1 ОРГАНИЗАЦИИ РЯДОМ

В этой главе мы разработаем сервис поиска «организации рядом со мной» (proximity service). Он используется для поиска ближайших ресторанов, отелей, театров, музеев и т. д. и является основным компонентом, благодаря которому, например, Yelp находит лучшие рестораны, расположенные неподалеку, а Google Карты показывают *k* ближайших заправок. На рис. 1.1 показан пользовательский интерфейс, с помощью которого можно найти на Yelp [1] рестораны по соседству. Обращаем внимание, что тайловые карты (map tiles), используемые в этой книге, созданы компанией Stamen Design [2], а данные взяты из OpenStreetMap [3].

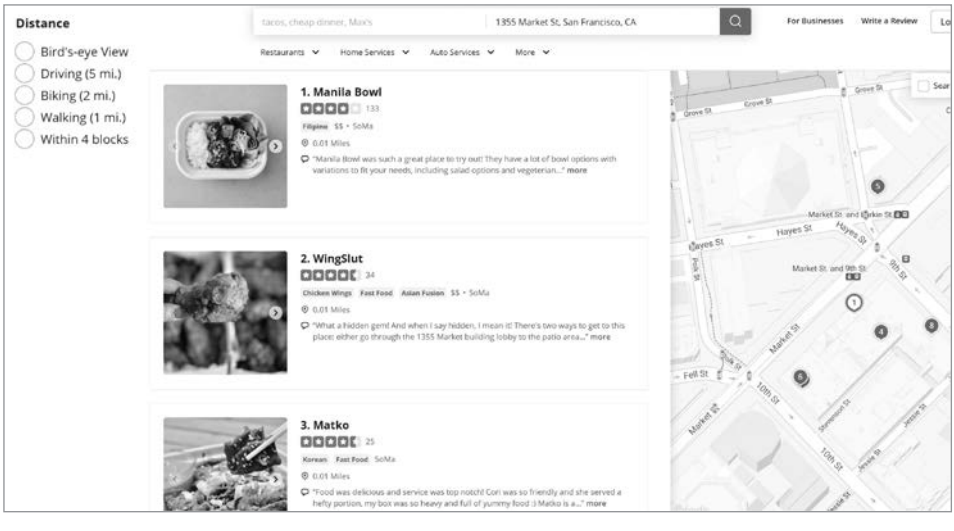

Рис. 1.1. Поиск ресторанов поблизости на Yelp

Шаг 1: понять задачу и определить масштаб решения

Yelp поддерживает множество функций, и разработать их все в ходе одного собеседования невозможно, поэтому соискателю важно конкретизировать поставленную перед ним задачу, задавая соответствующие вопросы. Это может выглядеть следующим образом.

Соискатель: Может ли пользователь указать радиус поиска? Если в этом радиусе недостаточно подходящих организаций, должна ли система расширить поиск?
Эксперт: Отличный вопрос. Предположим, что нас интересуют только организации в определенном радиусе. Если позволит время, мы можем обсудить, как расширить поиск, если найденных результатов окажется недостаточно.

Соискатель: Какой максимальный радиус разрешен? Могу ли я предположить, что это 20 км?
Эксперт: Это разумное предложение.

Соискатель: Может ли пользователь изменить радиус поиска?
Эксперт: Да, у нас есть следующие варианты: 0,5 км, 1 км, 2 км, 5 км и 20 км.

Соискатель: Как добавляется, удаляется или обновляется информация об организациях? Нужно ли отображать эти операции в режиме реального времени?
Эксперт: Владельцы могут добавлять, удалять или обновлять сведения о своих предприятиях. Предположим, что у нас есть соглашение с ними о том, что добавленные места и обновленные сведения о них начнут отображаться на следующий день.

Соискатель: Пользователь может перемещаться в пространстве во время работы с приложением/веб-сайтом, поэтому результаты поиска через некоторое время могут немного измениться. Нужно ли обновлять страницу, чтобы результаты были актуальными?
Эксперт: Предположим, что скорость перемещения пользователя невелика и нам не нужно постоянно обновлять страницу.

Функциональные требования

На основе этого разговора сосредоточимся на трех ключевых функциях разрабатываемой системы:

- возвращать все организации, основываясь на местоположении пользователя (широта и долгота) и заданном радиусе;
- владельцы могут добавлять, удалять или обновлять сведения о своей организации, но эта информация не обязательно должна отражаться в режиме реального времени;
- клиенты могут просматривать подробную информацию об организации.

Нефункциональные требования

Из бизнес-требований можно вывести список нефункциональных требований. Их также следует уточнить у эксперта.

- Низкая задержка: пользователи должны иметь возможность быстро найти расположенные рядом организации.

- Конфиденциальность данных: информация о местоположении — это конфиденциальные сведения. Когда мы разрабатываем сервис на основе местоположения (location-based service, LBS), то должны позаботиться о защите данных пользователя. Необходимо соблюдать соответствующие законы, такие как General Data Protection Regulation (GDPR) [4], California Consumer Privacy Act (CCPA) [5] и т. д.
- Высокая доступность и масштабируемость системы. Нужно убедиться, что мы сможем справиться с резким увеличением трафика в часы пик в густонаселенных районах.

Приблизительные оценки

Давайте проведем некоторые расчеты, чтобы определить потенциальный масштаб задачи и проблемы, которые необходимо будет решить. Предположим, что у нас 100 миллионов ежедневных активных пользователей и 200 миллионов организаций.

РАССЧИТЫВАЕМ QPS

- Секунд в сутках = $24 \times 60 \times 60 = 86\,400$. Для удобства вычислений округлим это число до 10^5. **Далее в книге число 10^5 всегда будет использоваться для количества секунд в сутках**.
- Предположим, что пользователь делает 5 поисковых запросов в день.
- Запросов в секунду (queries per second, QPS):

$$\frac{100 \text{ миллионов} \times 5}{10^5} = 5000\,.$$

Шаг 2: предложить высокоуровневый дизайн и получить одобрение

В этом разделе мы обсудим:

- дизайн API;
- высокоуровневый дизайн;
- алгоритмы для поиска организаций рядом;
- модель данных.

Дизайн API

Используем RESTful API для разработки упрощенной версии API.

```
GET /v1/search/nearby
```

Этот эндпоинт возвращает организации, соответствующие определенным критериям. В реальных приложениях результаты поиска обычно разбиваются на страницы. Пагинация [6] не является темой этой главы, но о ней стоит упомянуть во время собеседования.

Параметры запроса приведены в табл. 1.1.

Таблица 1.1. Параметры запроса

Поле	Описание	Тип данных
`latitude`	Широта заданного местоположения	decimal
`longitude`	Долгота заданного местоположения	decimal
`radius`	Радиус (необязательно). По умолчанию 5000 метров	int

```
{
  "total": 10,
  "businesses": [{business object}]
}
```

Объект бизнеса (организации) `business object` содержит все необходимое для отображения страницы результатов поиска, но для показа подробной информации об организации могут дополнительно понадобиться изображения, отзывы, рейтинги и т. д. Поэтому когда пользователь кликает для перехода на страницу с подробностями об организации, обычно требуется новый вызов эндпоинта для получения нужных сведений.

API для организации

API, связанные с объектом бизнеса, представлены в таблице ниже.

Таблица 1.2. API для предприятия

API	Что делает
`GET /v1/businesses/:id`	Возвращает подробную информацию об организации
`POST /v1/businesses`	Добавляет организацию
`PUT /v1/businesses/:id`	Обновляет информацию об организации
`DELETE /v1/businesses/:id`	Удаляет организацию

Если вас интересуют реальные API для поиска мест/организаций, прочитайте о Google Places API в [7].

Модель данных

В этом разделе обсудим соотношение операций чтения и записи (read/write ratio), а также составим схему данных. Особенно подробно рассмотрим ее масштабируемость.

Соотношение операций чтения и записи

Объем операций чтения высок из-за двух часто используемых функций:

- поиск организаций рядом;
- просмотр подробной информации о них.

С другой стороны, объем операций записи невелик, поскольку добавление, удаление и редактирование информации о бизнесе происходит нечасто.

Для системы с большим количеством операций чтения подходит реляционная база данных, например MySQL. Давайте рассмотрим ее схему более подробно.

Схема данных

Основными элементами нашей базы данных являются таблица организации и таблица геопространственных индексов.

Таблица организации

Эта таблица БД содержит подробную информацию о бизнесе (см. табл. 1.3 (адрес, город, штат, страна, широта, долгота)), а первичным ключом здесь является `business_id`.

Таблица 1.3. Таблица организации

business	
business_id	первичный ключ
address	
city	
state	
country	
latitude	
longtitude	

Таблица геоиндексов

Таблица геоиндексов используется для эффективной обработки данных с пространственной привязкой. Поскольку эта таблица требует некоторых знаний о геохешировании, мы обсудим ее в разделе «Масштабирование базы данных» на с. 43.

Высокоуровневый дизайн

Высокоуровневый дизайн показан на рис. 1.2. Система состоит из двух частей: геолокационного сервиса (location-based service, LBS) и сервиса организации (business-related service). Давайте рассмотрим каждый компонент системы.

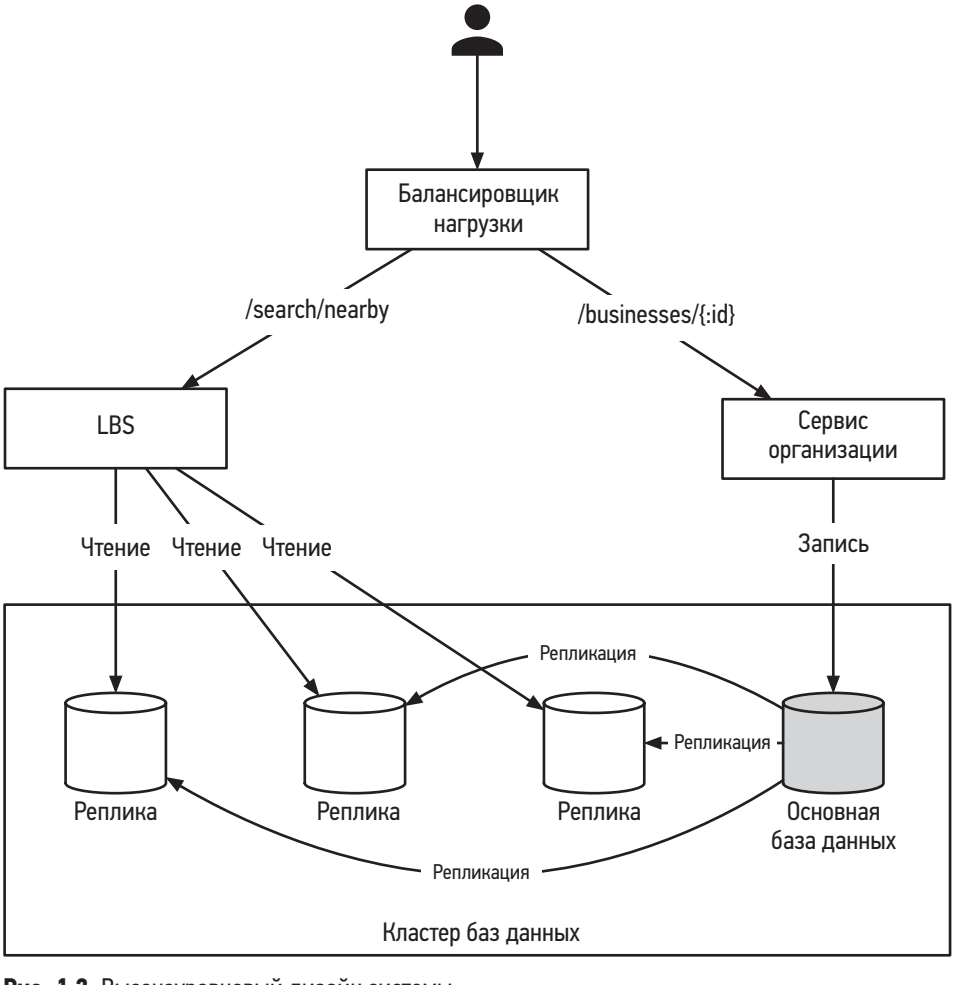

Рис. 1.2. Высокоуровневый дизайн системы

Балансировщик нагрузки

Балансировщик нагрузки автоматически распределяет входящий трафик между несколькими сервисами. Обычно компания предоставляет единую точку входа DNS и внутри направляет вызовы API на соответствующие сервисы на основе URL-путей.

Геолокационный сервис (LBS)

Сервис LBS является основной частью системы, которая находит организации в заданном радиусе для определенного местоположения пользователя. LBS обладает следующими характеристиками:

- Это сервис с большим количеством операций чтения и без запросов на запись.
- QPS высокий, особенно в часы пик в густонаселенных районах.
- Это сервис без состояния, поэтому его легко масштабировать по горизонтали.

Сервис организации

Сервис организации в основном работает с двумя типами запросов:

- Владельцы бизнеса заносят в систему информацию о своих предприятиях, удаляют или обновляют ее. Эти запросы в основном представляют собой операции записи, и QPS невысокий.
- Клиенты просматривают подробную информацию об организации. QPS высокий в часы пик.

Кластер баз данных

Кластер баз данных может использовать архитектуру primary-secondary с асинхронной репликацией. В этом случае основная (primary) база данных выполняет все операции записи, а для операций чтения используется несколько реплик. Данные сначала сохраняются в основной базе, а затем копируются в реплики. Из-за задержки репликации может возникнуть некоторое несоответствие между данными, считанными LBS, и данными, записанными в основную базу. Обычно это не является проблемой, поскольку информация о предприятии не нуждается в обновлении в режиме реального времени.

Масштабируемость сервиса организации и LBS

И сервис организации, и LBS — это сервисы без состояния, поэтому можно легко автоматически добавлять дополнительные серверы для подготовки к пиковым нагрузкам (например, в часы обеденного перерыва) и отключать серверы, когда трафик падает (например, в ночное время). Если система работает в облаке, мы можем сделать индивидуальные настройки для различных регионов и зон доступности, что повышает ее надежность [8]. Подробнее об этом мы поговорим позднее.

Алгоритмы для поиска организаций рядом

Компании часто используют существующие базы геопространственных данных, такие как Geohash в Redis [9] или Postgres с расширением PostGIS [10]. На собеседовании от вас не ожидают, что вы будете разбираться в их внутреннем устройстве. Лучше продемонстрировать свои навыки решения проблем и технические знания, объяснив, как работает геопространственный индекс, а не просто перечислять названия баз данных.

Следующим шагом будет изучение различных вариантов поиска близлежащих организаций. Мы перечислим несколько вариантов, подробно рассмотрим их и обсудим компромиссы.

Вариант 1: двумерный поиск

Самый интуитивно понятный, но наивный способ найти ближайшие организации — это нарисовать круг с заданным радиусом и найти все предприятия внутри него, как показано на рис. 1.3.

Рис. 1.3. Двумерный поиск

Этот процесс можно перевести в следующий SQL-запрос:

```
SELECT business_id, latitude, longitude,
FROM business
WHERE (latitude BETWEEN {:my_lat} – radius AND {:my_lat} + radius)
AND
    (longitude BETWEEN {:my_long} – radius AND {:my_long} + radius )
```

Но такой запрос неэффективен, поскольку нам нужно просканировать всю таблицу. А что, если мы создадим индексы для столбцов longitude и latitude? Повысит ли это эффективность? Ответ: не намного. Проблема в том, что у нас двумерные данные, и датасет, возвращаемый из каждого измерения, все равно может быть огромным. Например, как показано на рис. 1.4, мы можем быстро получить датасеты 1 и 2 благодаря индексам столбцов longitude и latitude.

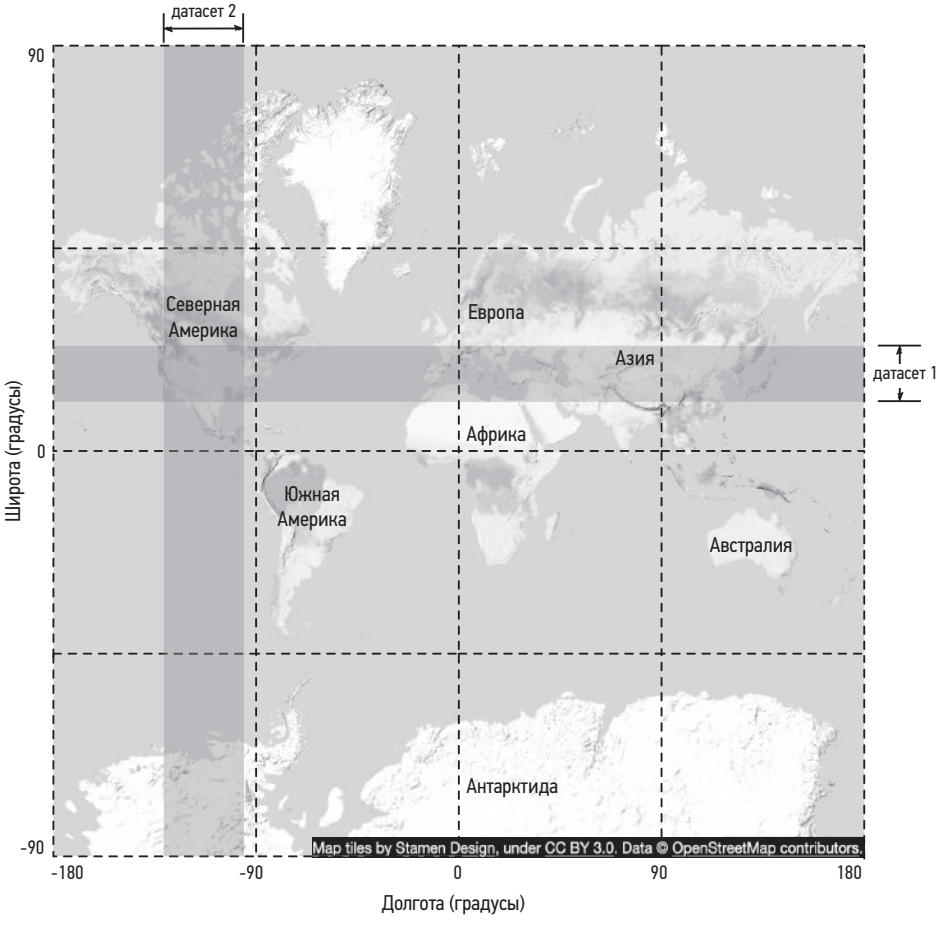

Рис. 1.4. Пересечение двух датасетов

Но чтобы найти нужные места в определенном радиусе, необходимо определить пересечения этих датасетов. Это неэффективно, поскольку каждый датасет содержит огромное количество данных.

Проблема с предыдущим подходом заключается в том, что индекс базы данных может ускорить поиск только в одном измерении. Естественно, возникает следующий вопрос: можно ли перевести двумерные данные в одно измерение? Ответ — да.

Прежде чем мы разберем возможные ответы, давайте рассмотрим различные типы методов геопространственного индексирования. В широком смысле существует два таких типа, как показано на рис. 1.5. На схеме выделены алгоритмы, которые мы рассмотрим подробнее, поскольку они широко используются в отрасли.

- Хеширование: равномерная сетка (even grid), геохеширование (geohash), cartesian tiers [11] и т. д.
- Деревья: дерево квадрантов (quadtree), Google S2, RTree [12] и др.

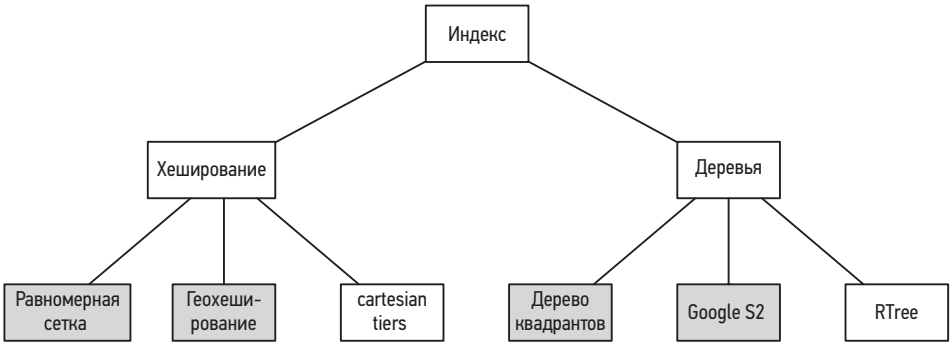

Рис. 1.5. Различные типы геопространственных индексов

Несмотря на то что базовые реализации этих подходов отличаются друг от друга, общая идея одна и та же — **разделить карту на более мелкие области и создать индексы для быстрого поиска**. Среди них наибольшее распространение в реальных приложениях получили геохеширование, дерево квадрантов и Google S2. Давайте рассмотрим их по очереди.

НАПОМИНАНИЕ

На реальном собеседовании обычно не требуется объяснять детали реализации вариантов геопространственного индексирования. Однако важно иметь некоторое базовое представление о том, как оно работает, а также о его ограничениях.

Вариант 2: равномерная сетка

Один из простейших подходов — равномерно разделить пространство сеткой на одинаковые небольшие ячейки (рис. 1.6). Таким образом, в одной ячейке может быть несколько организаций, и каждая из организаций на карте относится лишь к одной ячейке.

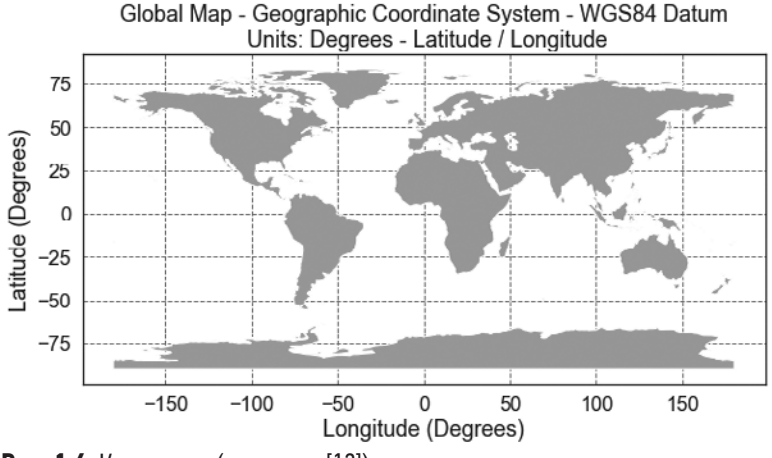

Рис. 1.6. Карта мира (источник: [13])

Этот подход в некоторой степени работает, но у него есть одна серьезная проблема: организации распределяются неравномерно. В центре Нью-Йорка их может быть много, в то время как на других территориях, таких как пустыни или океаны, нет вообще. Равномерно разделив пространство на одинаковые ячейки, мы получаем очень неравномерное распределение данных. В идеале нужно использовать более мелкие ячейки для густонаселенных районов и крупные для малонаселенных. Еще одна потенциальная проблема — поиск соседних ячеек фиксированной сетки.

Вариант 3: геохеширование

Геохеширование лучше, чем вариант с равномерной сеткой. Оно работает за счет сокращения двумерных данных о долготе и широте в одномерную строку букв и цифр. Алгоритмы геохеширования на каждом шаге рекурсивно делят карту мира на все более мелкие сетки. Давайте рассмотрим, как это работает на высоком уровне.

Сначала разделим карту земного шара на четыре квадранта по нулевому меридиану и экватору.

- диапазон широт [–90; 0] обозначен как 0;
- диапазон широт [0; 90] обозначен как 1;

- диапазон долгот [–180; 0] обозначен как 0;
- диапазон долгот [0; 180] обозначен как 1.

Рис. 1.7. Геохеширование

Потом разделим каждую ячейку на четыре меньшие. В итоге каждая из них может быть представлена сочетанием кодов долготы и широты (рис. 1.8).

Продолжаем это разделение до тех пор, пока размер сетки не достигнет требуемой точности. В геохешировании обычно используется кодировка base32 [14]. Давайте рассмотрим два примера.

- Геохеш головного офиса Google (длина = 6):

  `1001 10110 01001 10000 11011 11010` (base32 в двоичном формате) → `9q9hvu` **(base32)**

- Геохеш головного офиса Facebook (длина = 6):

  `1001 10110 01001 10001 10000 10111` (base32 в двоичном формате) → `9q9jhr` **(base32)**

Геохеш имеет 12 степеней точности (также называемых уровнями), как показано в табл. 1.4. Степень точности определяет размер ячейки сетки. Нас интересуют

только геохеши с длиной от 4 до 6. Это связано с тем, что если длина больше 6, то размер ячейки слишком мал, а если меньше 4, то слишком велик (см. табл. 1.4).

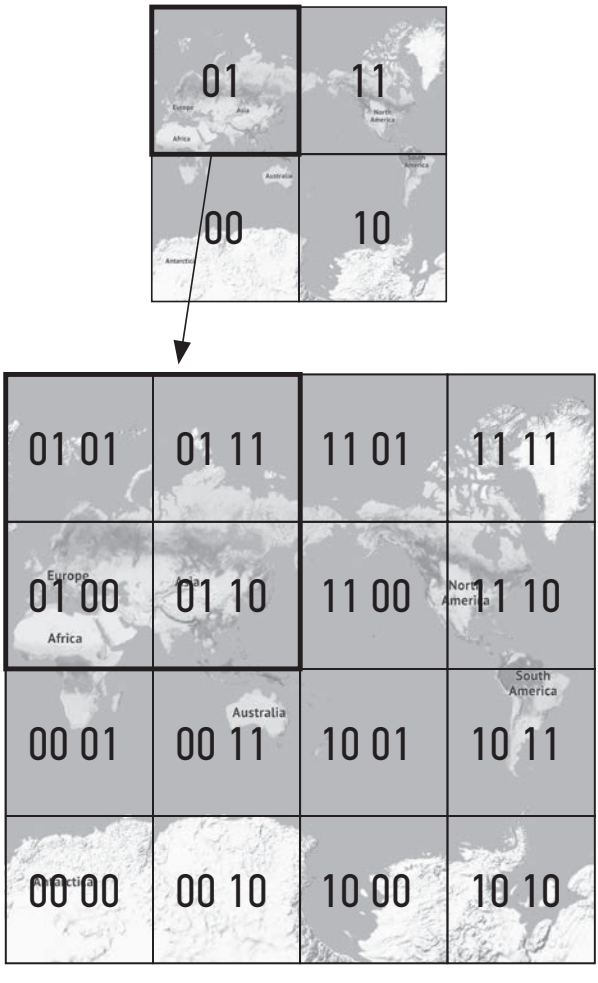

Рис. 1.8. Дополнительно разделенные ячейки

Как выбрать правильную точность? Мы хотим найти минимальную длину геохеша, который охватывает всю окружность радиуса, заданного пользователем. Соответствующая зависимость между радиусом и длиной геохеша показана в табл. 1.5.

В большинстве случаев этот подход работает отлично, но есть несколько пограничных случаев, связанных с обработкой границ геохешей, и стоит обсудить это с экспертом, проводящим интервью.