가상 면접 사례로 배우는

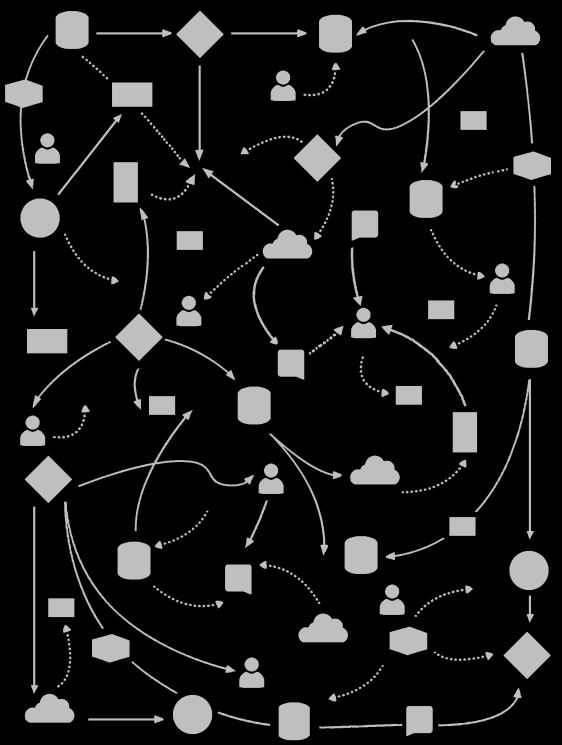
대규모 시스템 설계 기초 2

System Design Interview
Volume 2

가상 면접 사례로 배우는 대규모 시스템 설계 기초 2

전자책 1쇄 발행 2024년 1월 18일 (종이책 초판 2쇄 반영) **지은이** 알렉스 쉬, 산 람 **옮긴이** 이병준 **펴낸이** 한기성 **펴낸곳** (주)도서출판인사이트 **편집** 백혜영 **등록번호** 제2002-000049호 **등록일자** 2002년 2월 19일 **주소** 서울특별시 마포구 연남로5길 19-5 **전화** 02-322-5143 **팩스** 02-3143-5579 **블로그** https://blog.insightbook.co.kr **ISBN** 978-89-6626-433-9

프로그래밍 인사이트



가상 면접 사례로 배우는

대규모 시스템 설계 기초 2

알렉스 쉬·산 람 지음 | 이병준 옮김

인사이트

차례

옮긴이의 글

2권은 1권에서 다루지 못했던 많은 다양한 시스템들을 다룬다. 1장부터 3장까지는 지리적 위치 기반의 관계성 데이터를 저장하고 서비스할 때 발생하는 문제에 초점을 맞춘다. 4장과 5장은 메시지 큐에 기반한 느슨하게 결합된 아키텍처가 실무에서 어떻게 활용될 수 있는지를 깊이 있게 설명한다. 6장은 다량의 이벤트를 제때 정확히 처리하기 위해서는 무엇을 고려해야 하는지 살핀다. 7장은 예약 시스템 전반에 내재된 재미있는 문제들을 푸는 데 집중하며, 8장은 이메일처럼 역사가 오래된 기술을 대규모 서비스로 확장하는 것이 얼마나 도전적인 문제인지를 알려 준다. 9장은 단순하게 시작한 클라우드 서비스가 어떻게 다양한 쓰임을 지원하는 복잡한 서비스로 발전할 수 있었는지 보여 주고, 10장은 온라인 게임 서비스의 핵심 기술이 어떻게 구현되는지 설명한다. 그리고 11장부터 13장까지는 인터넷 생태계의 가장 핵심적 부분이라고 할 수 있는 자금 흐름의 무결성을 어떻게 보장할 수 있는지에 초점을 맞춘다.

이렇게 다양한 시스템을 살펴보며 얻은 소견을 공유하자면, 시스템 설계에 있어 정답은 없는 것 같다. 특히 13장에서 다루는 증권 거래소 시스템의 내부 구조는 응답 지연 시간을 낮춘다는 한 가지 목표를 깊이 추구하면 어떤 형태의 시스템이 만들어질 수 있는지를 보여 준다. 아마 거래소 이외의 업계에는 적용하기 까다로울 것이다. 하지만 여기서 우리는 사업적 목표에 부응하는 설계가 좋은 설계라는 교훈을 얻는다. 어떤 설계안도 맹신할 필요가 없다는 뜻이기도 하다. 이 단순한 교훈은 이 책 곳곳에서 수시로 반복된다. 특히 7장에서 설명하는 예약 시스템의 경우, 고객의 쓰임새만 제대로 지원할 수 있다면 배후의 시스템이 그렇게 복잡할 필요는 없다는 사실을 깨우쳐 준다. 고객에게 동일한 가치를 전달할 수 있다면, 가장 단순한 시스템이 최고로 좋은 시스템인 것이다.

인생의 많은 문제가 그렇듯이 좋은 설계는 하루 아침에 만들어지지 않으며 운영 관리 편의성, 비용, 개선 및 개발 용이성 등의 다양한 주제를 오랜 시간 동안 고단하게 탐구한 끝에 탄생한다. 이 책이 그런 수고로움을 조금이나마 줄여 줄 수 있기를 기대해 본다.

시애틀에서
이병준 드림

서문

시스템 설계 면접 기법을 배우기로 결정한 여러분, 환영한다. 설계 면접에 나오는 문제들은 기술 면접 문제 가운데서도 가장 까다롭다. 지원자는 어떤 소프트웨어 시스템의 아키텍처를 설계해야 한다. 이 시스템은 뉴스 피드일 수도 있고, 구글 검색 시스템일 수도 있고, 채팅 시스템일 수도 있다. 무섭게 느껴질 수 있을 법한 질문들이고, 공략 방법에도 정해진 패턴은 없다. 질문 범위도 보통 아주 넓고 모호하다. 정해진 결론(open-ended)이나 완벽한 답안이 없는 경우가 많다.

기업들은 시스템 설계 면접을 광범위하게 시행하고 있는데 이런 면접에서 드러나는 의사소통 및 문제 해결 기술이 소프트웨어 엔지니어가 업무에 일상적으로 사용하는 능력과 비슷하기 때문이다. 지원자를 평가할 때는 모호한 문제를 어떻게 분석하고 단계적으로 해결하는지를 살펴보게 된다.

시스템 설계 면접 문제에는 정해진 결론이 없다고 말했다. 우리의 현실이 그렇듯 하나의 설계에는 여러 가지 변종이 있을 수 있다. 우리가 원하는 것은 서로 합의한 설계 목표에 부합하는 아키텍처다. 면접관에 따라 토론도 다른 방향으로 흘러갈 수 있다. 시스템의 거의 모든 측면을 다루는 개략적(high-level) 아키텍처를 요구하는 면접관도 있고, 특정 영역에만 집중하라고 요구하는 면접관도 있다. 어느 쪽이건, 매끄러운 토론이 이루어지기 위해서는 시스템 요구사항, 제약사항, 그리고 성능 병목 지점을 잘 이해해야 한다.

이 책의 목적은 시스템 설계 면접 문제를 푸는 데 안정적으로 적용할 수 있는 전략과 지식 토대를 제시하는 것이다. 면접을 성공적으로 마치려면 올바른 전략과 지식을 갖추는 것이 무엇보다 중요하다.

또한 이 책은 시스템 설계 면접 문제들을 공략하는 단계적 프레임워크도 제공한다. 그 프레임워크를 적용하려면 어떻게 해야 하는지, 실제로 따라하면서 배울 수 있는 많은 예제를 상세한 설명과 함께 제공한다. 이 예제들과 함께 꾸준히 연습하다 보면, 시스템 설계 면접을 성공적으로 치를 준비가 끝나 있을 것이다.

이 책은 《가상 면접 사례로 배우는 대규모 시스템 설계 기초》[1]의 후속편이다. 1편도 읽으면 좋긴 하겠지만, 설사 읽지 않았어도 이 책을 읽는 데는 아무 무리가 없다. 분산 시스템에 대한 기본적 이해를 갖춘 독자라면 누구나 편안히 읽을 수 있을 것이다. 그럼 시작해 보자!

추가 자료

각 장(chapter) 말미에는 참고 자료에 대한 링크가 수록되어 있다. 아래 깃허브(GitHub) 저장소에 가면 그 모든 링크를 실제로 클릭하여 방문해 볼 수 있을 것이다.

https://bit.ly/systemDesignLinks

저자 알렉스와 직접 소통하고 싶은 독자는 아래 링크도 방문해 보기 바란다. 매일 새로운 시스템 인터뷰 팁이 올라온다.

twitter.com/alexxubyte

bit.ly/linkedinaxu

1 (옮긴이) 원제는 *System Design Interview – An insider's guide*. 알렉스 쉬 지음, 이병준 옮김, 인사이트, 2021년.

감사의 글

이 책의 모든 설계가 필자의 창작물이면 좋았겠으나, 사실 대부분의 아이디어는 엔지니어링 블로그, 연구 논문, 코드, 기술 발표 등 다른 많은 곳에서도 찾을 수 있는 것들이다. 우리가 한 일은 이 아이디어들을 모아 검토한 다음 개인적 경험을 더하여 알기 쉽게 정리한 것이다. 아울러, 이 책을 내기까지 많은 엔지니어와 매니저가 다양한 도움을 주었음을 밝히고 싶다. 결정적인 의견과 꼼꼼한 리뷰를 통해 도왔을 뿐 아니라, 그들 가운데 일부는 몇몇 장의 많은 부분을 직접 쓰기도 하였다. 이 자리를 빌려 깊은 감사를 드린다.

- 근접성 서비스(Proximity Service; 멍 돤(Meng Duan); 텐센트(Tencent))
- 주변 친구(Nearby Friends; 옌 궈(Yan Guo); 아마존(Amazon))
- 구글 맵(Google Maps; 알리 아미니안(Ali Aminian); 어도비(Adobe), 구글(Google))
- 분산 메시지 큐(Distributed Message Queue; 리오넬 리우(Lionel Liu); 이베이(eBay))
- 분산 메시지 큐(탄메이 데시판드(Tanmay Deshpande); 슐럼버거(Schlumberger))
- 광고 클릭 이벤트 집계(Ad Click Event Aggregation; 친다 비안(Xinda BIan); 앤트 그룹(Ant Group))
- 실시간 게임 리더보드(Real-time Gaming Leaderboard; 조쉬 헤인스(Jossie Haines); 틸(Tile))
- 분산 이메일 시스템(케빈 헨릭슨(Kevin Henrikson), 제이제이 좡(JJ Jhuang); 인스타카트(Instacart))

- S3와 유사한 객체 저장소(S3-like Object Store; 즈텅 황(Zhiteng Huang); 이베이(eBay))

이 책의 초고에 많은 의견을 준 다음 분들에게도 특별히 깊은 감사의 뜻을 표하고 싶다.

- 다싯 데이브(Darshit Dave; 블룸버그(Bloomberg))
- 드와라크나스 박시(Dwaraknath Bakshi; 트위터(Twitter))
- 페이 난(Fei Nan; 구스토(Gusto), 에어비앤비(Airbnb))
- 리처드 수(Richard Hsu; 아마존(Amazon))
- 사이먼 가오(Simon Gao; 구글(Google))
- 스탠리 매튜 토마스(Standly Mattew Thomas; 마이크로소프트(Microsoft))
- 원한 왕(Wenhan Wang; 틱톡(Tiktok))
- 시와칸트 바티(Shiwakant Bharti; 아마존(Amazon))

편집을 맡아 많은 귀중한 의견을 제시한 도미닉 가버(Dominic Gover)와 덕 워렌(Doug Warren)에게도 감사드린다.

그리고 마지막으로, 이 책에 너무나 중요한 공헌을 한 엘비스 렌(Elvis Ren)과 화 리(Hua Li)에게 아주 특별한 감사의 마음을 전하고 싶다. 이들이 없었다면 이 책은 지금 모습으로 세상에 나올 수 없었을 것이다.

1장

System Design Interview Volume 2

근접성 서비스

이번 장에서는 근접성 서비스(proximity service)를 설계한다. 근접성 서비스는 음식점, 호텔, 극장, 박물관 등 현재 위치에서 가까운 시설을 찾는 데 이용되며, 옐프(Yelp) 앱의 경우에는 주변에 있는 좋은 식당 검색, 구글 맵의 경우에는 가까운 k개 주유소 검색 등의 기능 구현에 이용된다. 그림 1.1은 옐프의 '주변 식당 검색' 기능의 사용자 인터페이스다.[1]

이 책에 실린 지도 타일(tile)은 스테이먼 디자인(Stamen Design)[2] 사가 만든 것이고, 지도 데이터로는 OpenStreetMap[3]이 제공한 것을 이용하였다.

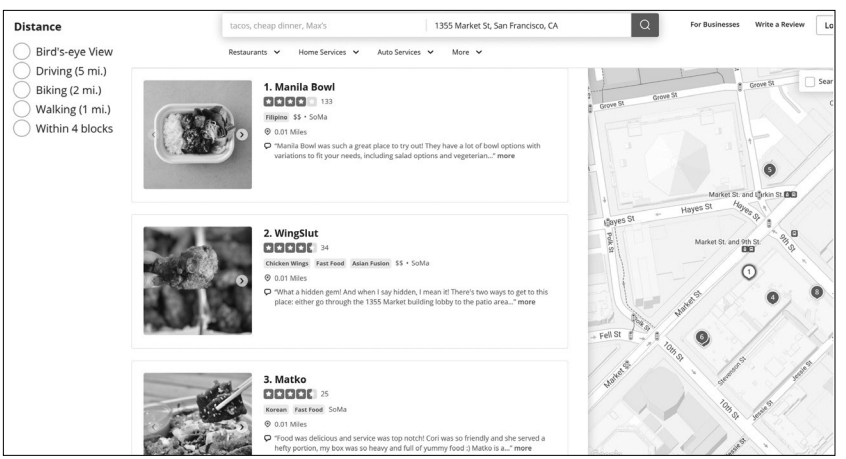


그림 1.1 옐프의 주변 검색 기능

1단계: 문제 이해 및 설계 범위 확정

옐프가 제공하는 모든 기능을 인터뷰 시간 내에 설계할 수는 없다. 그러니 질문을 던져 설계 범위를 좁혀야 한다. 다음은 면접관과 지원자의 대화 사례다.

지원자: 사용자가 검색 반경(radius)을 지정할 수 있어야 하나요? 검색 반경 내에 표시할 사업장이 충분치 않은 경우에는 검색 반경을 시스템이 알아서 넓혀도 괜찮을까요?

면접관: 좋은 질문입니다. 일단은 주어진 반경 내의 사업장만 대상으로 한다고 하죠. 시간이 남으면 주어진 범위 안에 사업장이 많지 않은 경우를 어떻게 처리할지 이야기해 볼 수 있겠네요.

지원자: 최대 허용 반경은 얼마입니까? 20km(12.5mile)로 가정해도 괜찮을까요?

면접관: 네, 괜찮은 가정인 것 같습니다.

지원자: 사용자가 UI에서 검색 반경을 변경할 수 있어야 하나요?

면접관: 네. 다음과 같은 선택지가 주어져야 합니다. 0.5km(0.31mile), 1km(0.62mile), 2km(1.24mile), 5km(3.1mile) 그리고 20km(12.42mile)입니다.

지원자: 사업장 정보는 어떻게 시스템에 추가되고, 삭제되고, 갱신됩니까? 사업장 정보에 대한 작업 결과가 사용자에게 실시간으로 보여져야 할까요?

면접관: 사업장 소유주가 사업장 정보를 시스템에 추가·삭제·갱신할 수 있어야 합니다. 새로 추가하거나 갱신한 정보는 다음날까지 반영되어야 한다고 계약서에 명시되어 있다고 가정하죠.

지원자: 사용자가 이동 중에 앱이나 웹사이트를 이용한다면 검색 결과는 시간이 흐름에 따라 달라져야 할 텐데요. 검색 결과를 항상 현재 위치 기준으로 유지하기 위해 화면을 자동 갱신해야 할까요?

면접관: 사용자의 이동 속도가 그리 빠르지 않아서 상시적으로 페이지를 갱신할 필요는 없다고 칩시다.

기능 요구사항

이 대화에 근거하여, 다음 세 가지 핵심 기능에 집중할 것이다.

- 사용자의 위치(경도와 위도 쌍)와 검색 반경 정보에 매치되는 사업장 목록을 반환
- 사업장 소유주가 사업장 정보를 추가·삭제·갱신할 수 있도록 하되, 그 정보가 검색 결과에 실시간으로 반영될 필요는 없다고 가정
- 고객은 사업장의 상세 정보를 살필 수 있어야 함

비기능 요구사항

방금 살펴본 사업 요구사항으로부터 다음과 같은 비기능 요구사항(non-functional requirements)을 도출할 수 있다. 이 각각을 면접관과 확인해야 한다.

- 낮은 응답 지연(latency): 사용자는 주변 사업장을 신속히 검색할 수 있어야 한다.
- 데이터 보호(data privacy): 사용자 위치는 민감한 정보다. 위치 기반 서비스(Location-Based Service, LBS)를 설계할 때는 언제나 사용자의 정보를 보호할 방법을 고려해야 한다. GDPR(General Data Protection Regulation)[4]이나 CCPA(California Consumer Privacy Act)[5] 같은 데이터 사생활 보호 법안을 준수하도록 해야 한다.
- 고가용성(high availability) 및 규모 확장성(scalability) 요구사항: 인구 밀집 지역에서 이용자가 집중되는 시간에 트래픽이 급증해도 감당할 수 있도록 시스템을 설계해야 한다.

개략적 규모 추정

시스템의 규모가 대략 어느 정도이며 어떤 수준의 도전적 과제를 해결해야 하는지 결정하기 위해, 개략적인 추정(back-of-the-envelope calculation)을 해 보도록 하자. 일간 능동 사용자(Daily Active User, DAU)는 1억 명(100million)이며 등록된 사업장 수는 2억(200million)이라고 하자.

QPS (Query per Second) 계산

- 1일 = 24시간 × 60분 × 60초 = 86,400초. 계산을 쉽게 하기 위해 대략 100000, 즉 10^5로 올림하여 쓰도록 하겠다. 이 책 전반에서 하루는 10^5초라고 가정할 것이다.
- 한 사용자는 하루에 5회 검색을 시도한다고 가정한다.
- 따라서 QPS = (1억 × 5) / 10^5 = 5,000

2단계: 개략적 설계안 제시 및 동의 구하기

이번 절에서는 다음 내용을 논의할 것이다.

- API 설계
- 개략적 설계안
- 주변 사업장 검색 알고리즘
- 데이터 모델

API 설계

RESTful API 관례를 따르는 간단한 API를 만들어 보도록 하겠다.

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

이 API는 특정 검색 기준에 맞는 사업장 목록을 반환한다. 실제로 사용되는 애플리케이션의 경우, 검색 결과는 보통 페이지 단위로 나눠 반환한다. 이번 장에서는 페이지 분할(pagination)[6]에 초점을 맞추지는 않을 것이나, 면접장에서는 언급하면 좋을 수 있다.

API 호출 시에 전달할 인자(parameter)는 다음과 같다.

필드	설명	자료형
latitude	검색할 위도	decimal
longitude	검색할 경도	decimal
radius	선택적 인자(optional). 생략할 경우 기본값은 5000m(대략 3마일)이다.	int

표 1.1 호출 인자

반환되는 결과는 다음과 같은 형태를 띤다.

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

위 코드에서 'business object', 즉 각 사업장을 표현하는 객체는 검색 결과 페이지에 표시될 모든 정보를 포함한다. 하지만 사업장 상세 정보 페이지에서는 사업장의 사진, 리뷰, 별점 등의 추가 정보가 필요할 수 있다. 그러므로 사용자가 사업장 상세 정보 페이지를 클릭하면 또 다른 API를 호출하여 사업장의 상세 정보를 가져올 필요가 있다.

사업장 관련 API

표 1.2는 사업장 객체 관련 API 목록이다.

APi	설명
GET /v1/businesses/:id	특정 사업장의 상세 정보 반환
POST /v1/businesses	새로운 사업장 추가
PUT /v1/businesses/:id	사업장 상세 정보 갱신
DELETE /v1/businesses/:id	특정 사업장 정보 삭제

표 1.2 사업장 관련 API

장소나 업장 검색과 관련하여 실제로 사용되는 API가 궁금하다면 구글의 장소 API(Google Places API)[7]와 옐프의 사업장 API(Yelp business endpoints)[8]를 참고하기 바란다.

데이터 모델

이번 절에서는 읽기/쓰기 비율(read/write ratio) 및 스키마 설계(schema design)에 대해 알아본다. 데이터베이스의 규모 확장성에 대해서는 "상세 설계" 절에서 자세히 살펴보겠다.

읽기/쓰기 비율

읽기 연산은 굉장히 자주 수행되는데, 다음 두 기능의 이용 빈도가 높기 때문이다.

- 주변 사업장 검색
- 사업장 정보 확인

한편 쓰기 연산 실행 빈도는 낮은데, 사업장 정보를 추가하거나 삭제, 편집하는 행위는 빈번하지 않기 때문이다.

읽기 연산이 압도적인 시스템에는 MySQL 같은 관계형 데이터베이스가 바람직할 수 있다. 이제 스키마 설계에 대해 좀 더 자세히 알아보자.

데이터 스키마

이 시스템의 핵심이 되는 테이블은 `business` 테이블과 지리적 위치 색인 테이블(geospatial index table)이다.

business 테이블

`business` 테이블은 사업장 상세 정보를 담는다. 표 1.3을 참고하라. 이 테이블의 기본 키(primary key)는 `business_id`다.

business	
business_id	PK
address	
city	
state	
country	
latitude	
longtitude	

표 1.3 business 테이블

지리적 위치 색인 테이블

지리적 위치 색인 테이블은 위치 정보 관련 연산의 효율성을 높이는 데 쓰인다. 지오해시(geohash)에 대한 지식이 필요하므로, "데이터베이스의 규모 확장" 절에서 다시 논의하도록 하겠다.

개략적 설계

그림 1.2는 개략적 설계안의 다이어그램이다. 이 시스템은 위치 기반 서비스(location-based service, LBS)와 사업장 관련 서비스 두 부분으로 구성된다. 각각의 컴포넌트를 좀 더 자세히 살펴보자.

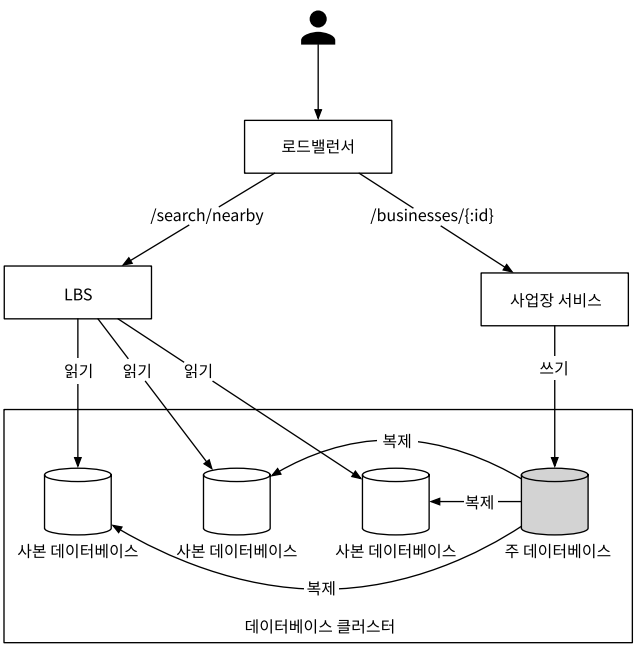


그림 1.2 개략적 설계안

로드밸런서

로드밸런서(load balancer)는 유입 트래픽을 자동으로 여러 서비스에 분산시키는 컴포넌트다. 통상적으로 로드밸런서를 사용하는 회사는 로드밸런서에 단일 DNS 진입점(entry point)을 지정하고, URL 경로를 분석하여 어느 서비스에 트래픽을 전달할지 결정한다.

위치 기반 서비스(LBS)

LBS는 시스템의 핵심 부분으로, 주어진 위치와 반경 정보를 이용해 주변 사업장을 검색한다. 다음과 같은 특징을 갖는다.

- 쓰기 요청이 없는, 읽기 요청만 빈번하게 발생하는 서비스이다.
- QPS가 높다. 특히 특정 시간대의 인구 밀집 지역일수록 그 경향이 심하다.
- 무상태(stateless) 서비스이므로 수평적 규모 확장이 쉽다.

사업장 서비스

사업장 서비스는 주로 다음 두 종류의 요청을 처리한다.

- 사업장 소유주가 사업장 정보를 생성, 갱신, 삭제한다. 기본적으로 쓰기 요청이며, QPS는 높지 않다.
- 고객이 사업장 정보를 조회한다. 특정 시간대에 QPS가 높아진다.

데이터베이스 클러스터

데이터베이스 클러스터는 주-부(primary-secondary) 데이터베이스 형태로 구성할 수 있다. 해당 구성에서 주 데이터베이스는 쓰기 요청을 처리하며, 부 데이터베이스, 즉 사본 데이터베이스는 읽기 요청을 처리한다. 데이터는 일단 주 데이터베이스에 기록된 다음에 사본 데이터베이스로 복사된다. 복제에 걸리는 시간 지연(delay) 때문에 주 데이터베이스 데이터와 사본 데이터베이스 데이터 사이에는 차이가 있을 수 있다. 보통은 그렇더라도 문제가 되지는 않는데, 사업장 정보는 실시간으로 갱신될 필요가 없기 때문이다.

사업장 서비스와 LBS의 규모 확장성

사업장 서비스와 LBS는 둘 다 무상태 서비스이므로 점심시간 등의 특정 시간대에 집중적으로 몰리는 트래픽에는 자동으로 서버를 추가하여 대응하고, 야간 등 유휴 시간 대에는 서버를 삭제하도록 구성할 수 있다. 시스템을 클라우드에 둔다면 여러 지역, 여러 가용성 구역(availability zone)에 서버를 두어 시

스템 가용성을 높일 수 있다.[9] 이에 대해서는 상세 설계를 진행할 때 좀 더 논의하겠다.

주변 사업장 검색 알고리즘

실제로는 많은 회사가 레디스 지오해시(Geohash in Redis)[10]나 PostGIS 확장(extension)[11]을 설치한 포스트그레스(Postgres) 데이터베이스를 활용한다. 면접관은 여러분이 이런 데이터베이스의 내부 구조를 알 거라고 기대하지 않는다. 그러니 그런 데이터베이스의 이름을 나열하기보다는 지리적 위치 색인이 어떻게 동작하는지 설명함으로써 문제 풀이 능력과 기술적 지식을 갖추었음을 보이는 것이 좋다.

이제 다음 순서로는 주변 사업장 검색 방법들을 살펴볼 것이다. 몇 가지 방안을 훑어보고 그 이면의 사고 프로세스(thought process)를 검토한 다음, 각 방안에 어떤 타협적 측면(trade-off)이 존재하는지 논의할 것이다.

방안 1: 2차원 검색

주어진 반경으로 그린 원 안에 놓인 사업장을 검색하는 방법이다.(그림 1.3) 가장 직관적이지만 지나치게 단순하다는 문제가 있다.

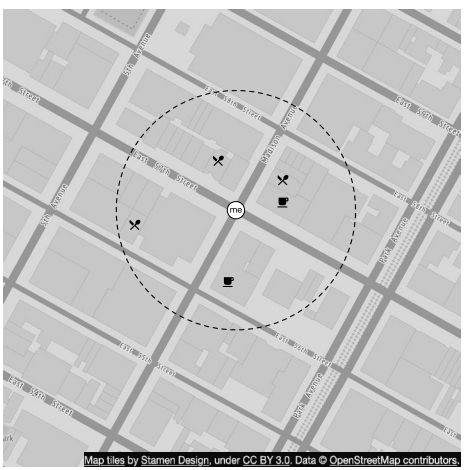


그림 1.3 2차원 검색

이 절차를 유사(pseudo) SQL 질의문으로 옮기면 다음과 같다.

```
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)
```

이 질의는 테이블 전부를 읽어야 하므로 효율적이지 않다. 위도와 경도 칼럼(column)에 색인을 만들어 두면 어떨까? 그렇게 하면 효율이 개선될까? 그래도 썩 좋아지지 않는다. 데이터가 2차원적이므로 칼럼별로 가져온 결과도 여전히 엄청난 양이다. 예를 들어 그림 1.4를 보자. 위도 칼럼과 경도 칼럼에 색인을 만들어 놓으면 데이터 집합 1과 데이터 집합 2는 신속히 추출할 수 있을 것이다. 하지만 주어진 반경 내 사업장을 얻으려면 이 두 집합의 교집합을 구해야 한다. 이 연산은 각 집합에 속한 데이터의 양 때문에 효율적일 수 없다.

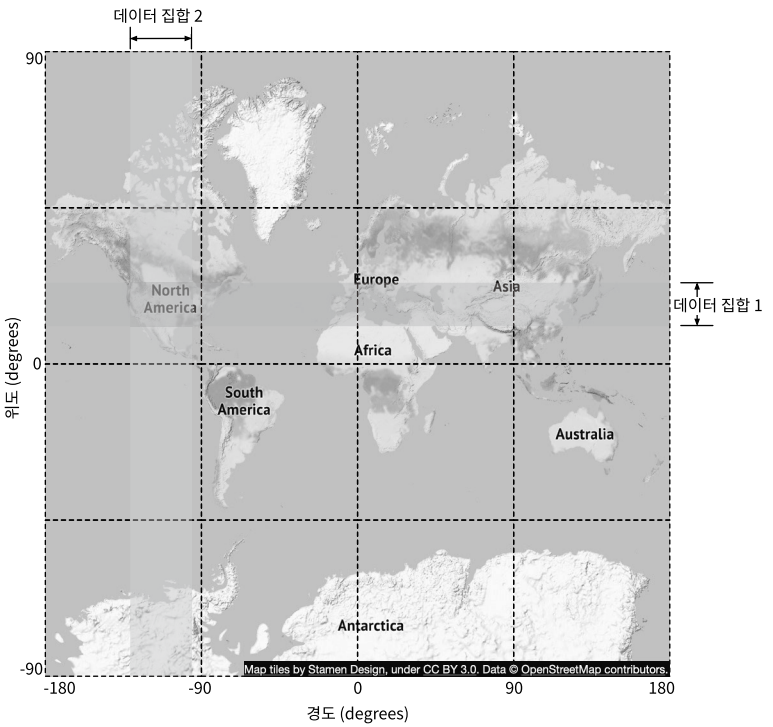


그림 1.4 두 데이터 집합의 교집합