

id	A	B	C	D
1	2	3	x	x
2	4	1	x	x
3	1	5	x	x
4	4	7	x	x
5	8	9	x	x

original table

A	B	id
2	3	1
4	1	2
4	7	4
8	9	5

index

按什么去建index 等价于 按什么去sharding? 这样理解对吗?

@hanli

按什么去建index 等价于 按什么去sharding? 这样理解对吗?

这样理解不对。对于 SQL 数据库而言, 每个column 都可以建 index。index 和sharding 是两个概念。按什么“取数据”才按什么sharding。

如果针对column A和B建立了联合索引, 在进行某个只针对col A的query的时候是不是也会调用联合索引table, 比如select col_A from table where col_A = xxx? 是否可以认为如果联合索引已经包括一个column的话, 就不需要为其单独来做索引了? 另外, 如果一个column存在多个索引table, 会使用哪个index table呢?

你举的例子是没问题的, 是会按照联合索引来检索A, 但是得出的结论 **是否可以认为如果联合索引已经包括一个column的话, 就不需要为其单独来做索引了** 是不对的, 因为如果联合索引是按照 A 在前 B 在后的方式的话, 那么你检索 B 还是用不上这个索引的, 只有检索 A 用得上。

如果一个 column 在多个索引上都在最前面, 检索的时候会根据索引的大小来挑选一个索引小的, 因为索引小的查的快。

联合索引, where clause的先后顺序有关系吗, where A=1 AND B >10; where B>10 and A =1. 数据库会自己优化吗?

助教 - 江畔 @九章用户 ZX6GDB

联合索引, where clause的先后顺序有关系吗, where A=1 AND B >10; where B>10 and A =1. 数据库会自己优化吗?

没有关系的, SQL执行优化器会优化语句的, 实际执行的时候会改成索引的顺序。

手机能通过 GPS 或者北斗卫星直接计算出来的与地理位置有关的信息有?

A: 经度 Longitude	B: 纬度 Latitude
C: 海拔 Altitude	D: 邮编 Zipcode
E: 城市 City	F: 国家 Country
G: Geohash	H: Google S2

已回答 我不会

没有答对哦! 正确答案是 A B C , 有19%的同学做对了这道题目哦, 继续努力!

基于地理位置的服务 Location Based Service

二维坐标 (x, y) 的正确排序方法是

A: x 不等的按照 x 排序, x 相等的按照 y 排序	B: y 不等的按照 y 排序, y 相等的按照 x 排序
C: 按照 x + y 排序	D: 按照离 (0, 0) 点的距离排序

已回答 我不会

恭喜你, 答对啦~正确答案是 A , 有35%的同学答错了, 你比他们厉害哦。

课程版本 v6.0 主讲 东邪

A: Memcached	B: Cassandra
C: Redis	D: MongoDB

已回答 我不会

答错了~正确答案就是 C , 你对这个知识点的掌握超过了50%的同学

假设我们在 User Table 中建立 city + birth_year 的复合索引, city 为第一关键字, birth_year 为第二关键字。

A: city = 'Beijing' AND birth_year = 2017	B: city = 'Beijing' AND birth_year >= 2017 AND birth_year <= 2019
C: birth_year >= 2017 AND birth_year <= 2019	D: birth_year = 2018
E: city >= 'Beijing' AND city <= 'Shanghai'	F: city = 'Beijing'
G: city >= 'Beijing' AND city <= 'Shanghai' AND birth_year >= 2017 and birth_year <= 2019	

已回答 我不会

答错了! 正确答案是 A B E F , 有6%的同学答对了, 加油赶上他

- Design Uber
 - Design Facebook Nearby
 - Design Yelp
 - Design Pokemon Go



Interviewer: Please design Uber

Similar questions:

How to design facebook nearby

How to design yelp



- RingPop
 - <https://github.com/uber/ringpop-node>
 - 一个分布式架构
 - 扩展阅读
 - <http://ubr.to/1S47b8g> [Hard]
 - <http://bit.ly/1Yg2dnd> [Hard]
- TChannel
 - <https://github.com/uber/tchannel>
 - 一个高效的RPC协议
 - RPC: Remote Procedure Call
- Google S2
 - <https://github.com/google/s2-geometry-library-java>
 - 一个地理位置信息存储与查询的算法
- Riak
 - Dynamo DB 的开源实现



告诉我你看到这些词的感受是不是这样？

[多选题] 【系统设计2020】现有的比较成熟的、广泛使用的RPC协议有哪些？

- | | |
|-------------|-----------|
| A. HTTP | 23.82% 选择 |
| B. Thrift | 40.72% 选择 |
| C. ProtoBuf | 35.46% 选择 |

✖ 答错了，您选择的答案是 A B

正确答案: A B C

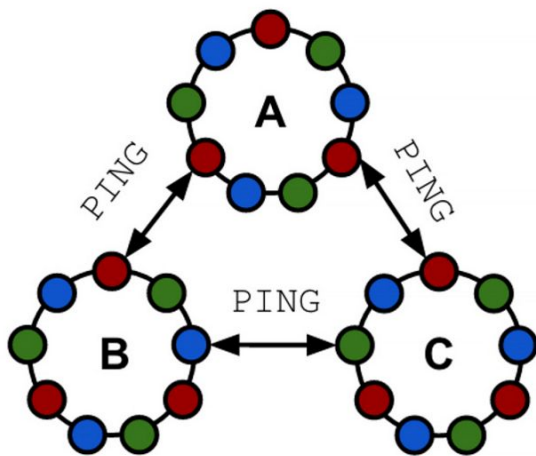
解析: Thrift 是 Facebook 的, ProtoBuf 是 Google 的。
Http 也可以认为是一个 RPC。

Read more on Uber Eng Blog

<http://eng.uber.com/>



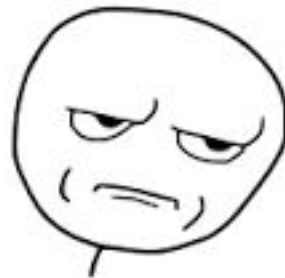
是不是答出Uber是怎么实现的, 就可以拿到Offer?



系统设计面试常见误区

以为答出该公司是怎么做的，就可以拿到Offer了

你他妈的在逗我？



Uber的架构非常小众

Uber用到的技术是自己设计出的一套东西

如果Uber面你这个题，你不可能比他们清楚，并且显得你是准备过的，不能代表你真实的能力

如果其他公司面你这个题，这也不会是期望答案

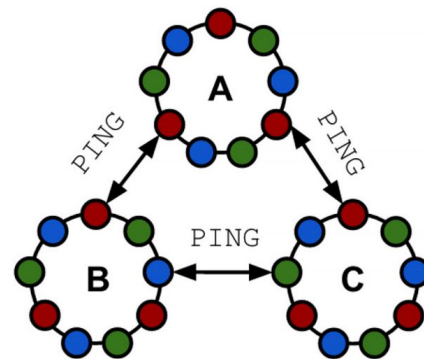
- Scenario 场景
 - Features
 - QPS / Storage
- Service 服务
 - Service Oriented Architecture
- Storage 数据
 - Schema

逻辑设计 Logic Design 50%
Make it work!



- Scale 进化
 - Robust
 - Feature

架构设计 Infrastructure Design 50%
Make it robust!



System Design = Logic Design + Infrastructure Design

系统设计 = 逻辑设计 + 架构设计



Scenario 场景

需要设计哪些功能，设计得多牛

试试看，你觉得对于一个打车软件来说最重要的是哪些功能？

- 第一阶段：
 - Driver report locations
 - Rider request Uber, match a driver with rider
- 第二阶段：
 - Driver deny / accept a request
 - Driver cancel a matched request
 - Rider cancel a request
 - Driver pick up a rider / start a trip
 - Driver drop off a rider / end a trip

- 第三阶段 *:

- Uber Pool
- Uber Eat

无所谓的加分项, 如果你都答到这里了, 说明你前面都秒杀了



Scenario - 设计得多牛？

可以直接向面试官询问一些基本数据

Uber has become a global service providing roughly **15 million** rides per day across 500 cities, and international markets are growing as fast as ever. So, if Uber is completing **15 million** rides per day (worldwide) with **2 million drivers**, that means there are 7.5 passengers per driver – on average.

2018年11月8日



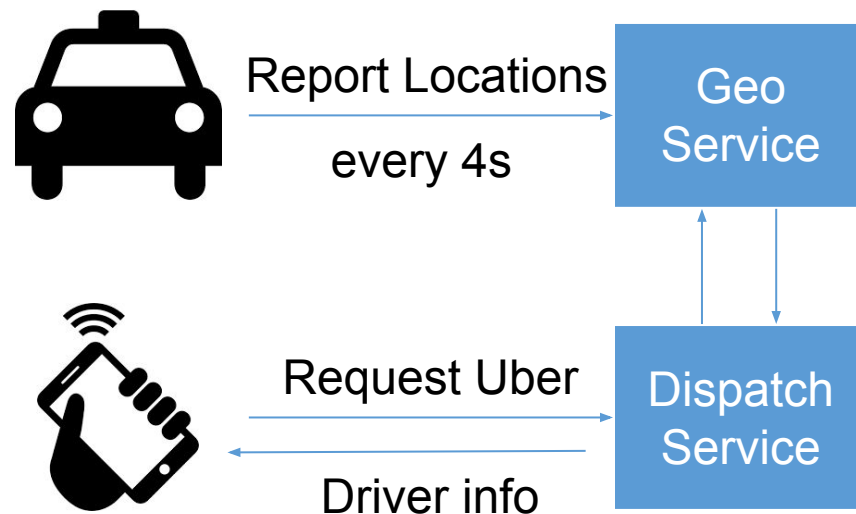
How Many Uber Drivers are There? - Ridester.com

<https://www.ridester.com/how-many-uber-drivers-are-there/>

- 2018年每天有 2M 司机载客
- 假设同时在线的司机平均约为 600k(猜的)
 - Average Driver QPS = $600k / 4 = 150k$
 - Driver report locations by every 4 seconds
 - Peak Driver QPS = $150k * 2 = 300k$
 - Uber 官方自己的说法:2015 新年夜的 Peak QPS 是 170K, 当时 Uber 约有 1M 的司机
 - Read More: <http://bit.ly/1FBSgMK>
 - Rider QPS 可以忽略
 - 无需随时汇报位置
 - 一定远小于Driver QPS
- 存储估算
 - 假如每条Location都记录: $600k * 86400 / 4 * 100\text{bytes}$ (每条位置记录) ~ 1.3 T / 天
 - 假如只记录当前位置信息: $600k * 100\text{bytes} = 60\text{M}$

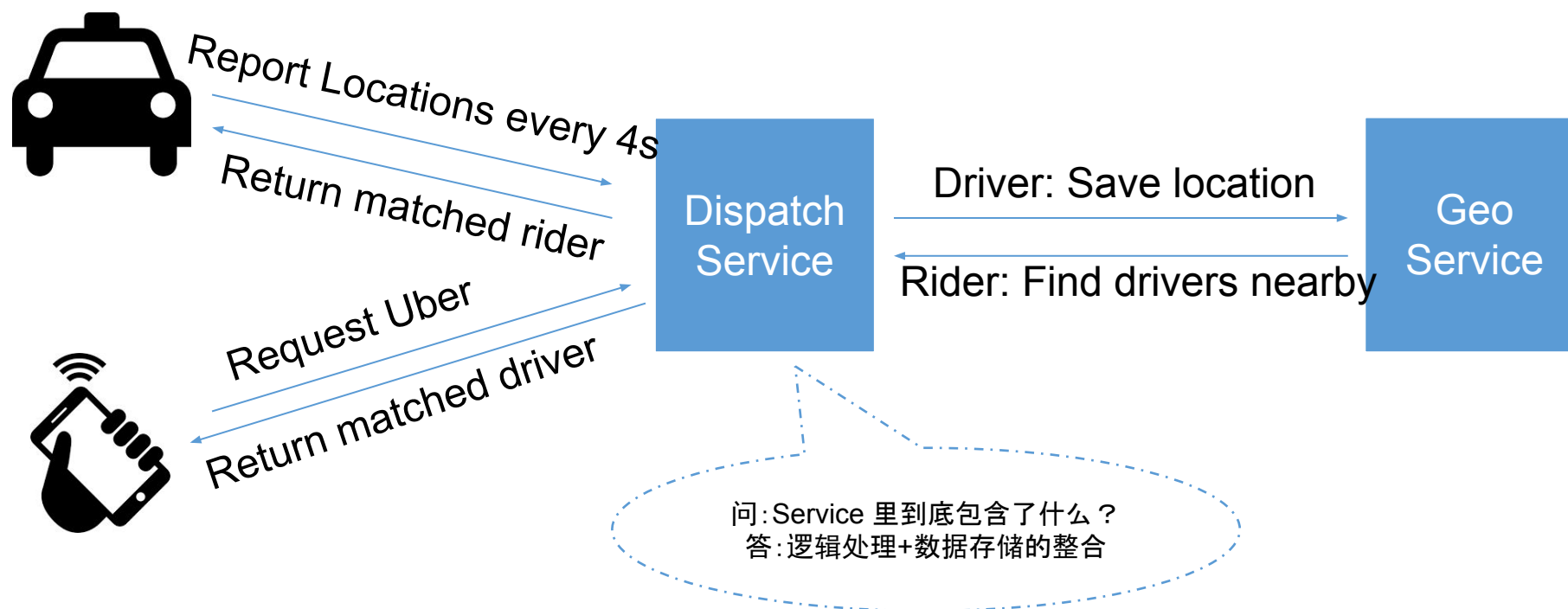
初步感觉:300k 的写操作是不容小觑的
必须找一个写速度快的存储!

- Uber 主要干的事情就两件
 - 记录车的位置 GeoService
 - 匹配打车请求 DispatchService



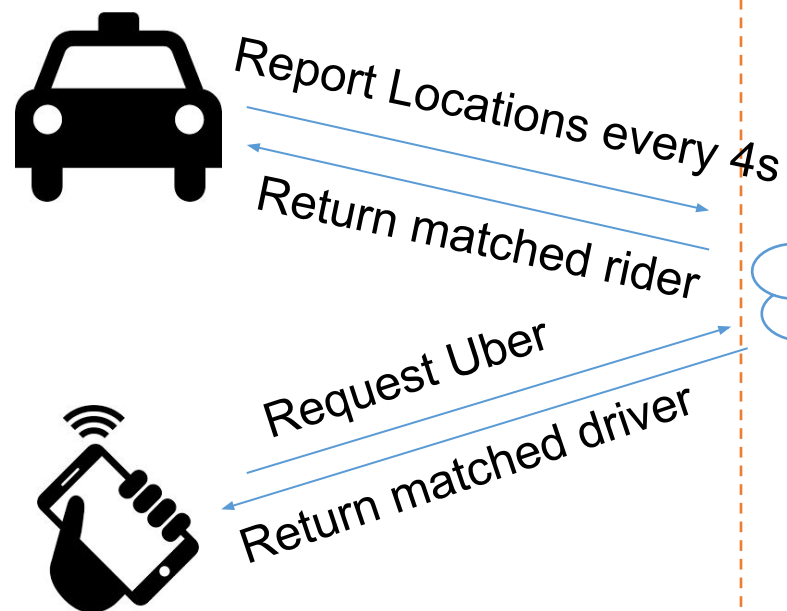
这张图里漏了什么？

- Driver 如何获得打车请求？
 - Report location 的同时, 服务器顺便返回匹配上的打车请求

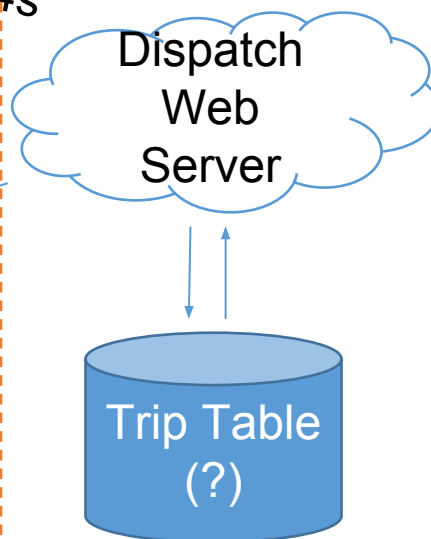


SearchNearbyDriver()需要过滤非空闲的 driver 吗? 如果过滤在DisptachService里实现, Trip Table需要加哪些索引优化查询? trip.driver_id列可能为空, 可以按 (driver_id, status)建复合索引吗?

可以过滤非空闲的 drive。过滤非空闲的 driver 应该不是在 Trip table 中, 而是在 location table 中, 可以加一个 status 字段, 来表示司机的状态, 在 Trip table 中是无法实现搜索附近空闲司机的功能的。不过这里的 location table 也不是最优化的表单设计, 后面引入 geohash 之后, 我们可以对 geohash 和 status 建立联合索引。



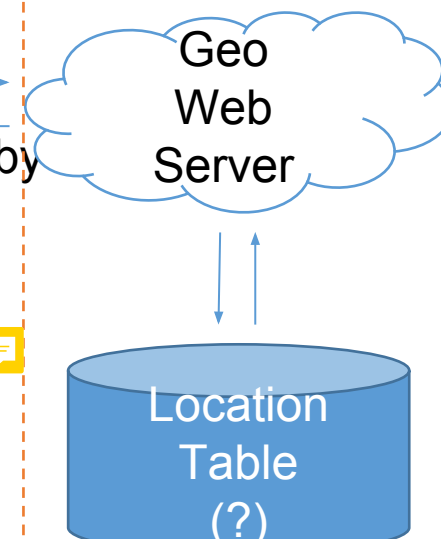
Dispatch Service



读 vs 写?

Driver: Save location
Rider: Find drivers nearby

Geo Service



读 vs 写?

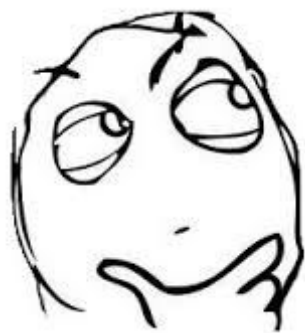

```
class Trip {  
    public Integer tripId;  
    public Integer driverId, riderId;  
    public Double startLatitude, startLongitude;  
    public Double endLatitude, endLongitude;  
    public Integer status;  
    public Datetime createdAt;  
}
```

```
class Location {  
    public Integer driverId;  
    public Double Latitude, Longitude;  
    public Datetime updatedAt;  
}
```

			Location Table	type	comments
			driver_id	fk	Primary key
			lat	float	纬度
			lng	float	经度
			updated_at	timestamp	存最后更新的时间
Trip Table	type	comments			
id	pk	primary key			
rider_id	fk	User id			
driver_id	fk	User id			
start_lat	float	起点的纬度 Latitude			
start_lng	float	起点的经度 Longitude			
end_lat	float	终点的维度 Latitude			
end_lng	float	终点的经度 Longitude			
created_at	timestamp	创建时间			
status	int	New request / waiting for driver / on the way to pick up / in trip / cancelled / ended			

LBS 类系统的难点： 如何存储和查询地理位置信息？

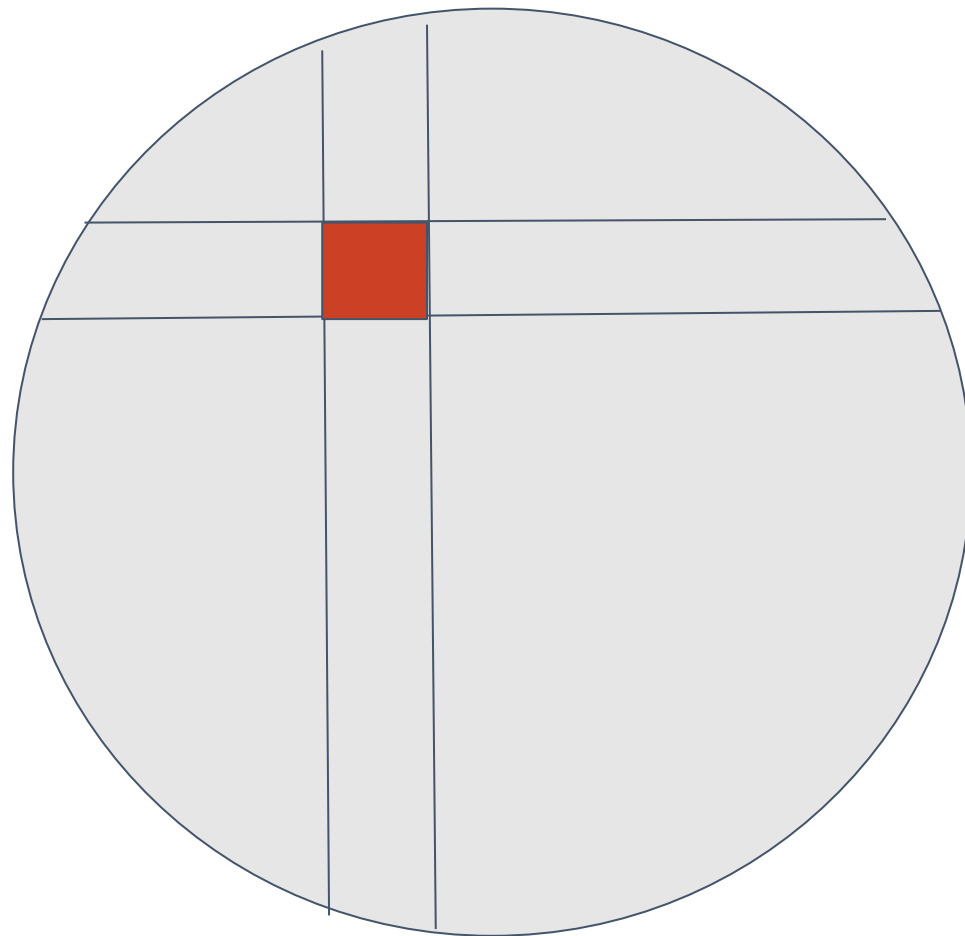
如，查询某个乘客周围 X 公里内的司机



```
SELECT * FROM Location
WHERE lat < myLat + delta
      AND lat > myLat - delta
      AND lng < myLng + delta
      AND lng > myLng - delta;
```

问: 分别对 lat 和 lng 建 index 是否可行? 

```
create index lat_idx on location_table(lat);
create index lng_idx on location_table(lng);
```



复合索引 Composite Index 能否解决问题？

什么是复合索引 —— 将多个 columns 合并起来做索引

```
create index lat_lng_idx on location_table(lat, lng);
```

复合索引 Composite Index 能否解决问题？

什么是复合索引 —— 将多个 columns 合并起来做索引

```
create index lat_lng_idx on location_table(lat, lng);
```

不行，复合索引只能解决 “lat=固定值 and lng 在某个范围” 的查询

数据库的 index 只能解决一个维度上的 Range Query

多个独立的维度的 Range Query 无法高效查询

解决思路: 把二维映射到一维

- Google S2

- Read more: <http://bit.ly/1WgMpSJ>
- Hilbert Curve: <http://bit.ly/1V16HRa>
- 将地址空间映射到 2^{64} 的整数
- 特性: 如果两个一维整数比较接近, 对应的二维坐标就比较接近
- Example: $(-30.043800, -51.140220) \rightarrow 10743750136202470315$

更精准, 库函数API丰富

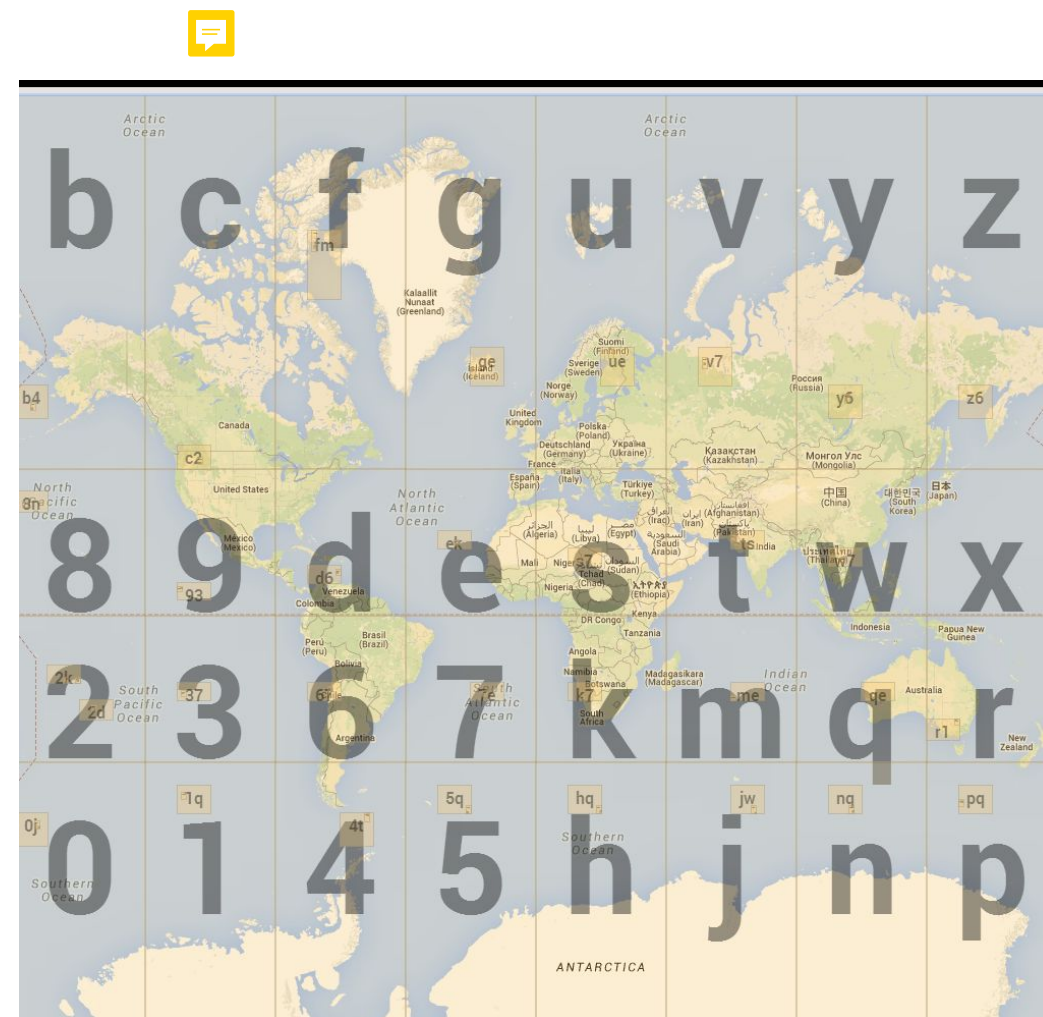
- Geohash

- Read more: <http://bit.ly/1S0Qzeo>
- Peano Curve
- Base32: 0-9, a-z 去掉 (a,i,l,o)
 - 为什么用 base32 ? 因为刚好 2^5 可以用 5 位二进制表示
- 核心思路二分法
- 特性: 公共前缀越长, 两个点越接近
- Example: $(-30.043800, -51.140220) \rightarrow 6feth68y4tb0$

比较简单, 准确度差一些

- Examples:
- LinkedIn HQ: 9q9hu3hhsjxx
- Google HQ: 9q9hvu7wbq2s
- Facebook HQ: 9q9j45zvr0se

geohash length	lat bits	lng bits	lat error	lng error	km error
1	2	3	± 23	± 23	± 2500
2	5	5	± 2.8	± 5.6	± 630
3	7	8	± 0.70	± 0.7	± 78
4	10	10	± 0.087	± 0.18	± 20
5	12	13	± 0.022	± 0.022	± 2.4
6	15	15	± 0.0027	± 0.0055	± 0.61
7	17	18	± 0.00068	± 0.00068	± 0.076
8	20	20	± 0.000085	± 0.00017	± 0.019



- 北海公园: lat=39.928167, lng=116.389550
- 二分(-180,180) 逼近经度┆左半部记0┆右半部记1
 - (-180, 180)里116.389550在右半部 → 1
 - (0, 180)里116.389550在右半部 → 1
 - (90, 180)里116.389550在左半部 → 0
 - (90, 135)里116.389550在右半部 → 1
 - (112.5, 135)里116.389550在左半部 → 0
- 二分(-90,90) 逼近纬度, 下半部记0, 上半部记1
 - (-90,90) 里 39.928167 在上半部 → 1
 - (0,90) 里 39.928167 在下半部 → 0
 - (0,45) 里 39.928167 在上半部 → 1
 - (22.5,45) 里 39.928167 在上半部 → 1
 - (33.75,45) 里 39.928167 在上半部 → 1
 - ... (还可以继续二分求获得更多的精度³)

课后作业:

<http://www.lintcode.com/problem/geohash/>

先经后纬, 经纬交替

1110011101

W X

查询Google半径2公里内的车辆

- 找到精度误差 > 2公里的最长长度



geohash length	lat bits	lng bits	lat error	lng error	km error
1	2	3	± 23	± 23	± 2500
2	5	5	± 2.8	± 5.6	± 630
3	7	8	± 0.70	± 0.7	± 78
4	10	10	± 0.087	± 0.18	± 20
5	12	13	± 0.022	± 0.022	± 2.4
6	15	15	± 0.0027	± 0.0055	± 0.61
7	17	18	± 0.00068	± 0.00068	± 0.076
8	20	20	± 0.000085	± 0.00017	± 0.019



- Google HQ: 9q9hvu7wbq2s
- 找到位置以9q9hv以开头的所有车辆



怎样在数据库中实现该功能？

- SQL 数据库
 - 首先需要对 geohash 建索引
 - CREATE INDEX on geohash;
 - 使用 Like Query
 - SELECT * FROM location WHERE geohash LIKE `9q9hv%`;
- NoSQL - Cassandra
 - 将 geohash 设为 column key
 - 使用 range query (9q9hv0, 9q9hvz)
- NoSQL - Redis 
 - Driver 的位置分级存储
 - 如 Driver 的位置如果是 9q9hvt, 则存储在 9q9hvt, 9q9hv, 9q9h 这 3 个 key 中
 - 6位 geohash 的精度已经在一公里以内, 对于 Uber 这类应用足够了
 - 4位 geohash 的精度在20公里以上了, 再大就没意义了, 你不会打20公里以外的车
 - key = 9q9hvt, value = **set** of drivers in this location



NoSQL - Redis

数据可持久化

原生支持list, set等结构

读写速度接近内存访问速度 >100k QPS



司机的位置变化会很频繁吧？那这样redis数据库应该读写压力很大吧？

这是一个写多读少的情况，对于司机的读请求一般只发生在用户 request Uber，去匹配一个司机的时候，司机的位置更新信息是需要定时写入数据库中的，所以写入压力会比较大。为此我们可以对数据库做 sharding，来分摊写请求，并且用 cache through 架构来优化。

- 用户发出打车请求, 查询给定位置周围的司机
 - $(lat, lng) \rightarrow geohash \rightarrow [driver1, driver2, \dots]$
 - 先查6位的 geohash 找0.6公里以内的
 - 如果没有 再查5位的 geohash 找2.4公里以内的
 - 如果没有 再查4位的 geohash 找20公里以内的

Location Table	
key	geohash
value	{driver1_id, driver2_id, driver3_id ...}

- 匹配司机成功, 用户查询司机所在位置
 - $driver1 \rightarrow (lat, lng)$

Driver Table	
key	driver_id
value	(lat, lng, status, updated_at, trip_id)



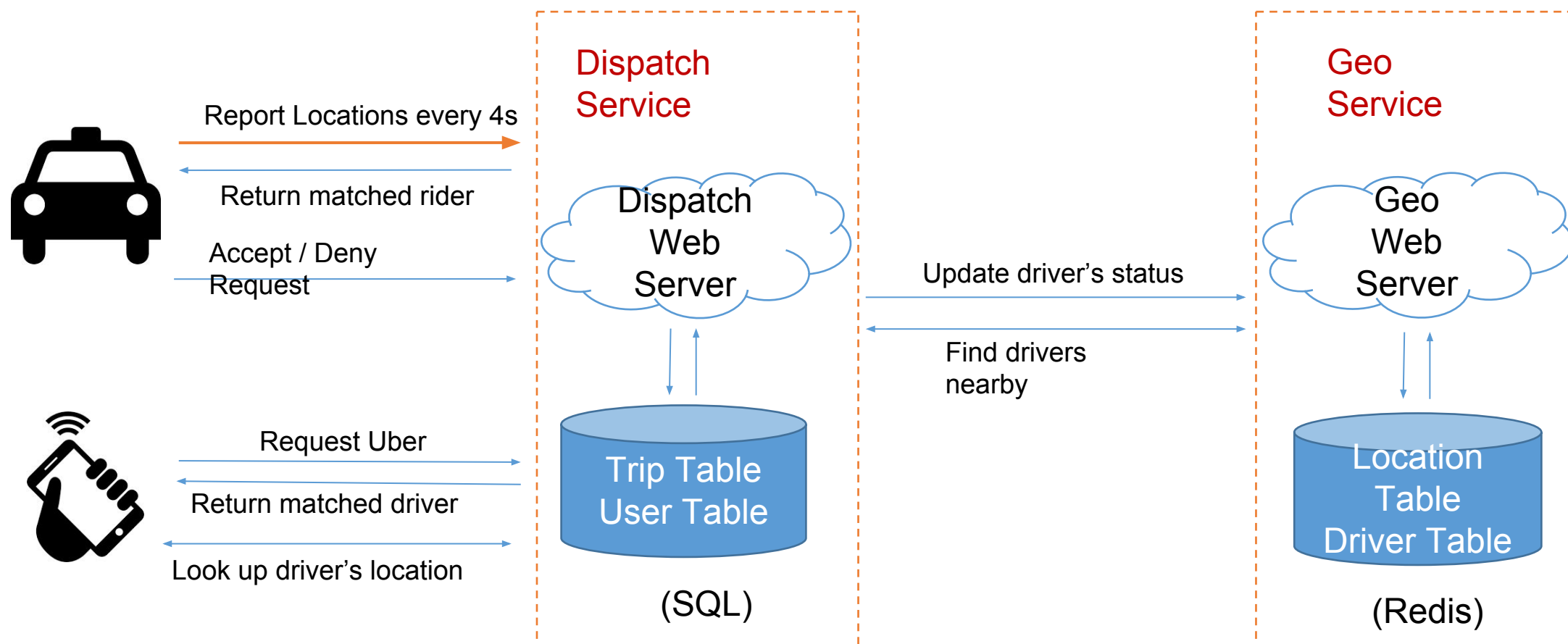
指向UserTable, UserTable存在其他数据库中, 可以是SQL数据库

- 司机汇报自己的位置
 - 计算当前位置 lat, lng 的 geohash
 - geohash4, geohash5, geohash6
 - 查询自己原来所在的位置
 - geohash4', geohash5', geohash6'

} 对比是否发生变化
并将变化的部分在 Redis 中进行修改

 - 在 Driver Table 中更新自己的最后活跃时间
- 司机接受打车请求
 - 修改 Trip 状态
 - 用户发出请求时就已经在 Trip Table 中创建一次旅程 T 并 Match 上最近的司机
 - 在 Driver Table 中标记自己的状态进入不可用状态
- 司机完成接送 T 结束一次 Trip
 - 在 Trip Table 中修改旅程状态
 - 在 Driver Table 中标记自己的状态进入可用状态

1. 乘客发出打车请求, 服务器创建一次Trip
 - 将 trip_id 返回给用户
 - 乘客每隔几秒询问一次服务器是否匹配成功
2. 服务器找到匹配的司机, 写入Trip, 状态为等待司机回应
 - 同时修改 Driver Table 中的司机状态为不可用, 并存入对应的 trip_id
3. 司机汇报自己的位置
 - 顺便在 Driver Table 中发现有分配给自己的 trip_id
 - 去 Trip Table 查询对应的 Trip, 返回给司机
4. 司机接受打车请求
 - 修改 Driver Table, Trip 中的状态信息
 - 乘客发现自己匹配成功, 获得司机信息
5. 司机拒绝打车请求
 - 修改 Driver Table, Trip 中的状态信息, 标记该司机已经拒绝了该trip
 - 重新匹配一个司机, 重复第2步



Scale 拓展

看看有哪些问题没有解决, 需要优化
出现故障怎么办

有什么隐患？

需求是 300k QPS

Redis 的读写效率 > 100k QPS

是不是 3-4 台就可以了？

Interviewer: Redis server is down?

随便挂一台，分分钟损失几百万\$ 💬



DB Sharding

目的1: 分摊流量

目的2: Avoid Single Point Failure



按照什么来Sharding?

[单选题] 【系统】司机的 Location 信息应该按照什么进行 Sharding?

- A. 按照 User Id 12.03% 选择
- B. 整个 GeoHash 13.23% 选择
- C. GeoHash的前4位 59.97% 选择
- D. GeoHash的前6位 14.78% 选择

✖ 答错了, 您选择的答案是 B

正确答案: C

解析: 按照 User Id 是显然不对的, 因为乘客的 UserId 和他附近的司机的 UserId 可能会被 sharding 到不同的地方。按照 GeoHash 进行 Sharding 的话, 只需要按照前 4 位进行 sharding 即可。因为乘客的查询可能是按照前 4, 5, 6 位 GeoHash 进行查询, 如果是按照前 6 位进行 sharding 的话, 那么 BBBB22 和 BBBB33 就会被拆分开, 而我们查询的时候其实希望如果用户在 BBBB22 的话, 他能够查到位置在 BBBB33 的司机。

关闭

传统做法: 按照前4位 Geohash

数据怎么查询就怎么拆分

查询是按照 4-6位的 geohash

那么拆分就可以按照 4位的 geohash 来 sharding



Uber 的做法：按城市 Sharding

难点1：如何定义城市？

难点2：如何根据位置信息知道用户在哪个城市？



为什么不能按照其他的比如 user_id 来 sharding？

Geo Fence

- 用多边形代表城市的范围
- 问题本质: 求一个点是否在多边形内
 - 计算几何问题
- 城市的数目: 500个
- 乘客站在两个城市的边界上怎么办?
 - 找到乘客周围的2-3个城市
 - 这些城市不能隔太远以至于车太远
 - 汇总多个城市的查询结果
 - 这种情况下司机的记录在存哪个城市关系不大

多边形地理位置和city的mapping存在哪? 也用geo server query么?

助教 - 江畔 助教

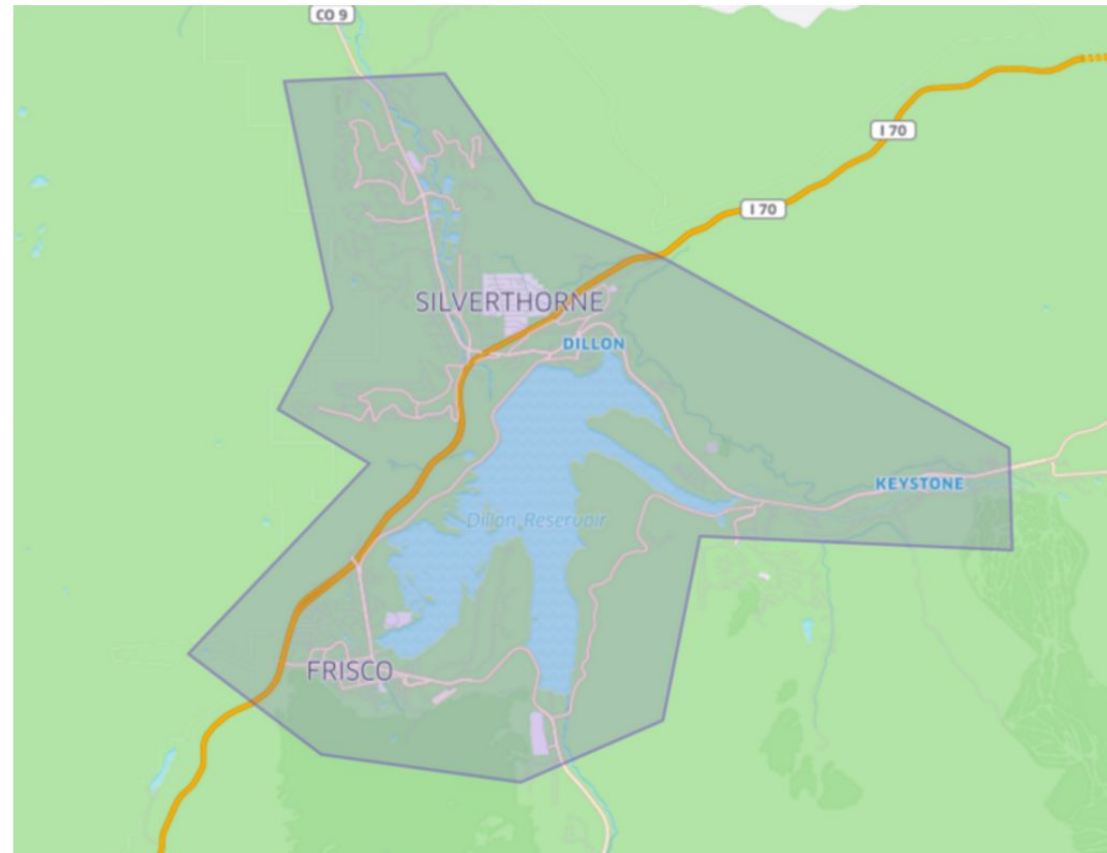
@jiuzhang123

多边形地理位置和city的mapping存在哪? 也用geo server query么?

如果是用 SQL 数据库存储, 可以再 location table 中再加一个 city_id 的字段, 用于表示多边形地理位置和 city 的 mapping 信息, 或者单独建一张表来存储 mapping 信息。

如果是 NoSQL - Cassandra, 可以用 row key 存储 city_id。

如果是 NoSQL - Redis, 可以把 city_id 作为 key, geohash 作为 value, 来表示 mapping 信息。



优化之前, 时间复杂度是 $O(\text{NumberOfCities} * \text{AverageFencesPerCity})$

```
for city in all_cities:
    for fence in city.all_fences:
        if check_point_in_fence(point, fence):
            return fence
```

优化之后, 时间复杂度是 $O(\text{NumberOfCities} + \text{AverageFencesPerCity})$

```
for city in all_cities:
    if check_point_in_city(point, city):
        break

for fence in city.all_fences:
    if check_point_in_fence(point, fence):
        return fence
```

Interviewer: How to check rider is in Airport?

同样可以用Geo Fence

类似机场这样的区域有上万个, 直接 $O(N)$ 查询太慢

分为两级Fence查询, 先找到城市, 再在城市中查询Airport Fence

Read More: <http://ubr.to/20qK4F4>

Interviewer: How to reduce impact on db crash?

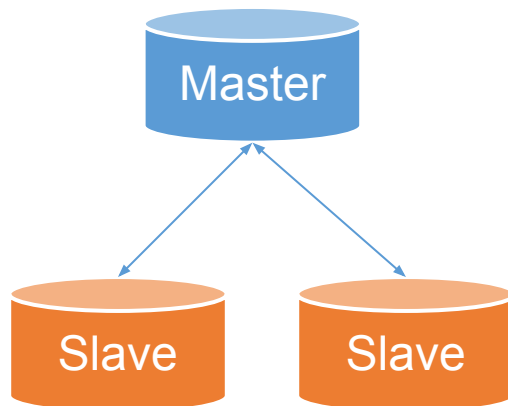
多台 Redis 虽然能减少损失
但是再小的机器挂了, 都还是会影响



如何减小损失?

方法1: Redis Master Slave

每个 Master 下挂两个 Slave
Master 挂了 Slave 就顶上



方法2: 让更强大的 NoSQL 数据库来处理

既然一定要用多台机器的话, 我们可以用 1000 台 Cassandra / Riak
这样的 NoSQL 数据库, 平均每台分摊 300 的 QPS 就不大了
这类数据库会帮你更好的处理 Replica 和挂掉之后恢复的问题

- <http://www.lintcode.com/problem/mini-yelp/>
- <http://www.lintcode.com/problem/geohash/>
- <http://www.lintcode.com/problem/geohash-ii/>
- <http://www.lintcode.com/problem/mini-cassandra/>

附录: 扩展阅读

- Uber’s Early Architecture
 - <http://bit.ly/1Q1IzGL> [Easy] [Video]
- Scaling Uber’s Real-time Market Platform
 - <http://bit.ly/1FBSgMK> [Medium] [Video]
- RingPop
 - <http://ubr.to/1S47b8g> [Hard] [Blog]
 - <http://bit.ly/1Yg2dnd> [Hard] [Video]
- Point in polygon
 - <http://bit.ly/1N1Zjlu> wiki
- Dynamo DB
 - <http://bit.ly/1mDs0Yh> [Hard] [Paper]

关于 Google S2，下列说法“不正确”的是

- A: Google S2 是一个地理位置信息的转换算法，能够将经纬度信息转换为一个大整数
- B: Google S2 比 GeoHash 准确度更高
- C: Google S2 的计算结果越接近，原始经纬度坐标就越接近
- D: 原始经纬度坐标越接近，Google S2 的计算结果就越接近

已回答 我不会

有点难吗，答错了呢。正确答案是 D，有53%的同学超过了你，但是千万不要气馁。

筛选进行中的 Trip，该如何创建 index?

查询北京的所有 Trip 里，正在进行的 Trip。
`SELECT * FROM trip_table WHERE city='beijing' AND status='in_trip';`

- A: 按照 city 建 index: `CREATE INDEX city_index ON trip_table(city);`
- B: 按照 status 建 index: `CREATE INDEX status_index ON trip_table(status);`
- C: 按照 city 和 status 分别建 index: `CREATE INDEX city_index ON trip_table(city);`
`CREATE INDEX status_index ON trip_table(status);`
- D: 按照 city 和 status 建 composite index: `CREATE INDEX city_status_index ON trip_table(city, status);`

已回答 我不会

正确答案是 D，有67%的同学答对了，要加油了。

关于 Database 的 Master Slave 架构，下列说法正确的有

- A: 一个 Master 只能有两个 Slaves
- B: Master 和 Slave 都可以进行读写操作
- C: Master 和 Slave 之间是定期同步数据
- D: Master 和 Slave 之间是通过 Write Ahead Log 同步数据
- E: Master 和 Slave 之间的数据是没有延迟的
- F: 读 Master 会比读 Slave 更快
- G: Master 会负责管理 Slave，如果 Slave 挂了，会自动重启一台新的 Slave

已回答 我不会

正确答案是 D，有5%的同学做对了这道题目哦，继续努力！

筛选进行中的 Trip 并排序，该如何创建 index?

查询北京的所有 Trip 里，正在进行的 Trip。
`SELECT * FROM trip_table WHERE city='beijing' AND status='in_trip' ORDER BY created_at LIMIT 20;`

- A: 按照 city 和 status 建 composite index: `CREATE INDEX city_status_index ON trip_table(city, status);`
- B: 按照 city, status, created_at 建 composite index: `CREATE INDEX city_status_created_at_index ON trip_table(city, status, created_at);`
- C: 按照 city, status 建 composite index (选项A)，再对 created_at 建 index: `CREATE INDEX created_at_index ON trip_table(created_at);`

已回答 我不会

正确答案是 B，有42%的同学答对了，加油赶上他们！

SQL 要被高效执行的话，所查询的数据，必须至少在某个 index 上是被连续放在一起的。我们查询希望的是按照 city, status 进行筛选，并按照 created_at 排序取出前 20 个数据，因此我们希望被查询数据按照 <city, status, created_at> 三元组进行排序。这样那些 city, status 满足筛选条件的 trip，才会按照第三关键字 created_at 进行排序，才能快速宣传前 20 个。