**Scalability Outline:**

**Introduction:**

How does Facebook handle 500 million users a day with PHP, a slower processing code?

How does Yelp maintain and return searches from thousands of daily users from millions of reviews?

How does Netflix maintain its millions terabytes of data and return your content within seconds?

**Overview:**

Objectives:

Understand common system architecture of applications

Understand what causes an application to not be able to scale and why

Understand the common *bottlenecks* of a system/server/network/database and why

Be able to resolve these *bottlenecks*

Understand scaling systems horizontally, not vertically

Understand the data management and tradeoffs of reading time vs writing time

Difference between response and fetch time in MySQL

What is scalability?

The ability of a system to handle increase loads.

We will handle each part of a system separately, identifying and resolving scaling issues.

***Database***:

Making sure your database is set up correctly in the first place is essentially. After realizing your app has grown and then making database changes can be drastic. Imagine if Netflix had to change its schema and the updates it would have to do to make sure no data is loss between its schemas. It is important to think about what your app is predominately doing to prepare for tradeoffs like insert times vs retrieval times.

**ERD:**

Download the file and run the SQL schema creation script as well as scalability\_insert\_1.js using node.

Create a database with this SQL script and run this insert script.

Fix this database so that you can get these pieces of information, in under a second:

The count of all the business

The top favorited business

The top 5 highest profitable business in each sector

The most dislike business in SECTOR and closest to the EQUATOR

The all businesses in a specific country

The top 10 profitable business in any country

**-*UPDATE scalability\_bad\_erd.businesses***

***SET name = "minh's company",***

***owner\_first\_name = "Minh",***

***owner\_last\_name = "Nguyen",***

***created\_at = NOW(),***

***updated\_at = NOW(),***

***image\_link = "hello.com"***

***WHERE image\_link = "" –***

***Index, maximize queries, load balance/separate read and write***

Part 1 – Optimizing the database (4 hours)

1. Objective: Have students understand the power of using the right field type. Give them a crappy database with un-necessary text, etc.
2. Objective: Have students understand the power of index. How: Give them a database with no index and have them run 3-5 queries. Have them record how long each of these queries took and how long it took after creating an index.
3. Objective: Have students understand the power of foreign keys. How: Give them 3-5 queries with bunch of join statements where only the primary key is indexed. Have some queries that are using GROUP BY \_\_\_\_ or ORDER BY \_\_\_\_ WHERE \_\_\_\_ is not a primary key but could be but also provide cases where foreign keys are not used properly. Have them record before and after time performance of these queries.
4. Objective: Come up with 3-5 very inefficient queries (have some that are inner nested) and have the student change the queries. (SELECT \* FROM (SELECT \*)).
5. Objective: Power of my.cnf – configuring the database properly. Give them a task where they can see speed improvements and show about caching.
6. Objective: Have students understand the power of sharding. How: Give them one giant table and have them run 3-5 queries. Have them record how long each of these quries took and how long it took after sharding the data. Have them explore at least 2 different ways to shard the data.
7. Objective: Power of load balancer

Part 2 – Connecting with the backend (4-8hrs)

1. Have the homepage display certain information as wireframed. Have a for loop that does a query inside each for loop. If N records exist, have it do ~N number of queries to get the desired stuff. Have the student fix this to be 1 query and record the load difference. Good example could be for(i=0; i<count(cities); i++) { cities[] = total number of business in each city obtained from the query} sorting cities and displaying the top 10.
2. Have the homepage display a few other information with inefficient queries. Have the student optimize these queries.
3. Have the homepage include this new algorithm that’s super inefficient (N^3). Have the user optimize this algorithm and record the difference.
4. Implement caching and record the difference.
5. Have the homepage display large pictures and videos (each picture being like 10MB, etc). Have them fix this. Show cmdabout compression. Send 50 different CSS files and have them measure load time. Group all the CSS into a single file and then measure load time.
6. Implement event based vs thread based web server.
7. Implement load balancer

Ever wonder how does Facebook handle 500 million users a day or yelp handle thousands of searches through millions of record?