数。

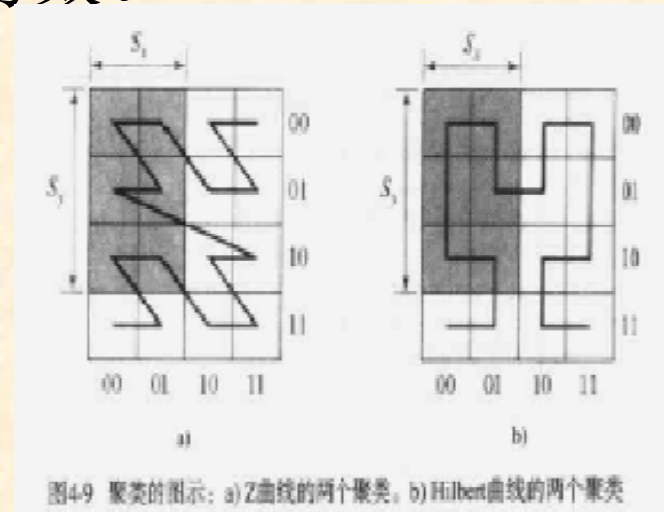


图4-9 聚类的图示：a) Z曲线的两个聚类，b) Hilbert曲线的两个聚类

- 对于a)2~3,8~11两个散列单元
- 对于b)2~7一个散列单元