Географические данные в компьютере

Чтобы воспользоваться преимуществом возможностей компьютера для пространственного моделирования, обязательно надо понимать, как модели данных закодированы в нем.

Компьютеры кодируют данные и выполняют команды, используя серию переключателей, которые существуют в одном из двух состояний "да" или "нет". Эти состояния закодированы числами 1 и 0, соответственно и основаны на бинарной системе - это основа для вычисления. Поэтому географические данные должны быть преобразованы в двоичные записи в компьютере, чтобы представить явления.

Главные структуры, используемые для данных, названы вектором и растром, они делят пространство в ряд точек, линий и полигонов, или в правильную мозаику.

Введенные в компьютер данные должны быть организованы, чтобы обеспечить эффективный доступ, поиск и манипуляцию. Системы для организации данных в компьютере ранжируются от простых списков до высоко структурированных баз данных, основанных на иерархическом, сетевом, реляционном принципах. Различные варианты налагают ограничения на представление и обработку данных.

Компьютерная карта может быть цифровым изображением бумажного документа, сохраненного на запоминающем устройстве, или это может быть интерактивное отражение информации в сложной ГИС.

Главная задача состоит в том, чтобы изобрести пути кодирования формализованных географических моделей данных так, чтобы компьютер мог обрабатывать их.

Топологические и пространственные аспекты географических данных отличают их от данных, используемых в других системах - банковских, библиотечных и др. Кроме того ГИС должна отобразить данные, использующие графические компоненты - форму, цвет, символы, тени и типы линий. Следовательно, результатом природы, объема и сложности географических данных, является разработка ряда логических компьютерных схем для эффективного хранения, модификации и поиска данных.

Пространственные модели и структуры данных

Создание аналоговых и цифровых пространственных наборов данных включает семь уровней развития модели и абстракции:

1. Представление реальности (концептуальная модель)
2. Человеческое осмысление, ведущее к аналоговой абстракции (аналоговая модель)
3. Формализация аналоговой абстракции без всяких ограничений на выполнение (пространственная модель данных)

4. Представление модели данных, которая отражает, как данные записаны в компьютере (модель базы данных)
5. Файловая структура, которая является частным представлением структуры данных в компьютерной памяти (физическая вычислительная модель)
6. Принятые аксиомы и правила для обработки данных (модель манипуляции данных)
7. Принятые правила и процедуры для отображения и представления пространственных данных людям (графическая модель)

Данные в компьютере

Люди представляют числа и выполняют арифметические действия, используя десятичную систему с основой 10. Каждая позиция в записанной форме числа указывает, сколько единиц 10 соответствуют той позиции. Компьютеры используют другую систему подсчета. Они сохраняют данные в форме массивов переключателей, которые имеют только два положения; они "включены", или они "выключены" и эти положения кодированы числами 1 и 0 соответственно. Поэтому компьютеры кодируют данные и делают вычисления, используя бинарную арифметику.

Двоичные числа и компьютерное представление чисел и текста

Данные в двоичной системе представлены рядом переключателей (бит). Переключатели - маленькие намагниченные области на компьютерном диске или памяти обычно группируются по восемь в пакет (байт).

Несколько байтов могут быть связаны вместе, образуя компьютерное слово. Слова сгруппированы вместе в записи данных, а записи сгруппированы вместе в компьютерный файл. Наборы компьютерных файлов могут быть сгруппированы вместе иерархически в каталогах или подкаталогах.

С байтом из 8 битов мы можем представлять 256 чисел от 0 до (2^8), то есть от 0 до 255. Например, последовательность восьми переключателей 11111111_2 есть:

$$2^7*1 + 2^6*1 + 2^5*1 + 2^4*1 + 2^3*1 + 2^2*1 + 2^1*1 + 2^0*1 = 255_{10}$$

Если мы объединяем 2 байта в 16-разрядное слово, то можно закодировать числа от 0 до 65535. Однако также возможно кодировать положительные и отрицательные числа так, только первые пятнадцать битов используется чтобы программировать числа, и шестнадцатый бит используется, чтобы определить знак. Это означает, что шестнадцатью битами мы можем закодировать числа от -32767 до +32767. Число без

дробного компонента называется целым числом, а числа в диапазоне -32767 до +32767 называются 16-разрядными целыми числами.

Часто необходимо закодировать числа, которые являются большими или меньшими чем от -32767 до +32 767, или числа, которые имеют дробные компоненты, необходимы компьютерные слова с большим количеством битов. Вещественные числа с положительными и отрицательными значениями и десятичными числами требуют 32-х или даже 64-х битных слов (так называемая двойная точность) в котором некоторые из битов зарезервированы для десятичной части и остальные используются для больших чисел. Этот метод кодирования чисел означает, что точность, с которой любое число может быть кодировано - функция количества используемых битов.

Организация данных и шестнадцатеричное кодирование

В компьютерах хранятся и текстовые символы. Одна из наиболее обычно используемых систем для кодирования алфавитно-цифровых данных известна как Американский Стандарт Компьютерных Информационных Индексов (ASCII) и основана на шестнадцатеричной системе.

Шестнадцатеричные числа могут быть представлены единичными символами (0-9) и буквенными символами заменяя числа 10,11, 12,13, 14, 15 символами A, B, C, D, E, F соответственно. Например: бинарное 10001, десятичное 182, шестнадцатеричное B6.

Шестнадцатеричная система также обеспечивает удобное основание для кодирования алфавитно-цифровых данных типа букв алфавита и чисел от 0 до 9. Так что коды ASCII для текстовых символов могут также быть написаны как шестнадцатеричные числа от нуля до FF. Например:
буква A кодируется как 41;
цифра 5 кодируется как 35;

И числа и коды для букв представлены строками битов, которые установлены в положение 0 или 1.

Система ASCII - не единственный способ кодирования текста. Организация битов определяется форматом; если формат не известен, информация не может быть извлечена из строки битов.

Тома данных

Тома битов, необходимых для кодирования информации выражены в тысячах (кило), миллионы (мега) или миллиарды (giga) байтов. Стандартная A4 страница, кодированная в ASCII с 64 строками текста, имеющего 80 символов в строке (включая пробелы) требует 40 960 битов (5 120 байтов или 5 k (1 k = 1024 байта); книга 200 страниц требует 1.024 Мегабайта для букв и дополнительное пространство требуется для информации

относительно номеров страниц, текстовых шрифтов, размещения, и т.д. Карты и другие формы пространственных данных требуют больших объемов памяти.

Кодирование основных типов данных для ввода в компьютер

Операции ввода данных - преобразование информации от формы, которую понимают люди, к форме, подходящей для числовой обработки в компьютере.

В ГИС используются дискретные стандартные блоки географических данных, способные к представлению объектов и непрерывных полей, а также способные представить точные или неточные атрибуты, местоположение и отношения. Основные пространственные единицы, используемые в представлении - это векторные (точка, линия и полигон) и растровые (пиксель) примитивы; с ними связаны разнообразные атрибутивные данные.

Для организации их хранения в компьютере, эти основные стандартные блоки могут быть формализованы в логические схемы (структуры данных), которые используются как для представления объектов, так и непрерывных полей.

Векторные структуры данных

Структура данных, которая использует точки, линии или полигоны для описания географических явлений, известна как векторная структура данных. Векторные единицы характерны фактом, что их географическое местоположение может быть установлено независимо и очень точно, как и их топологические отношения. Они однородны, их атрибуты относятся ко всему модулю.

Простых пространственных модулей недостаточны для эффективной обработки данных. Очень эффективно иметь возможность обрабатывать более сложные явления, как отдельные модули в информационной системе. Самый простой составной объект - "звено", или "дуга" который используется, чтобы представить изогнутую или непрямую линию. Когда линии объединяются, они должны быть связаны специальной точкой - узлом, чья функция должна нести информацию о способе, которым дуги связаны вместе и как движение может идти от одного к другому. Если пересечение линий не имеет этого, компьютер не может делать различия между мостом, тоннелем или перекрестком.

Узлы также используются, чтобы нести информацию о том, как граничные дуги полигонов соединены и обеспечивают информацию о левых и правых соседних полигонах и островах. Атрибуты векторных модулей хранятся в компьютерных файлах. Типичные операции с геометрическими данными комбинируют слои векторной информации, чтобы получить

ответы на логические запросы или определять количественные отношения (геометрические или топологические) между различными модулями в пересекаемых слоях. Операции, основанные на атрибутах, позволяют искать определенный набор значений, или вычислять новые из существующих данных.

Ячейка сетки: растровые структуры данных

Пространственные явления могут также быть представлены наборами правильных мозаичных модулей. Самая простая форма - квадратная ячейка (пиксель), мозаичная правильная сетка известна как растровая структура данных. Расположение объектов определено прямой ссылкой на сеть сетки с каждым пикселем.

Данные структурированы и компьютер кодирует отношения между размером пикселя и ячейкой на земле. В векторных данных структуры явно зарегистрированы через указатели баз данных, в растровых базах данных - неявное кодирование через величину атрибута пикселя.

Изменения явлений возможно представить в мозаике через различные реально-оцененные номера пикселей; каждый атрибут описывается отдельным слоем. В более сложных растровых структурах (грид), значения ячейки могут быть связаны с записью, несущей значение многих отдельных атрибутов.

Типичные операции включают логические операции и запрос соседних областей.

Структура БД: организация данных в компьютере

Перед рассмотрением подробно способов, которыми пространственные объекты могут быть сохранены эффективно в компьютере, мы должны сначала рассмотреть общие проблемы организации данных для оптимального хранения и доступа. Хотя это - не обязательно для пользователей ГИС (водить можно и без знания двигателя внутреннего сгорания), но полезно для понимания возможностей и ограничений системы.

Файл и доступ к данным

Обязательные особенности любой системы хранения данных - то, что она должна позволить быстро обращаться к данным и перекрестно ссылаться. Есть несколько путей достижения этого, некоторые из которого более эффективны чем другие. К сожалению, нет единственного "лучшего" метода, который может использоваться во всех ситуациях.

Простые списки

Самая простая форма базы данных - простой список всех элементов. Каждый новый элемент добавляется к концу списка, файл становится все больше и больше. Очень просто добавлять данные в такой системе, но поиск неэффективен. Так, для информационной системы, содержащей 10000 описаний элементов на карточках, учитывая, что требуется 1 секунда, чтобы прочитать имя карточки или номер, потребуется полтора часа, чтобы найти нужную карточку.

Упорядоченные последовательные файлы

Слова в словаре или имена в телефонной книге структурированы в алфавитном порядке. Добавление нового элемента означает, что должен быть создан дополнительный участок памяти, чтобы его вставить, но преимущества - быстрый доступ. Вместо поиска с начала списка, запись ищется в середине. Если ключевое значение (например, последовательность букв в слове) совпадает, тогда средняя запись - то что мы ищем. Если значения не соответствуют, легко увидеть, находится ли требуемый элемент перед или после среднего элемента. Повторный поиск идет в нужной половине файла. Если в файле 10000 элементов и время поиска - элемент в 1 секунду, среднее время поиска составит приблизительно 14 секунд.

Индексированные файлы

В простых последовательных и упорядоченных файлах данные находятся по ключевому атрибуту. В случае словаря, ключевой атрибут - запись по буквам. Но в многих прикладных программах, особенно в ГИС, индивидуальные элементы (пиксели, точки, линии или полигоны) имеют не только ключевой атрибут, но также несут информацию о связанных атрибутах. Для примера, мы имеем упорядоченный список профилей почвы, который был структурирован по названию почвы, но мы хотели бы отыскать информацию о мощности почвы, дренаже, pH, структуре, или эрозии. Если мы не примем другую стратегию базы данных, наши процедуры поиска возвращаются к простому последовательному списку.

С индексированными файлами, поиск данных может быть ускорен. Если элементы данных в файлах сами обеспечивают главный порядок файла, то это прямой доступ. Расположение элементов в главном файле может также быть определено согласно теме, которая дана во втором файле, называемом индексированный файл.

Поиск осуществляется последовательным поиском индекса, с последующим последовательным поиском соответствующего блока данных.

Индексированные файлы разрешают быстрый доступ к базам данных. К сожалению, им сопутствуют проблемы, когда записи непрерывно добавляются или удаляются, что часто случается с интерактивными

системами отображения. Добавление или стирание записи в файле прямого доступа означают, что и файл и его индекс должны измениться.

Структура баз данных и управление базами данных

Пространственные базы данных, содержат много файлов с данными относящихся к сходным объектам, или данных относительно объектов, которые из-за их пространственной близости или связности должны быть связаны или сгруппированы вместе. Важно организовать способ, которым эти файлы сохранены и связаны в компьютере, чтобы моделировать реальные мировые явления и гарантировать эффективное хранение и поиск данных. Компьютерная программа, которая предназначена, чтобы сохранять и управлять большими количествами данных, называется системой управления базы данных (СУБД).

Современные СУБД используют много методов для эффективно хранения и поиска данных, но все они основаны на трех фундаментальных принципах организации информации, которые отражают логические модели: они известны как иерархические, сетевые, и реляционные. Все они используются в ГИС. В последнее время появился четвертый тип СУБД, который называется Объектно-ориентированный. Это дальнейшее развитие сетевой модели.

Моделирование Базы данных - задача проектирования базы данных, которая работает эффективно, содержит правильную информацию, имеет логическую структуру и проста, насколько это возможно. Чтобы понимать, как это может быть достигнуто в ГИС, сперва необходимо исследовать фундаментальные принципы различных организационных структур.

Иерархическая структура базы данных

Когда данные имеют родительско-дочернее отношение или отношение "один ко многим", типа областей различного уровня администрирования, род почвы в пределах семейства почв или пикселей в пределах полигона, иерархические методы обеспечивают быстрые и удобные средства доступа к данным. Иерархические системы организации данных известны в естественных науках. Они используются в таксономии растений и животных, классификаций почв и так далее. Они предполагают, что каждая часть в иерархии может быть получена, используя набор отличительных критериев, полностью описывающих структуру данных.

Записи данных

Во всех видах структур баз данных, данные находятся форме записей. Самый простой вид записи - одномерный массив установленной длины, разделенной на ряд равных разделов. Эта форма идеальна, когда все

элементы имеют одинаковое число атрибутов, например, когда ряд профилей почв был исследован стандартным способом.

Фиксированная длина записи неудобна, особенно когда атрибуты имеют разную длину, и разный набор измеренных показателей. Например, не все профили почвы имеют одинаковое число горизонтов, и не все границы полигонов имеют одинаковое число координат. В таких ситуациях используются записи переменной длины. Каждая запись имеет "заголовок", дополнительный атрибут, который содержит информацию о типе информации в подразделе и его размере (длине). Серия записей составляет файл.

Главные преимущества модели - ее простота и легкость доступа через ключи, которые определяют иерархию. Ее просто понять, модифицировать. Она полезна для организации данных в запоминающих устройствах большой емкости. Доступ к данным через ключевые атрибуты прост и быстр, но к сожалению очень затруднен для связанных атрибутов. Следовательно, иерархические системы хороши для поиска данных, если структура всех возможных запросов известна заранее. Это обычно имеет место с библиографическими, банковскими системами.

Данные об окружающей среде, однако, носят исследовательский характер и не приспособлены к твердой иерархии, поэтому данная структура малопригодна.

Сетевая структура базы данных

В иерархических структурах, путешествие в пределах базы данных ограничено путям вверх и вниз по таксономическим путям. Во многих ситуациях требуется более быстрое редактирование, особенно в структурах данных для графических элементов, где смежные элементы в карте должны быть связаны вместе, при этом фактические данные об их координатах могут быть написаны в различных частях базы данных.

Как избыточность, так и проблемы связности преодолены в компактной сетевой структуре, в котором каждая линия и каждая координата появляются только однажды.

Сетевые структуры базы данных очень полезны, когда отношения или связи могут быть определены заранее. Они избегают избыточности данных и хорошо используют имеющиеся данные. Недостатки - то, что база данных увеличивается в размере на верхний из указателей, который в сложных системах может стать существенной частью базы данных. Эти указатели должны поддерживать изменение в базе данных.

Реляционная структура базы данных

В самой простой форме реляционная структура базы данных не хранит никаких указателей и не имеет никакой иерархии. Вместо этого,

данные сохранены в простых записях (кортежах), которые являются наборами полей, каждое из которых содержащие атрибут; кортежи сгруппированы вместе в двумерных таблицах, известных как отношения, напоминающих электронные таблицы. Каждая таблица или отношение - обычно отдельный файл. Структуры указателей в сетевых структурах и ключи в иерархических структурах заменены данными в форме идентифицирующих кодов, которые используются как уникальные ключи, чтобы идентифицировать записи в каждом файле (Рисунок 3.2а).

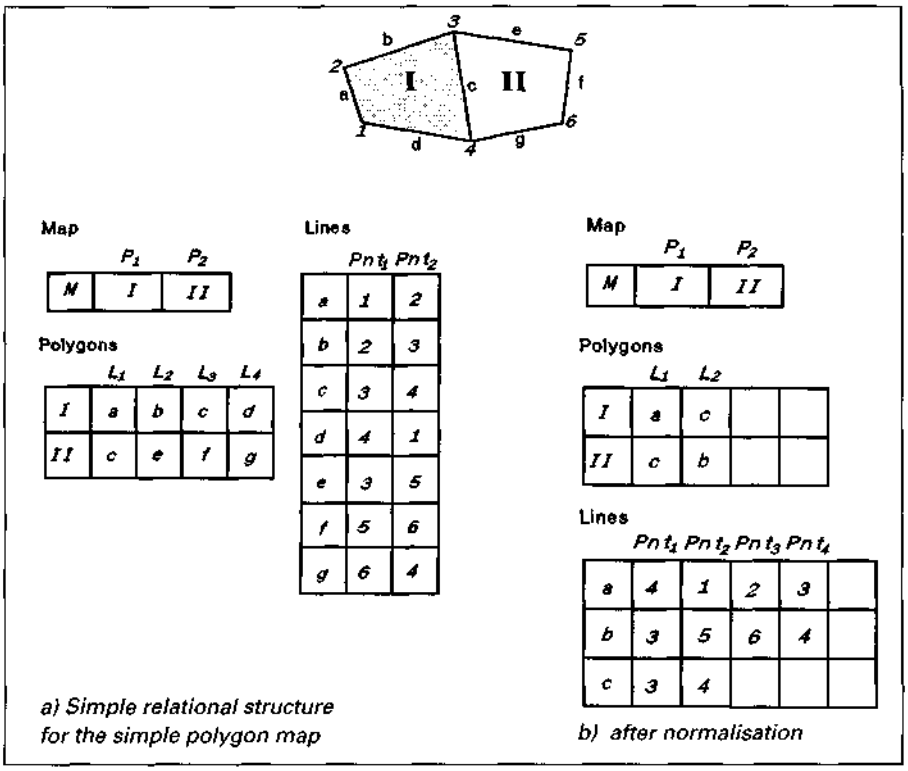


Рисунок 3.2. Реляционная организация векторных данных

Данные извлекаются из реляционной базы данных определением отношения, которое является соответствующим запросу. Это отношение не обязательно уже представлено в существующих файлах, программа управления использует методы относительной алгебры, чтобы создать новые таблицы. Эти правила часто кодируются в так называемом Языке структурированных запросов (SQL).

Реляционные базы данных имеют большое преимущество, так как их структура очень гибка и может выполнять все запросы, которые могут быть сформулированы, используя правила Булевой логики и математических операций. Они позволяют отыскивать, объединять и сравнивать различные виды данных. Добавление или удаление данных также просто, это добавление или удаление кортежей, или даже целых таблиц.

Поперечный запрос различных реляционных таблиц производится при соединении их через общие поля. Это хорошо в том случае, когда все записи имеют те же самые номера атрибутов, и нет никакой естественной иерархии.

Однако, если отношения между таблицами сложны, и необходим ряд объединений, операции займут много времени даже на быстрых компьютерах. Следовательно, реляционные системы базы данных должны быть очень умело разработаны, чтобы поддерживать возможности поиска с разумным быстродействием.

Системы управления базами данных

Системы управления базами данных (СУБД) - компьютерные программы для организации и управления базой данных, они могут быть созданы, используя любой структуры или комбинацию: иерархические, сетевые и реляционные. Цель системы управления базы данных состоит в том, чтобы делать данные быстро доступными множеству пользователей при поддержании их целостности, защищать данные против стирания и искажения и облегчать добавление, удаление и обновление данных по мере необходимости. СУБД должна обеспечить следующие функциональные возможности:

(a) Позволять хранение и поиск данных и делать выборки данных по одному или более атрибутах или отношениях.
(b) Стандартизировать доступ к данным, и разделять хранение и поиск при использовании данных в прикладных программах, чтобы поддержать независимость в тех программах.
(c) Обеспечить интерфейс между базой данных и прикладной программой, основанной на логическом описании данных.
(d) Делать функции доступа в прикладных программах, независимыми от физической структуры памяти, чтобы на программы не воздействовали изменения в носителях данных.
(e) Позволить нескольким пользователям обращаться к данным одновременно.
(f) Защитить базу данных от нелегальных изменений.
(g) Обеспечить четкие правила для манипуляции данными, соблюдаемые автоматически.

Большинство СУБД обеспечивает доступ к данным через языки программирования высокого уровня и через дружественные языки запросов, из которых SQL (Структурированный Язык Запросов) является наиболее общим для больших реляционных СУБД.

Хорошая СУБД гарантирует также, что пространственно непрерывные данные будут сохранены в физически смежных областях носителя данных, чтобы ускорить доступ и передачу данных. Быстрый доступ к большим количествам данных может быть критическим аспектом проектирования ГИС. Многие ГИС используют СУБД как часть их системы.

Выбор подходящей структуры базы данных

Должно быть очевидно, что все основные типы баз данных имеют достоинства, которые могут предложить для ГИС. Иерархические системы позволяют большие базы данных разделять в легко в управляемые фрагменты, но они негибки для формирования новых путей поиска файлов, и они могут содержать много избыточных данных. Сетевые системы содержат немного избыточных данных, если таковые вообще имеются и обеспечивают быстрые, направленные, но негибкие связи между объектами. Реляционные системы открыты, гибки и приспосабливаемы, но могут страдать от больших объемов данных, избыточности и долгого времени поиска. Следовательно, не удивительно, что эти методы часто используются вместе в пространственных информационных системах, чтобы дополнить друг друга.

Структуры данных для представления геометрии пространственных явлений

Эти главные структуры базы данных влияют на то, как географические данные дискретизируются и хранятся в ГИС. Много систем, как растровых, так и векторных, используют методы "оверлеев" или "слоев" для структурирования пространственных данных в интуитивно полезные группы.

Организация данных в растровых структурах данных

Растровая база данных строится из того, что пользователь осознает как ряд координатных слоев (Рис. 3.5а).

В простых растровых структурах, где каждая ячейка на каждом оверлее принимается как независимый модуль в базе данных, каждая ячейка идентифицирована координатной парой и набором атрибутов со значением для каждого оверлея.

Иерархическая структура, устанавливает отношение "многие к одному" между значениями атрибута и набором пунктов в модуле отображения, так что однородные области могут быть адресованы легко. Структурирование также позволяет выполнить сжатие файла, чтобы уменьшить потребности памяти.

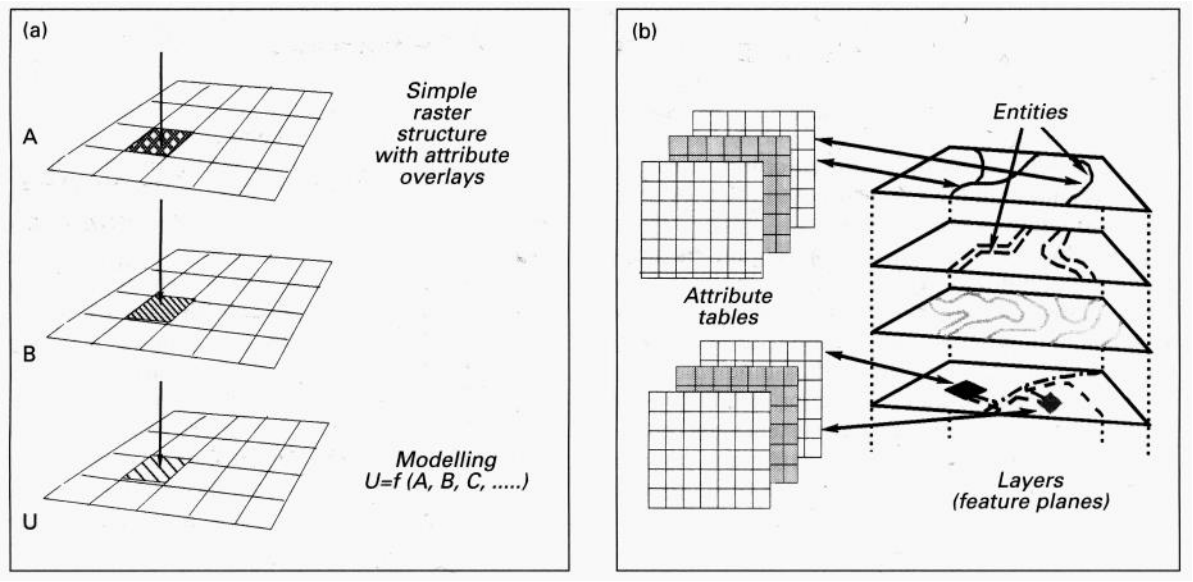


Рисунок 3.5. Планы данных и слои в (a) растре и (b) векторных структурах данных.

Возможно, также каждый слой сохранять как отдельный файл с общим заголовком, содержащим информацию о проекции карты, размере ячейки, числа строк и столбцов и типе данных. Это более эффективно, поскольку координатные значения не хранятся для каждой ячейки.

Компактные методы для хранения растровых данных

Когда растровые структуры данных используются, чтобы представить непрерывную поверхность, каждая ячейка имеет уникальное значение и требуется общее количество nrows*mcolumns, чтобы кодировать каждый оверлей, плюс общая информация относительно проекции карты, начала координат, размера сетки и типа данных. Наибольшие требования к памяти предъявляет тот случай, где тип данных - реальный и каждая ячейка содержит вещественное число.

Когда растровые структуры используются, чтобы представить строки или области, в которых пиксели всюду имеют то же самое значение, возможно сэкономить память. Есть четыре главных пути, которыми пространственные данные однородного полигона могут быть сохранены более экономно: цепочки кодов, коды текущей длины, блоки кодов и деревья квадрантов.

Цепочка кодов. См. Рис. 3.7. Граница области A может быть дана последовательностью векторных единиц в главных направлениях. Эти направления могут быть пронумерованы (Восток = 0, Север = 1, Запад = 2, Юг = 3). Например, если мы стартуем в строке =10, столбец =1, граница области кодируется по часовой стрелке: 0, 1, 0^2, 3, 0^2, 1, 0, 3, 0, 1, 0^3, 3^2, 2, 3^3,

0^2, 1, 0^5, 3^2, 2^2, 3, 2^3, 3, 2^3, 1, 2^2, 1, 2^2,1, 2^2, 1, 2^2, 1^3 где количество пикселей в каждом направлений показано степенью,

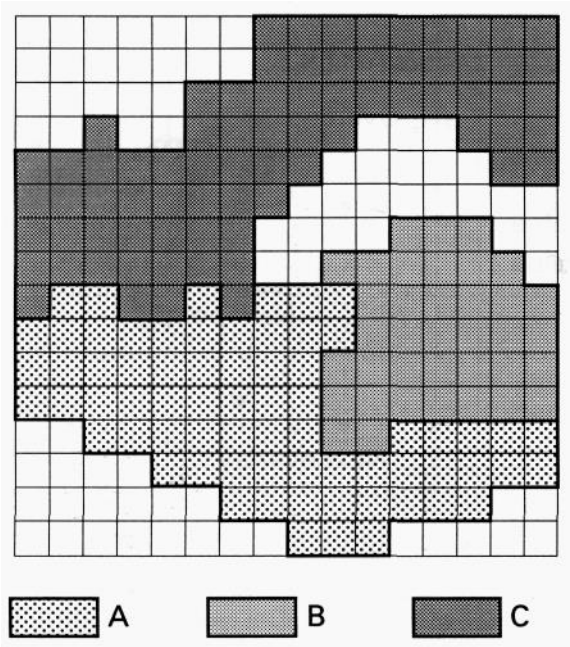


Рисунок 3.7. Простая растровая карта, где число шагов (пиксели) в каждом направлении дается номером верхнего индекса.

Цепочка кодов может быть сохранена, используя целочисленные типы данных, и поэтому обеспечивает очень компактный способ хранения области; она позволяют легко проводить некоторые операции типа оценки площади и периметра или обнаружения острых поворотов и вогнутостей.

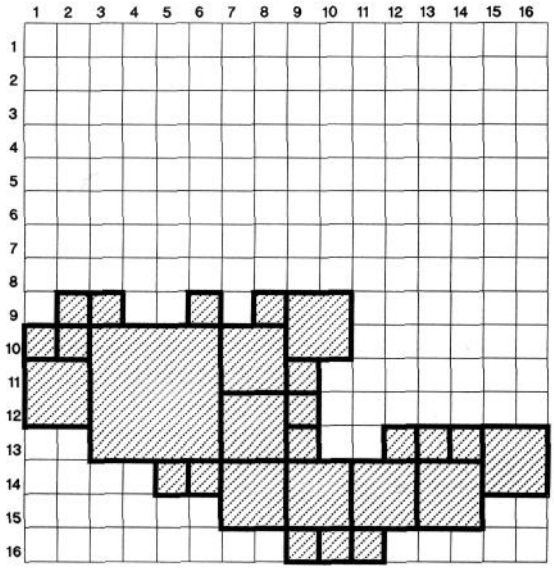
Коды текущей длины позволяют сохранять единицы растра строкой (слева направо) для каждого класса от первой до последней ячейки и атрибута. Целочисленный тип данных - вполне достаточен.

Для области А, показанной на рис. 3.7, коды будут выглядеть следующим образом:

Row 9 2,3 6,6 8,10
Row 10 1,10
Row 11 1,9
Row 12 1,9
Row 13 3,9 12,16
Row 14 5,16
Row 15 7,14
Row 16 9,11

Блоки кодов. Идея кодов текущей длины может быть расширена на два измерения, используя квадратные блоки. Рис. 3.8 показывает, как это может быть сделано для области растровой карты с рисунка 3.7.

Структура данных состоит из только трех чисел, положение (центр или левая нижняя часть) и размер каждого квадрата и их количества. Область А может быть сохранена 17 единичными квадратами + с 9 (4 квадрата) + 1 (16 квадратов). Учитывая, что две координаты необходимы для каждой площади, область может быть сохранена, используя 57 числа (54 для координат и 3 для размеров ячейки). И коды текущей длины и блоки кодов наиболее эффективны для больших простых форм и меньше для маленьких сложных областей.



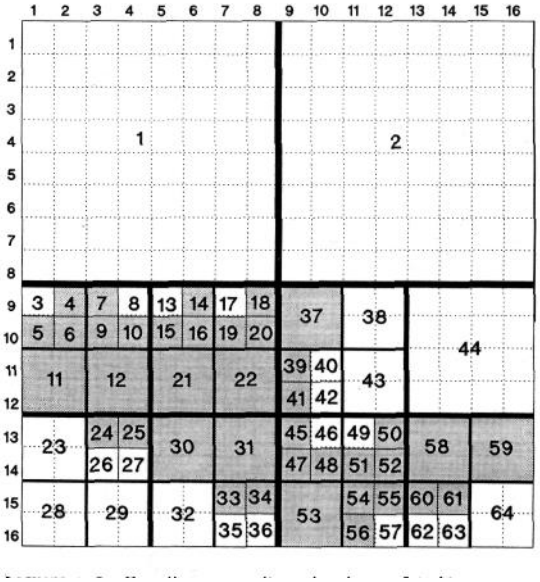


Рисунок 3.8. Преобразование полигона А в блоки.
Рисунок 3.9. Квадратомическое кодирование полигона А.

Квадратомические и бинарные деревья. Единственная проблема с регулярными сетками состоит в том, что разрешающая способность данных ограничена размером ячейки сетки. Квадратомические деревья обеспечивают успешный подход к адресованию с бесконечным набором уровней.

Наиболее эффективные методы компактного представления пространства основаны на последовательном, иерархическом делении массива. Если область составлена из квадрантов и указывается, какие квадранты полностью содержатся в области, то такое деление известно как дерево квадрантов. Самый низкий предел деления - отдельный пиксель.

Организация данных в векторных структурах данных

Векторная база данных построена, исходя из того, что пользователь чувствует как набор отдельных плоскостей с разными классами явлений. Единицы представлены как четкие мировые объекты, использующие координатное пространство, которое непрерывно, не квантуемо, как растровая структура. Все позиции, длины, и измерения определенны точно.

Фактически это не вполне возможно из-за ограничений длины компьютерного слова для точного представления координат и потому что все устройства отображения имеют основной размер шага или разрешающую способность. Помимо предположения о математически точных координатах, векторные методы хранения данных используют неявные отношения, которые позволяют сложным данным быть сохраненными в минимуме места.

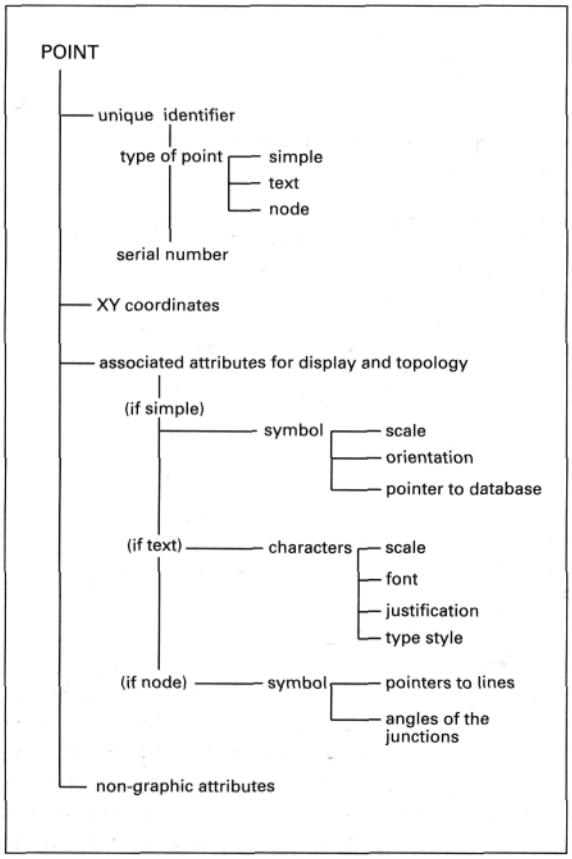


Рис. 3.13. Структура векторных данных для простой точки

Не существует никакого единственного, привилегированного метода. Этот раздел описывает диапазон векторных структур, используемых в ГИС для хранения точек, линий и полигонов.

Точечные объекты. Точечные объекты охватывают все географические и графические объекты, которые установлены единственной координатной парой. В дополнение к XY координатам другие данные должны существовать, чтобы указать то, какой 'точкой' они являются и любая другая информация, связанная с этим. Например, 'точка' может быть символом, не связанный с любой другой информацией. Но запись данных должна включить информацию о символе (размере, ориентации, цвете и др. Если бы 'точка' была текстовым объектом, запись данных должна включить информацию о текстовых символах, которые будут отображены, шрифте, выравнивании, масштабе и т.д. Рисунок 3.13 иллюстрирует возможную структуру данных для точечных объектов.

Линейные объекты. Линейные объекты могут быть определены как все линейные явления, созданные прямолинейными сегментами, составленных из двух или более координатных пар. Самая простая линия требует хранения начальной и конечной точки (две XY координатные пары) плюс возможная запись, указывающий символ изображения, который нужно использовать. Например, параметр символа изображения может использоваться, для прорисовки пунктирных линий на устройстве отображения даже при том что все отдельные сегменты линии не содержат записи о символе в базе данных.

Дуга - это набор XY координатных пар, описывающих непрерывную комплексную линию. Чем короче сегменты линии и чем больший число координатных пар, тем лучше цепочка аппроксимирует комплексную кривую. Пространство для хранения данных может быть сэкономлено, если указана функция интерполяции между координатными парами (например, сплайн), для отображения линии на мониторе. Как с точками и простыми линиями, дуги могут быть сохранены записями данных, указывающими тип символа линии дисплея, который нужно использовать.

Сети. Простые линии и цепочки не несут никакой пространственной информации о связности, которая могла бы потребоваться для дорог, транспортных путей или исследований сети дренажа. Чтобы получить линейную сеть, которую компьютер может отследить от линии к линии, необходимо добавить топологические указатели в структуру данных. Это достигается с помощью узлов. Рисунок 3.14 иллюстрирует вид структуры данных, которая была бы необходима для обеспечения связи между всеми ветвями сети. Помимо переноса указателей на дуги, узлы могут нести записи данных, указывающие угол, под которым каждая цепочка соединяет с узлом, таким образом, полностью определяя топологию сети. Эта простая структура соединения включает некоторую избыточность данных, потому что координаты в каждого узла зарегистрированы несколько (цепочка + 1) раз, где n - число цепочек, соединяющихся в узле. Атрибуты линий (обозначено черными точками на рисунке 3.14) могут использоваться для выбора предпочтительного маршрута.

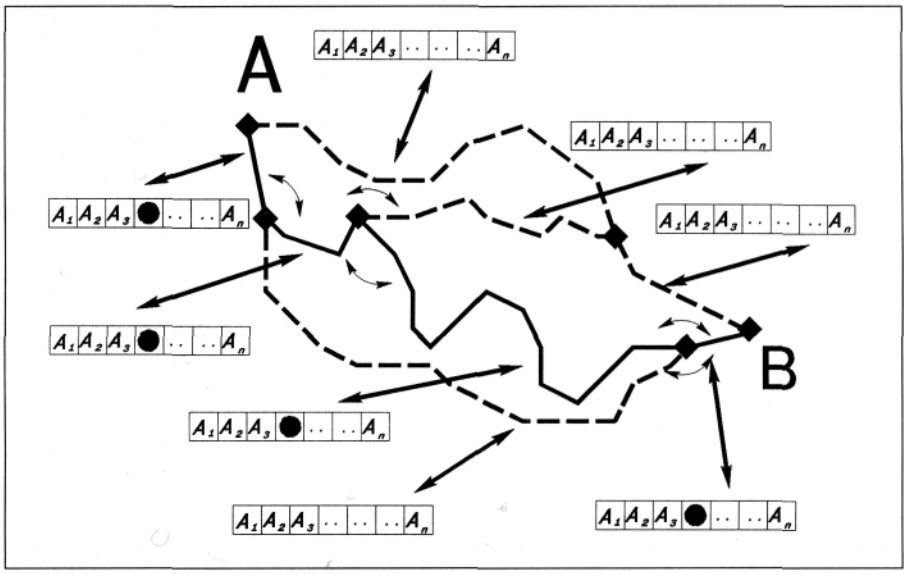


Рис. 3.14. Гибридная структура данных для сетевого анализа

Полигоны. Полигоны могут быть представлены различными способами в векторной базе данных. Так как многие виды пространственных данных связаны с полигонами, способы, которыми эти объекты представляются и управляются, заслуживают большого внимания.

Цель полигональной структуры данных состоит в том, чтобы описать топологические свойства областей (форму, соседей и иерархию) таким способом, что связанные атрибуты этих основных пространственных блоков могли бы отображаться и управляться как тематические данные карты. Перед описанием путей, которыми полигональная структура данных может быть создана, было бы полезно установить требования к полигональной сети, которые географические налагают географические данные.

Во-первых, все составляющие полигоны на карте будут иметь уникальную форму, периметр и площадь. Нет никакого стандартного модуля, как в растровых системах. Во-вторых, географические исследования требуют, чтобы структура данных была способна записывать соседей каждого полигона таким же образом, как требует связности потоковая сеть. В-третьих, полигоны на тематических картах очень разные - на озерах могут быть острова, на больших островах - озера и так далее.

Простые полигоны. Самый простой способ представить полигон как простую цепочку, то есть представлять каждый многоугольник набором XY координат границы (рис. 3.15). Названия или символы используются для того, чтобы сообщить пользователю, чем каждый многоугольник является, это набор простых текстовых единиц. Преимущество этого метода - простота, но он имеет много недостатков.

Это - (a) линии между смежными многоугольниками должны быть оцифрованы и сохранены дважды. При этом могут появиться серьезные ошибки в несоответствии (перекрытия и зазоры) по общей границе, (b) нет никакой информации о соседних полигонах, (c) острова невозможны, (d) нет

простых способов проверить, является ли топология границы правильна в случае висячих узлов («тупик») или паразитных полигонов - см. рисунок 3.16.

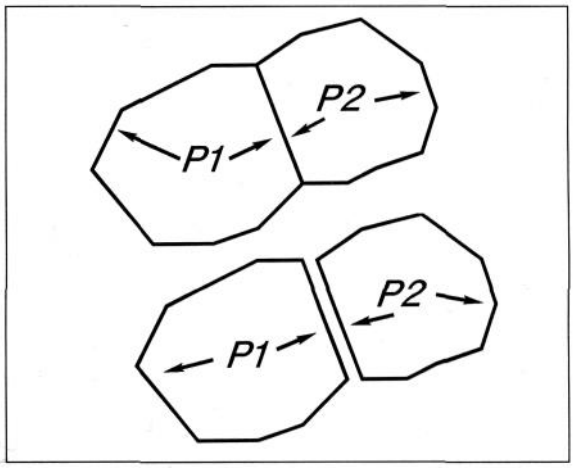


Рис. 3.15. Без топологии база данных не может различить полигоны, которые совместно используют границы (верхний рисунок), или действительно отдельные объекты (внизу)

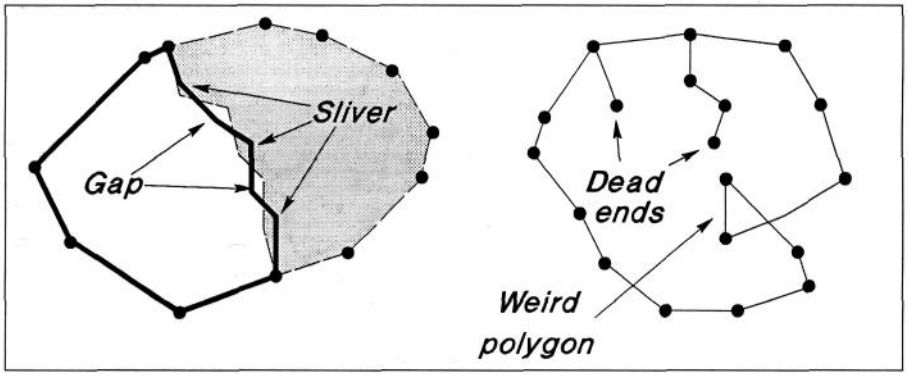


Рис. 3.16. Топологические ошибки в полигональной сети

Полигоны со списками точек. В этом представлении все координатные пары пронумерованы последовательно и упомянуты в списке, где записано - какие точки связаны с каждым полигоном (Рис. 3.17а). База данных со списками точек имеет преимущество в том, что границы между смежными многоугольниками уникальны, но проблема соседних полигонов все еще существует. Также, структура не позволяет легко удалить границы между смежными полигонами при переклассификации, когда оба полигона будут отнесены к одному классу. Проблема островных полигонов все еще существует, также как и проблемы проверки висячих дуг и паразитных полигонов.

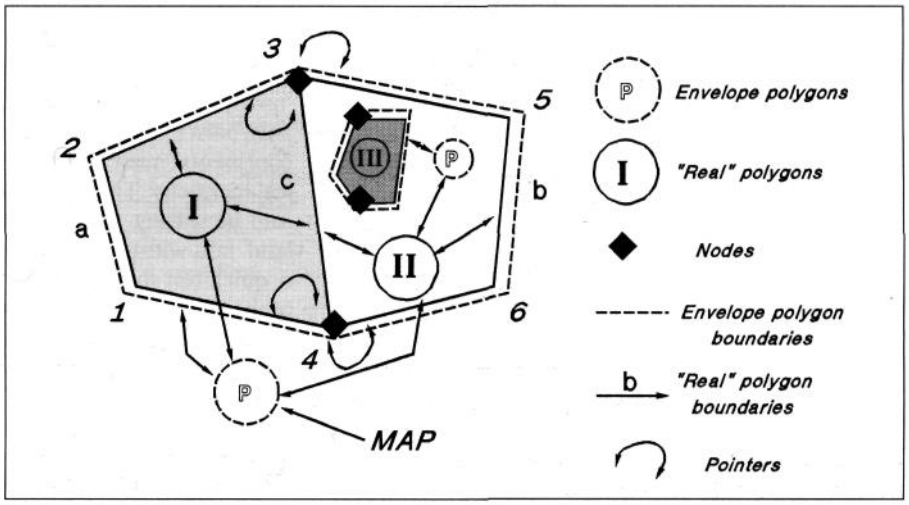


Рис. 3.18. Полная топологическая структура карты полигонов

Полная топологическая структура полигональной сети. Структура, показанная на рисунке 3.18, может быть создана из набора граничных цепочек или строк, которые были оцифрованы в любом порядке и в любом направлении.

Система допускает острова и озера, вложенные на любом уровне; это позволяет автоматически проверять паразитные полигоны и висячие дуги; а также автоматизировать ассоциацию непространственных атрибутов с результирующими полигонами. Исследования соседства полностью поддержаны. Оцифровывание границ полигона (оцифровка описана в Главе 4) и создание полигональной топологии выполняется отдельно. Процедуры, используемые для построения граничной топологии, должен соблюдать два предположения о входных данных, а именно что границы полигона были кодированы в форме дуг и что названия полигонов или другие записи, используемые для связи графики с атрибутными данными, цифруются в форме идентифицируемых точки где-нибудь в пределах каждого полигона.

Стадия 1. Соединение дуг в сеть границ. Дуги сначала сортируются согласно их протяжению (минимальные и максимальные X и Y координаты дуг) так, чтобы дуги топологически сходные между собой рядом находились в файле данных. Это экономит время при поиске смежных дуг. Затем дуги исследуются на предмет пересечений. Точки соединения (узлы) строятся в конце всех соединенных дуг. Записи данных дуги расширены, чтобы содержать указатели и углы. Дуги, пересекающиеся, автоматически разрезаются в новые дуги.

Стадия 2. Проверка полигонов на замкнутость. Результирующая сеть проверяется на замкнутость, просматривая модифицированные записи дуг, чтобы увидеть, имеют ли они указатели на другие дуги. Другие дуги могут быть дугами сами по себе в случае отдельных островов. Все дуги, не имеющие соединений с другими дугами, отмечаются для отображения специфическими способами или удаляются из подмножества дуг, которые нужно использовать для полигональной сети.

Стадия 3. Соединение линий в полигон. Первый шаг в соединение линий в полигоны - создание новый "внешнего" полигона, отделяющего от внешней границы карты (рисунок 3.18).

Внешний полигон не заметен пользователю; его собственная цель - формирование топологической структуры сети. Он создается рассмотрением дуг вокруг внешней границы по часовой стрелке и выборе самой левой дуги в каждом соединении. Уникальный идентификатор каждой дуги регистрируется и сохраняется наряду с другими данными и устанавливается флажок, указывающий на то, что каждая дуга была рассмотрена однажды.

После построения внешнего полигона могут быть созданы индивидуальные полигоны. Начинается процедура в том же месте, как и в предыдущем случае, но на сей раз по часовой стрелке включаются наиболее правая дуга в каждом соединении. Прибытие назад в отправную точку, означает, что идентифицированы все составляющие линии. В то же самое время выполняется проверка суммарного угла поворотов (рисунок 3.18) и если это - не 360 °, то была ошибка оцифровывания. Как в случае с внешним полигоном, каждый индивидуальный полигон получает несколько наборов информации:

(a) Уникальный идентификатор;
(b) Порядковый код полигона;
(c) Кольцевой указатель от внешнего полигона. В то же самое время идентификатор этого полигона записывается в кольцевом указателе внешнего полигона;
(d) Список всех граничных дуг. В то же самое время, уникальный идентификатор полигона записывается в данные линии;
(e) Кольцевой указатель на смежный полигон в сети;
(f) Минимальные и максимальные XY координаты граничного прямоугольника.

Поиск продолжается для следующего полигона на том же уровне в иерархии, и так далее пока все индивидуальные полигоны не будут созданы. Когда последний полигон в сети будет прослежен, его кольцевой указатель (e) будет указывать обратно на внешний полигон. Это гарантирует, что все граничные линии связаны с двумя многоугольниками.

Та же самая процедура выполняется для всех "островов" и "несвязанных субконтинентов". Как только все граничные дуги были связаны в полигоны, "острова" и "субконтиненты" должны быть размещены в надлежащей топологической иерархии.

Стадия 4. Вычисление площади полигонов. Следующая стадия включает вычисление площади полигонов, используя формулу трапеции (см. Блок 3.3). Полигоны в географических данных могут иметь много сотен координат граничных дуг и много островных полигонов, так что обычно более эффективно вычислить площади один раз, вычитая площади островов по мере необходимости, и затем хранить эти данные как связанный атрибут.

Стадия 5. Связывание неграфических атрибутов к полигонам. Последняя стадия формирования базы данных должна связать полигоны с их атрибутами, которые описывают то, что они представляют. Это может быть сделано несколькими способами. Первый - оцифровать уникальный текстовый объект в пределах каждого полигона, или ввести данные в интерактивном режиме после того, как полигоны были сформированы. Этот текст может использоваться как указатель на связанные атрибуты. Текст может использоваться для визуального отображения; он привязывается к полигону, используя точку внутри его (см. Рис 3.4).

Компьютер пишет уникальный идентификатор каждого полигона в его центре; в то же самое время компьютер печатает список всех идентификаторов многоугольника. Этот список можно объединить с файлом, содержащим другие неграфические атрибуты, на которые можно перекрестно ссылаться через уникальные идентификаторы полигонов.

Редактирование и обновление полигональной сети.

Векторные полигональные сети могут быть отредактированы: а) перемещением координат отдельных точек и узлов, б) изменением атрибутов полигонов, в) разрезанием (делением) линий или добавлением сегментов к линии, и даже целых полигонов. Изменение координат или связанных атрибутов не требует никакой модификации топологии. Изменение сети, удаление или прибавление линий и полигонов требует перевычисления топологии и перестройки базы данных. Следовательно, эти виды структур данных не очень эффективны для меняющихся географических объектов и явлений.

Структуры данных для пространственных данных: выбор между растром и вектором.

Растровые и векторные методы для пространственных структур данных совершенно различны при моделировании географической информации. Относительные достоинства обеих систем могут быть суммированы следующим образом:

Векторные структуры данных

Преимущества

- Хорошее представление моделей конечных объектов.
- Компактная структура данных.
- Топология может быть описана явно, что хорошо для сетевого анализа. Координатное преобразование и растягивание листов достигается просто.

- Точное графическое представление во всех масштабах. Возможен поиск, обновление и обобщение графики и атрибутов.

Недостатки

- Сложная структура данных.
- Объединение или пересечение нескольких слоев с полигональной сетью требует значительной производительности компьютера.
- Отображение и качественная печать занимает много времени и дорого.
- Пространственный анализ внутри основных модулей типа полигонов невозможен без дополнительных данных, потому что они внутренне гомогенны.
- Моделирование процессов пространственного взаимодействия в структурах, где нет явной топологией более трудно чем с растровыми структурами, потому что каждый пространственный объект имеет различную форму и форму.

Растровые структуры данных

Преимущества

- Простые структуры данных.
- Определенные местоположением манипуляции данными атрибутов просты. Много видов пространственного анализа и фильтрации могут использоваться. Математическое моделирование просто, потому что все пространственные объекты имеют простую, правильную форму.
- Технология дешева.
- Много типов данных доступно.

Недостатки

- Большие объемы данных.
- Использование крупных ячеек сетки, чтобы уменьшить объемы данных, приводит к потере информации и неспособности распознать феноменологически определенные структуры.
- Грубые растровые карты неэлегантны.
- Координатные преобразования трудны и требуют много времени и даже могут приводить к потере информации или искажению формы ячейки сетки.