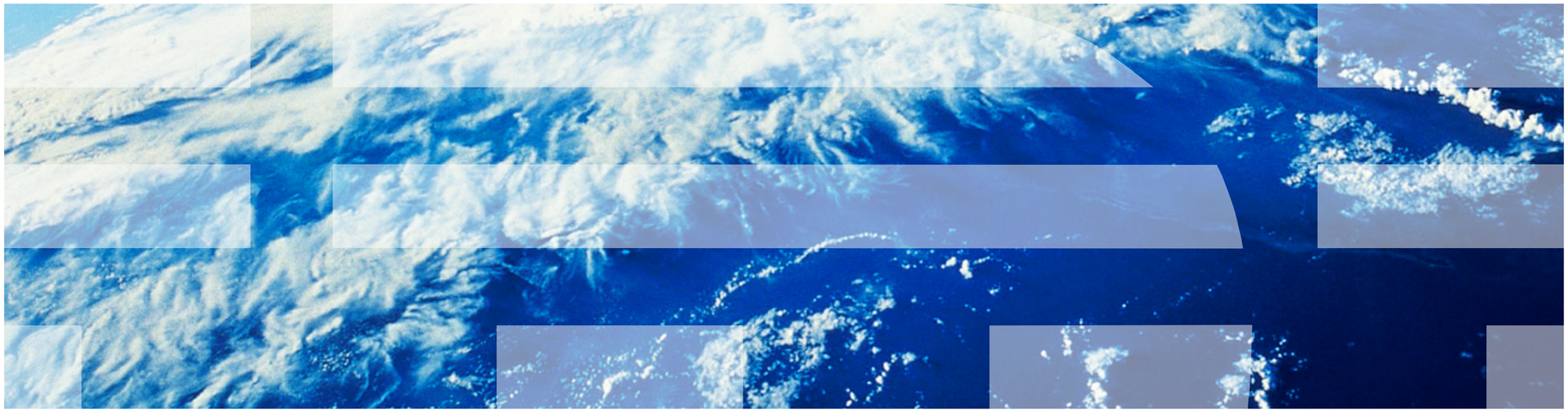
Lecture 2



Lecture 1: summary

Lecture 1: summary

- Class is about **computer systems** for data science

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases
- Performance concepts and rules of thumbs

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases
- Performance concepts and rules of thumbs
  - Throughput and latency: two orthogonal metrics to evaluate computer systems (and pizzerias)

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases
- Performance concepts and rules of thumbs
  - Throughput and latency: two orthogonal metrics to evaluate computer systems (and pizzerias)
  - Amdahl's law:

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases
- Performance concepts and rules of thumbs
  - Throughput and latency: two orthogonal metrics to evaluate computer systems (and pizzerias)
  - Amdahl's law:
  - $$Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1-p) + \frac{p}{S}}$$

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases
- Performance concepts and rules of thumbs
  - Throughput and latency: two orthogonal metrics to evaluate computer systems (and pizzerias)
  - Amdahl's law:
  - $$Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1-p) + \frac{p}{S}}$$
  - Speedup bounded by: $\frac{1}{fraction\ of\ time\ not\ enhanced}$

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases
- Performance concepts and rules of thumbs
  - Throughput and latency: two orthogonal metrics to evaluate computer systems (and pizzerias)
  - Amdahl's law:
  - $$Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1-p) + \frac{p}{S}}$$
  - Speedup bounded by: $\frac{1}{fraction\ of\ time\ not\ enhanced}$
  - Time scales in computer systems can vary by millions

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases
- Performance concepts and rules of thumbs
  - Throughput and latency: two orthogonal metrics to evaluate computer systems (and pizzerias)
  - Amdahl's law:
  - $$Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1-p) + \frac{p}{S}}$$
  - Speedup bounded by: $\frac{1}{fraction\ of\ time\ not\ enhanced}$
  - Time scales in computer systems can vary by millions
    - CPUs and memory operate in nanoseconds

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases
- Performance concepts and rules of thumbs
  - Throughput and latency: two orthogonal metrics to evaluate computer systems (and pizzerias)
  - Amdahl's law:
  - $$Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1-p) + \frac{p}{S}}$$
  - Speedup bounded by: $\frac{1}{fraction\ of\ time\ not\ enhanced}$
  - Time scales in computer systems can vary by millions
    - CPUs and memory operate in nanoseconds
    - Datacenter networks and SSDs operates in microseconds

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases
- Performance concepts and rules of thumbs
  - Throughput and latency: two orthogonal metrics to evaluate computer systems (and pizzerias)
  - Amdahl's law:
  - $$Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1-p) + \frac{p}{S}}$$
  - Speedup bounded by: $\frac{1}{fraction\ of\ time\ not\ enhanced}$
  - Time scales in computer systems can vary by millions
    - CPUs and memory operate in nanoseconds
    - Datacenter networks and SSDs operates in microseconds
    - Sending stuff over the Internet operates in milliseconds

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases
- Performance concepts and rules of thumbs
  - Throughput and latency: two orthogonal metrics to evaluate computer systems (and pizzerias)
  - Amdahl's law:
  - $$Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1-p) + \frac{p}{S}}$$
  - Speedup bounded by: $\frac{1}{fraction\ of\ time\ not\ enhanced}$
  - Time scales in computer systems can vary by millions
    - CPUs and memory operate in nanoseconds
    - Datacenter networks and SSDs operates in microseconds
    - Sending stuff over the Internet operates in milliseconds
- Intro to datacenters

Lecture 1: summary

- Class is about **computer systems** for data science
  - Very little math or algorithms
  - Broad overview of computer systems and databases
- Performance concepts and rules of thumbs
  - Throughput and latency: two orthogonal metrics to evaluate computer systems (and pizzerias)
  - Amdahl's law:
  - $$Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1-p) + \frac{p}{S}}$$
  - Speedup bounded by: $\frac{1}{fraction\ of\ time\ not\ enhanced}$
  - Time scales in computer systems can vary by millions
    - CPUs and memory operate in nanoseconds
    - Datacenter networks and SSDs operates in microseconds
    - Sending stuff over the Internet operates in milliseconds
- Intro to datacenters
  - Modern datacenter design: standard hardware, replicated in racks (cabinets), rows, deployed in football stadium-sized warehouses

Lecture 2: What are we covering today?

Lecture 2: What are we covering today?

- Wrapping up data centers
  - Power/cooling
  - Networking
  - AI datacenters

Lecture 2: What are we covering today?

- Wrapping up data centers
  - Power/cooling
  - Networking
  - AI datacenters
- Single table SQL
  - Relational model
  - Schemas
  - Data types
  - Limits
  - Basic queries

Lecture 2: What are we covering today?

- Wrapping up data centers
  - Power/cooling
  - Networking
  - AI datacenters
- Single table SQL
  - Relational model
  - Schemas
  - Data types
  - Limits
  - Basic queries
- Schema evolution
  - API schema
  - Case studies [1, 2]

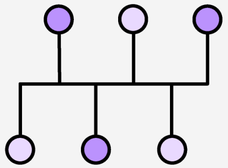
Row/Cluster

- 30+ racks

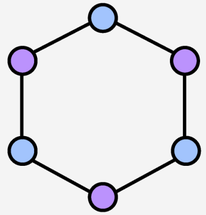
Network Topology

Network Topology

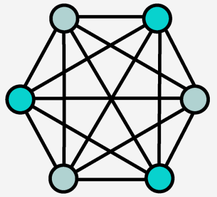
Bus



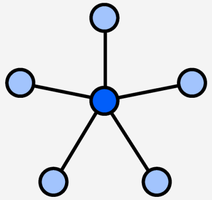
Ring



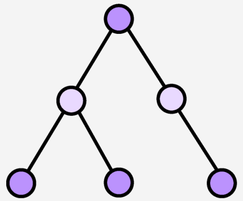
Mesh



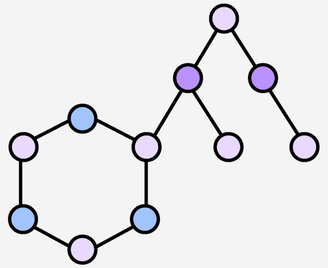
Star



Tree



Hybrid



Networking - Switch locations

Networking - Switch locations

- Top-of-rack switch
  - Connecting machines in rack
  - Multiple links going to end-of-row routers

Networking - Switch locations

- Top-of-rack switch
  - Connecting machines in rack
  - Multiple links going to end-of-row routers

Networking - Switch locations

- Top-of-rack switch
  - Connecting machines in rack
  - Multiple links going to end-of-row routers
- End-of-row router
  - Aggregate row of machines
  - Multiple links going to core routers

Networking - Switch locations

- Top-of-rack switch
  - Connecting machines in rack
  - Multiple links going to end-of-row routers
- End-of-row router
  - Aggregate row of machines
  - Multiple links going to core routers

Networking - Switch locations

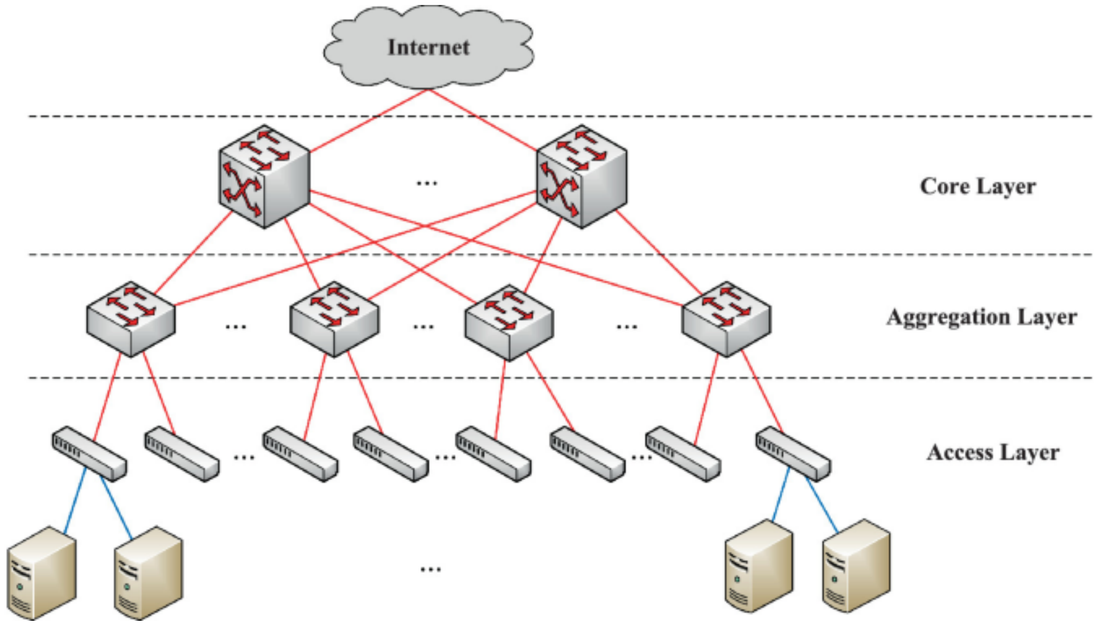
- Top-of-rack switch
  - Connecting machines in rack
  - Multiple links going to end-of-row routers
- End-of-row router
  - Aggregate row of machines
  - Multiple links going to core routers
- Core router
  - Multiple core routers

Networking - Switch locations

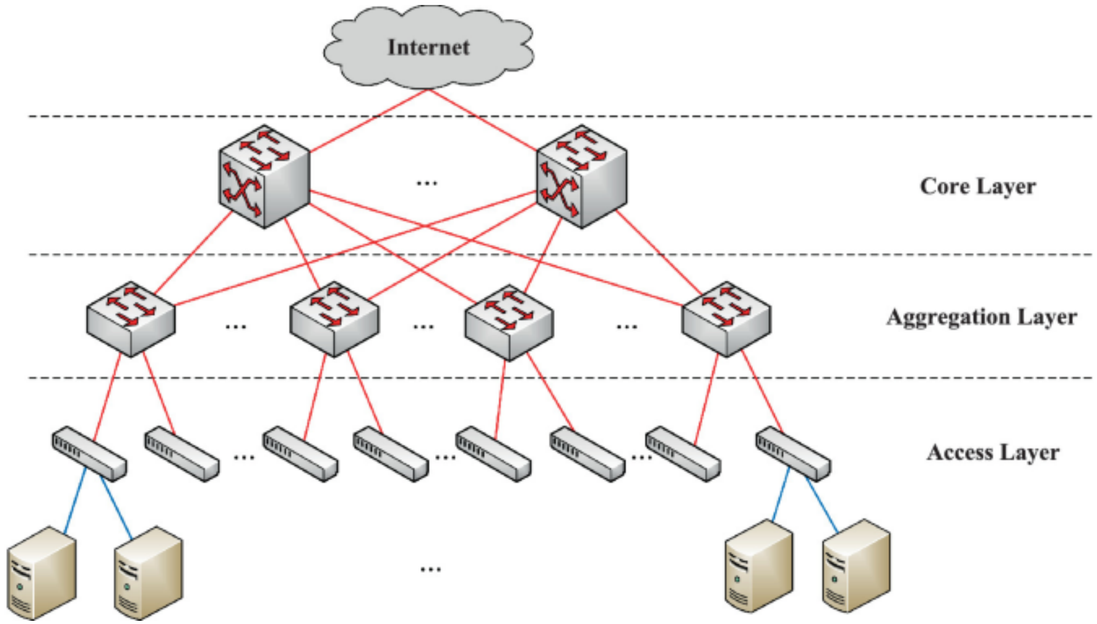
- Top-of-rack switch
  - Connecting machines in rack
  - Multiple links going to end-of-row routers
- End-of-row router
  - Aggregate row of machines
  - Multiple links going to core routers
- Core router
  - Multiple core routers
- Each of these have different latencies, throughput
  - Higher in hierarchy -> higher throughput

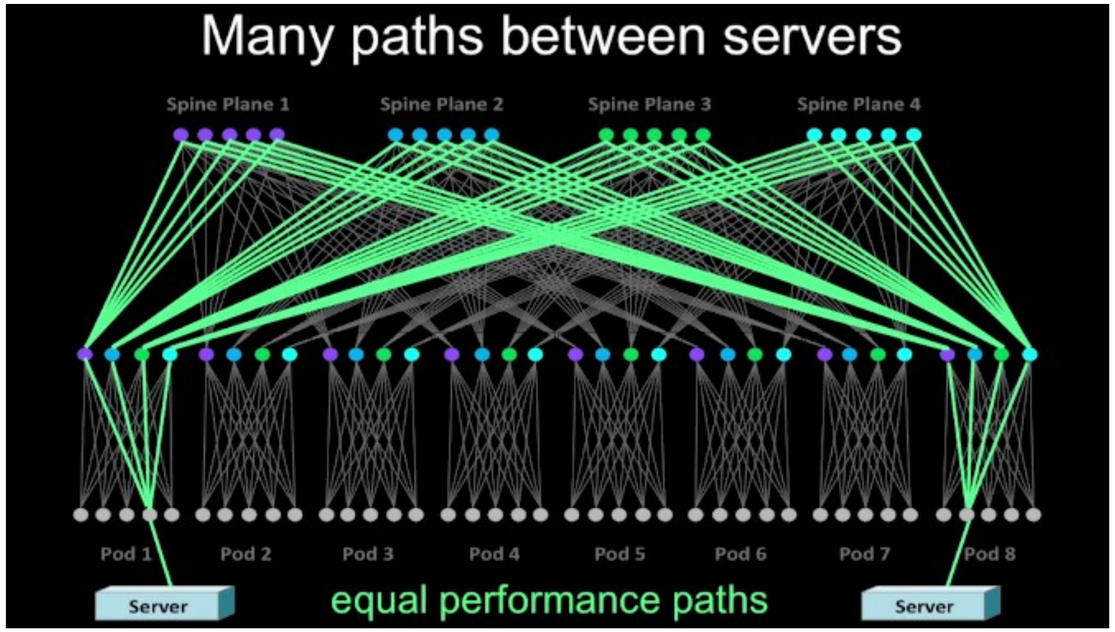
Multipath routing

Multipath routing



Multipath routing





Power Usage Effectiveness (PUE)

- Early data centers built with off-the-shelf components
  - Standard servers
  - HVAC unit designs from malls

$$\text{PUE ratio} = \frac{\text{Total Facility Power}}{\text{Server/Network Power}}$$

Inefficient: early data centers had PUE of 1.7-2.0

Power Usage Effectiveness (PUE)

- Early data centers built with off-the-shelf components
  - Standard servers
  - HVAC unit designs from malls

$$\text{PUE ratio} = \frac{\text{Total Facility Power}}{\text{Server/Network Power}}$$

Inefficient: early data centers had PUE of 1.7-2.0

- Average PUE for AWS datacenters in 2024: 1.15 (only 15% from optimal!)

Power Usage Effectiveness (PUE)

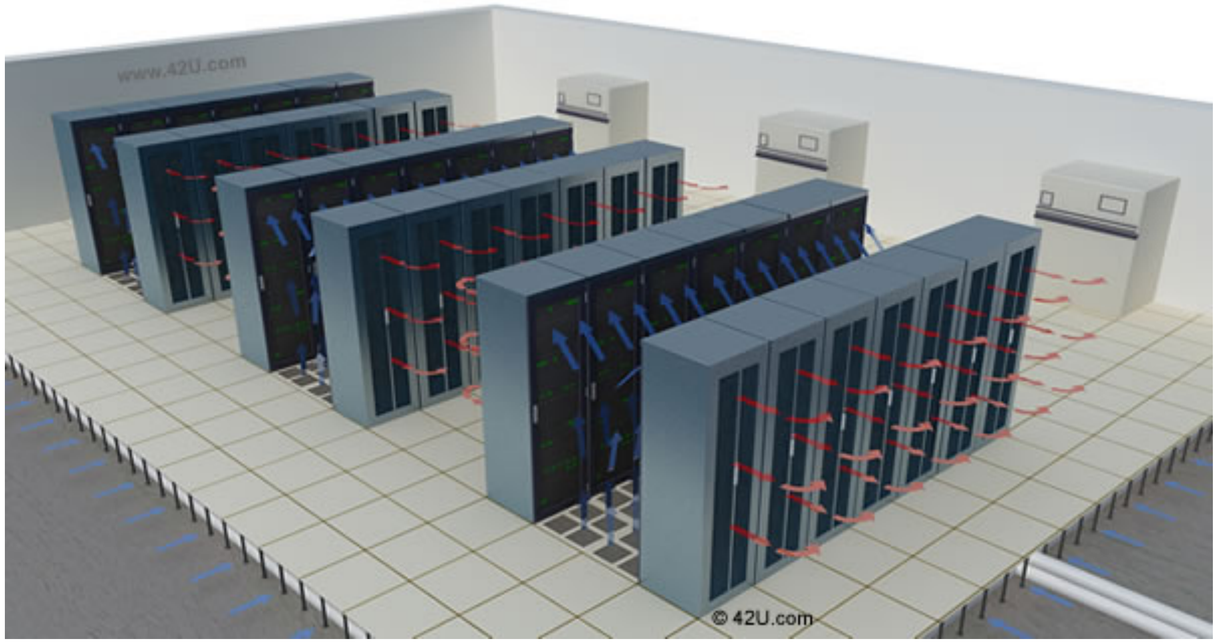
- Early data centers built with off-the-shelf components
  - Standard servers
  - HVAC unit designs from malls

  $$\text{PUE ratio} = \frac{\text{Total Facility Power}}{\text{Server/Network Power}}$$

  Inefficient: early data centers had PUE of 1.7-2.0
- Average PUE for AWS datacenters in 2024: 1.15 (only 15% from optimal!)
- Some can be 1.04
- Power is about 25% of monthly operating cost
  - And is (one of) the limiting factor in how large the datacenter can be

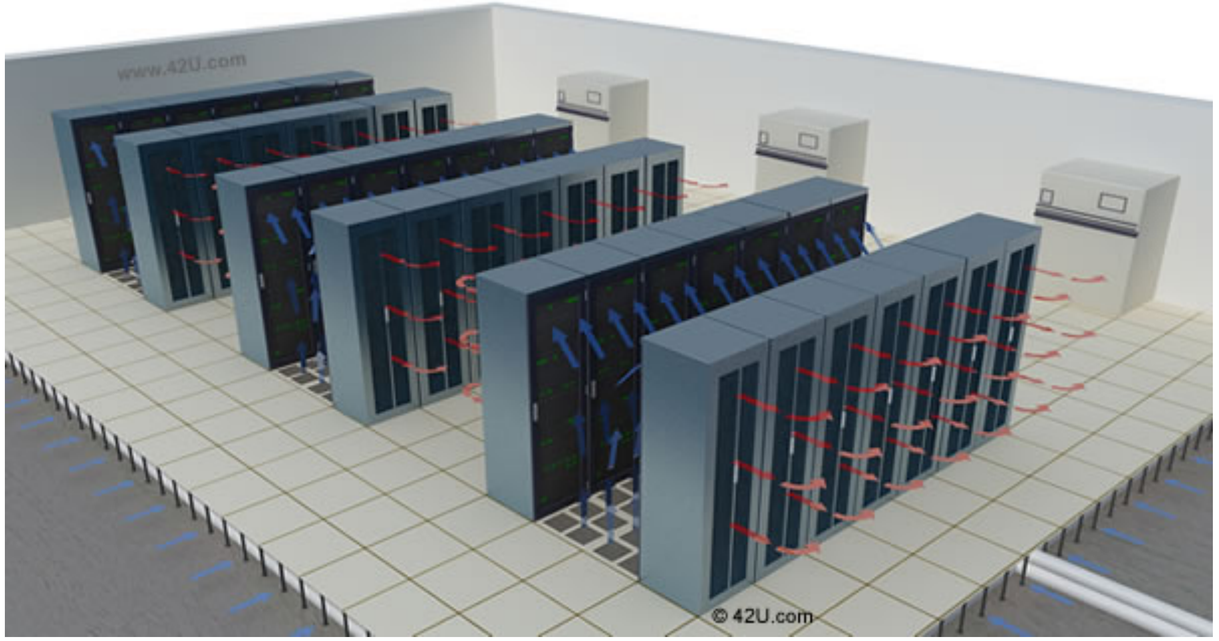
Energy Efficient Data Centers

- Better power distribution - Fewer transformers



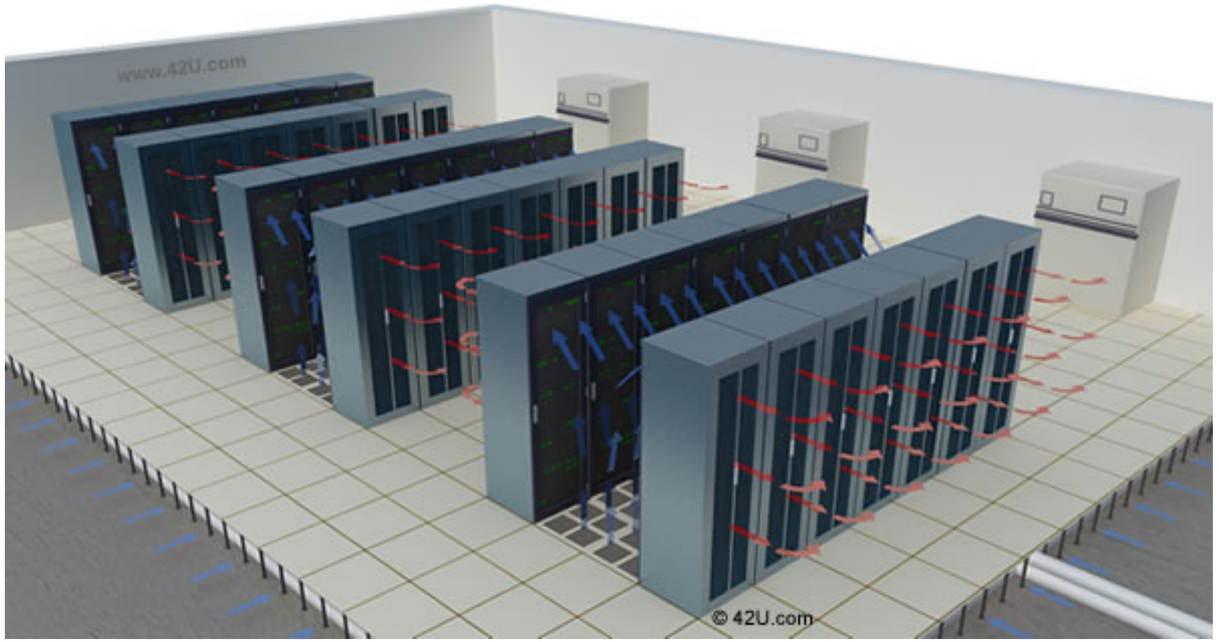
Energy Efficient Data Centers

- Better power distribution - Fewer transformers
- Better cooling - use environment (air/water) rather than air conditioning
  - Bring in outside air
  - Evaporate some water



Energy Efficient Data Centers

- Better power distribution - Fewer transformers
- Better cooling - use environment (air/water) rather than air conditioning
  - Bring in outside air
  - Evaporate some water
- IT Equipment range
  - OK up to +115°F



Liquid immersion is the “hottest” new technology for cooling data centers

Liquid immersion is the “hottest” new technology for cooling data centers

Liquid immersion is the “hottest” new technology for cooling data centers

Backup Power

Backup Power

- Massive amount of batteries to tolerate short glitches in power

Backup Power

- Massive amount of batteries to tolerate short glitches in power
  - Just need long enough for backup generators to startup

Backup Power

- Massive amount of batteries to tolerate short glitches in power
  - Just need long enough for backup generators to startup
- How do glitches occur?

Backup Power

- Massive amount of batteries to tolerate short glitches in power
  - Just need long enough for backup generators to startup
- How do glitches occur?
  - Thunder, earthquake, power loss from power company, cyber attack, …

Backup Power

- Massive amount of batteries to tolerate short glitches in power
  - Just need long enough for backup generators to startup
- How do glitches occur?
  - Thunder, earthquake, power loss from power company, cyber attack, …
- Massive collections of backup generators

Backup Power

- Massive amount of batteries to tolerate short glitches in power
  - Just need long enough for backup generators to startup
- How do glitches occur?
  - Thunder, earthquake, power loss from power company, cyber attack, …
- Massive collections of backup generators
- Huge fuel tanks to provide fuel for the generators

Backup Power

- Massive amount of batteries to tolerate short glitches in power
  - Just need long enough for backup generators to startup
- How do glitches occur?
  - Thunder, earthquake, power loss from power company, cyber attack, …
- Massive collections of backup generators
- Huge fuel tanks to provide fuel for the generators
- Fuel replenishment transportation network (e.g. fuel trucks)

Backup Power

- Massive amount of batteries to tolerate short glitches in power
  - Just need long enough for backup generators to startup
- How do glitches occur?
  - Thunder, earthquake, power
- Massive collections of backu
- Huge fuel tanks to provide f
- Fuel replenishment transpo

CNN World Africa Americas Asia Australia China Europe More Watch Subscribe Sign in

START THE DAY HERE

New details in fatal shooting of Alex Pretti. Rep. Ilhan Omar attacked during town hall. Social media platforms on trial

WORLD > ASIA • 2 MIN READ

Explosive battery blaze in South Korea ‘paralyzes’ vital government services

SEP 27, 2025

By Laura Sharman

Energy sources

Energy sources

- Increasingly, data centers powered by renewable energy

Energy sources

- Increasingly, data centers powered by renewable energy
  - But, solar/wind are intermittent

Energy sources

- Increasingly, data centers powered by renewable energy
  - But, solar/wind are intermittent
  - Hydro, nuclear are more reliable

Energy sources

- Increasingly, data centers powered by renewable energy
  - But, solar/wind are intermittent
  - Hydro, nuclear are more reliable

Energy sources

- Increasingly, data centers powered by renewable energy
  - But, solar/wind are intermittent
  - Hydro, nuclear are more reliable
- In practice, many new data centers powered by solar / wind but rely on fossil fuels from the electric grid when the wind isn’t blowing / sun isn’t shining

Energy sources

- Increasingly, data centers powered by renewable energy
  - But, solar/wind are intermittent
  - Hydro, nuclear are more reliable
- In practice, many new data centers powered by solar / wind but rely on fossil fuels from the electric grid when the wind isn’t blowing / sun isn’t shining

Fault Tolerance

Fault Tolerance

- At the scale of new data centers, things are breaking constantly

Fault Tolerance

- At the scale of new data centers, things are breaking constantly
- Every aspect of the data center must be able to tolerate failures

Fault Tolerance

- At the scale of new data centers, things are breaking constantly
- Every aspect of the data center must be able to tolerate failures
- Solution: Redundancy

Fault Tolerance

- At the scale of new data centers, things are breaking constantly
- Every aspect of the data center must be able to tolerate failures
- Solution: Redundancy
  - Multiple independent copies of all data

Fault Tolerance

- At the scale of new data centers, things are breaking constantly
- Every aspect of the data center must be able to tolerate failures
- Solution: Redundancy
  - Multiple independent copies of all data
    - Independent? Consistency :(

Fault Tolerance

- At the scale of new data centers, things are breaking constantly
- Every aspect of the data center must be able to tolerate failures
- Solution: Redundancy
  - Multiple independent copies of all data
    - Independent? Consistency :(

Fault Tolerance

- At the scale of new data centers, things are breaking constantly
- Every aspect of the data center must be able to tolerate failures
- Solution: Redundancy
  - Multiple independent copies of all data
    - Independent? Consistency :(
  - Multiple independent network connections

Fault Tolerance

- At the scale of new data centers, things are breaking constantly
- Every aspect of the data center must be able to tolerate failures
- Solution: Redundancy
  - Multiple independent copies of all data
    - Independent? Consistency :(
  - Multiple independent network connections
    - Under utilized capacity

Fault Tolerance

- At the scale of new data centers, things are breaking constantly
- Every aspect of the data center must be able to tolerate failures
- Solution: Redundancy
  - Multiple independent copies of all data
    - Independent? Consistency :(
  - Multiple independent network connections
    - Under utilized capacity

Fault Tolerance

- At the scale of new data centers, things are breaking constantly
- Every aspect of the data center must be able to tolerate failures
- Solution: Redundancy
  - Multiple independent copies of all data
    - Independent? Consistency :(
  - Multiple independent network connections
    - Under utilized capacity
  - Multiple copies of every service

Fault Tolerance

- At the scale of new data centers, things are breaking constantly
- Every aspect of the data center must be able to tolerate failures
- Solution: Redundancy
  - Multiple independent copies of all data
    - Independent? Consistency :(
  - Multiple independent network connections
    - Under utilized capacity
  - Multiple copies of every service
    - Releases, update cycles, resource use

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

It’s always DNS (unless it’s BGP)

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

It’s always DNS (unless it’s BGP)

~3 router failures (have to immediately pull traffic for an hour)

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

It's always DNS (unless it's BGP)

~3 router failures (have to immediately pull traffic for an hour)

~12 router reloads (takes out DNS and external VIPs for a couple minutes)

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

It's always DNS (unless it's BGP)

~3 router failures (have to immediately pull traffic for an hour)

~12 router reloads (takes out DNS and external VIPs for a couple minutes)

~8 network maintenances (might cause ~30-minute random connectivity losses)

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

It's always DNS (unless it's BGP)

~3 router failures (have to immediately pull traffic for an hour)

~12 router reloads (takes out DNS and external VIPs for a couple minutes)

~8 network maintenances (might cause ~30-minute random connectivity losses)

~5 racks go wonky (40-80 machines see 50% packet loss)

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

It's always DNS (unless it's BGP)

~3 router failures (have to immediately pull traffic for an hour)

~12 router reloads (takes out DNS and external VIPs for a couple minutes)

~8 network maintenances (might cause ~30-minute random connectivity losses)

~5 racks go wonky (40-80 machines see 50% packet loss)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

It's always DNS (unless it's BGP)

~3 router failures (have to immediately pull traffic for an hour)

~12 router reloads (takes out DNS and external VIPs for a couple minutes)

~8 network maintenances (might cause ~30-minute random connectivity losses)

~5 racks go wonky (40-80 machines see 50% packet loss)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~1 network rewiring (rolling ~5% of machines down over 2-day span)

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

It's always DNS (unless it's BGP)

~3 router failures (have to immediately pull traffic for an hour)

~12 router reloads (takes out DNS and external VIPs for a couple minutes)

~8 network maintenances (might cause ~30-minute random connectivity losses)

~5 racks go wonky (40-80 machines see 50% packet loss)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

It's always DNS (unless it's BGP)

~3 router failures (have to immediately pull traffic for an hour)

~12 router reloads (takes out DNS and external VIPs for a couple minutes)

~8 network maintenances (might cause ~30-minute random connectivity losses)

~5 racks go wonky (40-80 machines see 50% packet loss)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)

~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

It's always DNS (unless it's BGP)

~3 router failures (have to immediately pull traffic for an hour)

~12 router reloads (takes out DNS and external VIPs for a couple minutes)

~8 network maintenances (might cause ~30-minute random connectivity losses)

~5 racks go wonky (40-80 machines see 50% packet loss)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)

~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips for DNS

It's always DNS (unless it's BGP)

~3 router failures (have to immediately pull traffic for an hour)

~12 router reloads (takes out DNS and external VIPs for a couple minutes)

~8 network maintenances (might cause ~30-minute random connectivity losses)

~5 racks go wonky (40-80 machines see 50% packet loss)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)

~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)

→ **Reliability must come from software!**

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips fo

It's always DNS (unless it's B

~3 router failures (have to immediate

~12 router reloads (takes out DNS a

~8 network maintenances (might cau

~5 racks go wonky (40-80 machines

~20 rack failures (40-80 machines ir

~1 network rewiring (rolling ~5% of r

~1 rack-move (plenty of warning, ~5

~1 PDU failure (~500-1000 machine

~0.5 overheating (power down most

→ **Reliability must come fror**

Cloudflare 1.1.1.1 incident on July 14, 2025

2025-07-15


Ash Pallarito


Joe Abley

8 min read

On 14 July 2025, Cloudflare made a change to our service topologies that caused an outage for 1.1.1.1 on the edge, resulting in downtime for 62 minutes for customers using the 1.1.1.1 public DNS Resolver as well as intermittent degradation of service for Gateway DNS.

Failures in first year for a new data center (Jeff Dean)

~thousands of hard drive failures

~1000 individual machine failures

~dozens of minor 30-second blips fo

It's always DNS (unless it's B

~3 router failures (have to immediate

~12 router reloads (takes out DNS a

~8 network maintenances (migh

~5 racks go wonky (40-80 mach

~20 rack failures (40-80 machine

~1 network rewiring (rolling ~5% of r

~1 rack-move (plenty of warning, ~5

~1 PDU failure (~500-1000 machine

~0.5 overheating (power down most

→ **Reliability must come fror**

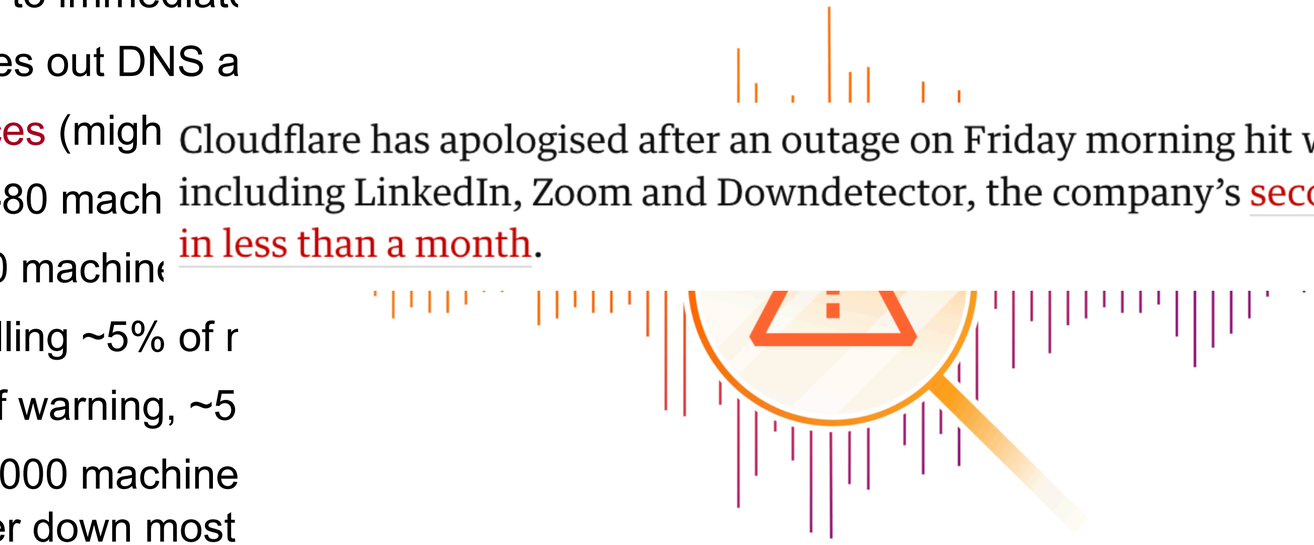
Cloudflare 1.1.1.1 incident on July 14, 2025

2025-07-15


Ash Pallarito


Joe Abley

8 min read



Cloudflare has apologised after an outage on Friday morning hit websites including LinkedIn, Zoom and Downdetector, the company's second outage in less than a month.

On 14 July 2025, Cloudflare made a change to our service topologies that caused an outage for 1.1.1.1 on the edge, resulting in downtime for 62 minutes for customers using the 1.1.1.1 public DNS Resolver as well as intermittent degradation of service for Gateway DNS.

Comparing AI datacenters to traditional ones

Comparing AI datacenters to traditional ones

- Similarities

Comparing AI datacenters to traditional ones

- Similarities
  - Same rack/row topology

Comparing AI datacenters to traditional ones

- Similarities
  - Same rack/row topology
  - Cooling still a big problem (e.g., GPU immersive cooling is coming soon)

Comparing AI datacenters to traditional ones

- Similarities
  - Same rack/row topology
  - Cooling still a big problem (e.g., GPU immersive cooling is coming soon)

Comparing AI datacenters to traditional ones

- Similarities
  - Same rack/row topology
  - Cooling still a big problem (e.g., GPU immersive cooling is coming soon)
- Differences

Comparing AI datacenters to traditional ones

- Similarities
  - Same rack/row topology
  - Cooling still a big problem (e.g., GPU immersive cooling is coming soon)
- Differences
  - Compute: Thousands of GPUs, small ratio of CPU/GPU

Comparing AI datacenters to traditional ones

- Similarities
  - Same rack/row topology
  - Cooling still a big problem (e.g., GPU immersive cooling is coming soon)
- Differences
  - Compute: Thousands of GPUs, small ratio of CPU/GPU
  - Memory: Don’t need as much traditional CPU memory, require lots of on-GPU High Bandwidth Memory (HBM), which is much more expensive

Comparing AI datacenters to traditional ones

- Similarities
  - Same rack/row topology
  - Cooling still a big problem (e.g., GPU immersive cooling is coming soon)
- Differences
  - Compute: Thousands of GPUs, small ratio of CPU/GPU
  - Memory: Don’t need as much traditional CPU memory, require lots of on-GPU High Bandwidth Memory (HBM), which is much more expensive
  - Network: AI training has much more demanding networking requirements. Requires dedicated high-bandwidth networking both within a server (e.g., NVIDIA’s NVLINK) and across servers (e.g., Infiniband)

Comparing AI datacenters to traditional ones

- Similarities
  - Same rack/row topology
  - Cooling still a big problem (e.g., GPU immersive cooling is coming soon)
- Differences
  - Compute: Thousands of GPUs, small ratio of CPU/GPU
  - Memory: Don’t need as much traditional CPU memory, require lots of on-GPU High Bandwidth Memory (HBM), which is much more expensive
  - Network: AI training has much more demanding networking requirements. Requires dedicated high-bandwidth networking both within a server (e.g., NVIDIA’s NVLINK) and across servers (e.g., Infiniband)

Comparing AI datacenters to traditional ones

- Similarities
  - Same rack/row topology
  - Cooling still a big problem (e.g., GPU immersive cooling is coming soon)
- Differences
  - Compute: Thousands of GPUs, small ratio of CPU/GPU
  - Memory: Don’t need as much traditional CPU memory, require lots of on-GPU High Bandwidth Memory (HBM), which is much more expensive
  - Network: AI training has much more demanding networking requirements. Requires dedicated high-bandwidth networking both within a server (e.g., NVIDIA’s NVLINK) and across servers (e.g., Infiniband)
- We will cover these topics more deeply in the second half of the class

Where should you build your datacenter?

Where should you build your datacenter?

- Plentiful, inexpensive electricity
  - Examples - Oregon: Hydroelectric; Iowa: Wind
  - Increasingly: nuclear, thermal

Where should you build your datacenter?

- Plentiful, inexpensive electricity
  - Examples - Oregon: Hydroelectric; Iowa: Wind
  - Increasingly: nuclear, thermal
- Good network connections
  - Access to the Internet backbone

Where should you build your datacenter?

- Plentiful, inexpensive electricity
  - Examples - Oregon: Hydroelectric; Iowa: Wind
  - Increasingly: nuclear, thermal
- Good network connections
  - Access to the Internet backbone
- Inexpensive land

Where should you build your datacenter?

- Plentiful, inexpensive electricity
  - Examples - Oregon: Hydroelectric; Iowa: Wind
  - Increasingly: nuclear, thermal
- Good network connections
  - Access to the Internet backbone
- Inexpensive land
- Geographically near users
  - Speed of light latency
  - Country laws (e.g. Our citizen's data must be kept in our county.)

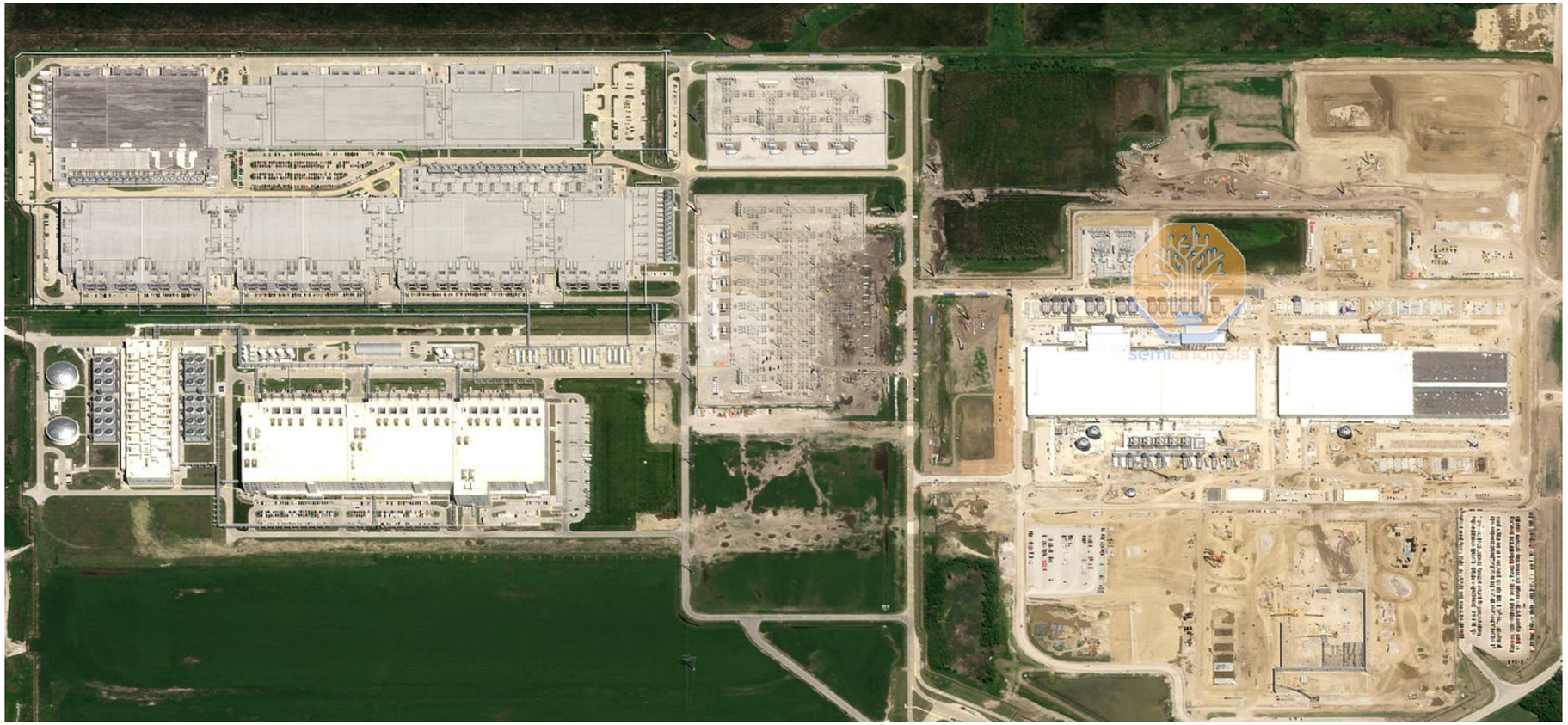
Where should you build your datacenter?

- Plentiful, inexpensive electricity
  - Examples - Oregon: Hydroelectric; Iowa: Wind
  - Increasingly: nuclear, thermal
- Good network connections
  - Access to the Internet backbone
- Inexpensive land
- Geographically near users
  - Speed of light latency
  - Country laws (e.g. Our citizen's data must be kept in our county.)
- Available labor pool

Where should you build your datacenter?

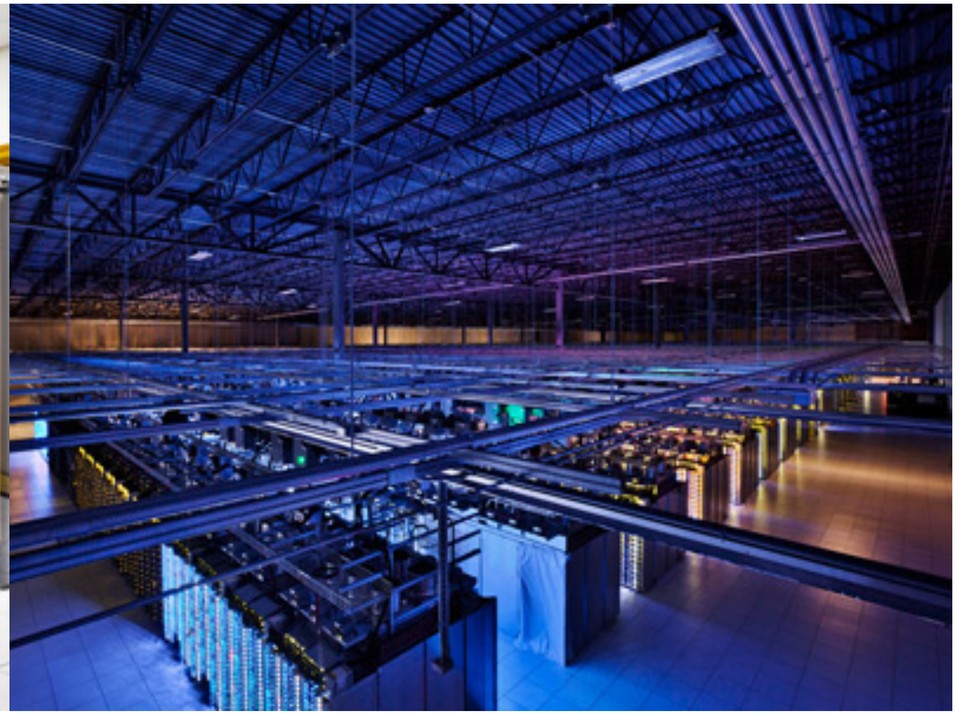
- Plentiful, inexpensive electricity
  - Examples - Oregon: Hydroelectric; Iowa: Wind
  - Increasingly: nuclear, thermal
- Good network connections
  - Access to the Internet backbone
- Inexpensive land
- Geographically near users
  - Speed of light latency
  - Country laws (e.g. Our citizen's data must be kept in our county.)
- Available labor pool
- Politics
  - Crime and corruption
  - Tax breaks
  - AI regulations

Google Data Center - Council Bluffs, Iowa, USA



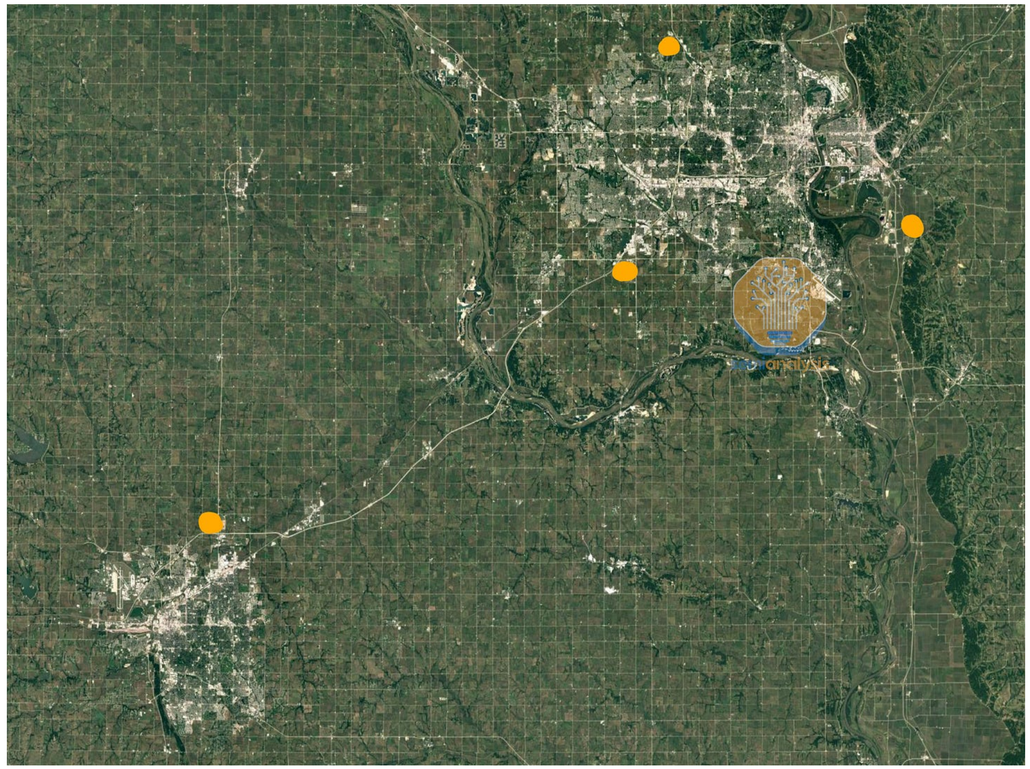
Source: semianalysis

Google data center pictures: Council Bluffs



Datacenter “megasites”

- Four Google datacenter sites within a 50-mile radius of each other, in the Iowa/Nebraska region
- May reach GW of total power consumption



Source: semianalysis

General application guidance

General application guidance

- A “responsible” data center tenant

General application guidance

- A “responsible” data center tenant
  - Organizes jobs and tasks into tiers

General application guidance

- A “responsible” data center tenant
  - Organizes jobs and tasks into tiers
  - Localizes and colocates

General application guidance

- A “responsible” data center tenant
  - Organizes jobs and tasks into tiers
  - Localizes and colocates
  - Minimizes bandwidth usage

General application guidance

- A “responsible” data center tenant
  - Organizes jobs and tasks into tiers
  - Localizes and colocates
  - Minimizes bandwidth usage
  - Uses the right models (pubsub, RPCs, batch workflows)

General application guidance

- A “responsible” data center tenant
  - Organizes jobs and tasks into tiers
  - Localizes and colocates
  - Minimizes bandwidth usage
  - Uses the right models (pubsub, RPCs, batch workflows)
  - Respects quotas

General application guidance

- A “responsible” data center tenant
  - Organizes jobs and tasks into tiers
  - Localizes and colocates
  - Minimizes bandwidth usage
  - Uses the right models (pubsub, RPCs, batch workflows)
  - Respects quotas
  - Provides telemetry for observation and adjustment

Summary

Summary

- It’s easy as data scientists (or software engineers) to lose sight that our code actually runs somewhere **physically**

Summary

- It’s easy as data scientists (or software engineers) to lose sight that our code actually runs somewhere **physically**
- The cloud is not some abstract concept: these are huge physical sites consuming power equivalent to entire cities

Summary

- It’s easy as data scientists (or software engineers) to lose sight that our code actually runs somewhere **physically**
- The cloud is not some abstract concept: these are huge physical sites consuming power equivalent to entire cities
- AI is accelerating the construction of new data centers

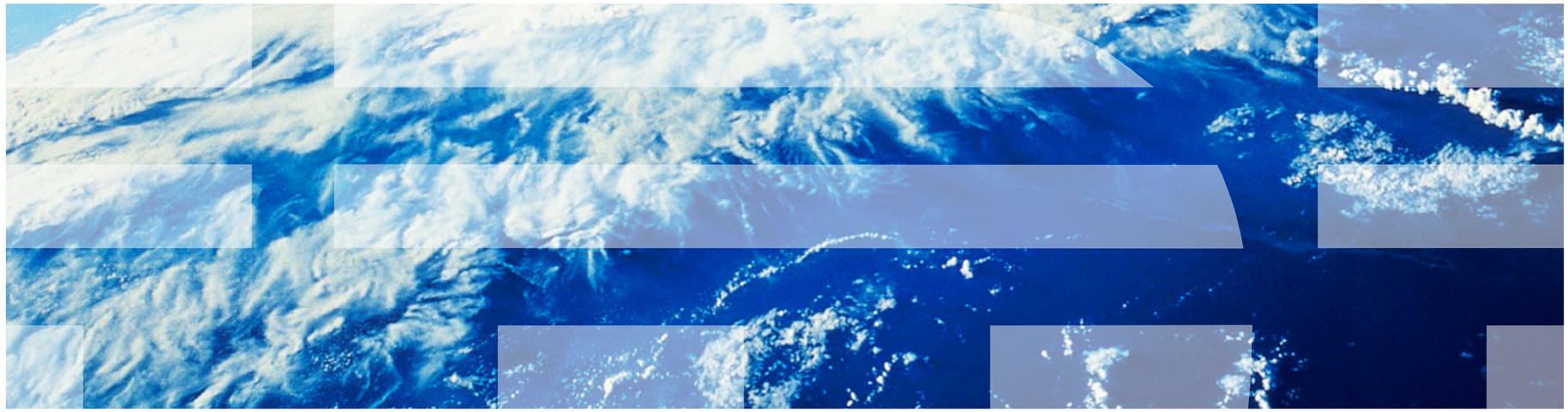
Summary

- It’s easy as data scientists (or software engineers) to lose sight that our code actually runs somewhere **physically**
- The cloud is not some abstract concept: these are huge physical sites consuming power equivalent to entire cities
- AI is accelerating the construction of new data centers
- Datacenter sustainability (especially in the age of AI) is going to be extremely important in the coming years

Borrowed from Shiva Shivakumar and Theodoros Rekatsinas

Computer Systems for Data Science
Topic 2

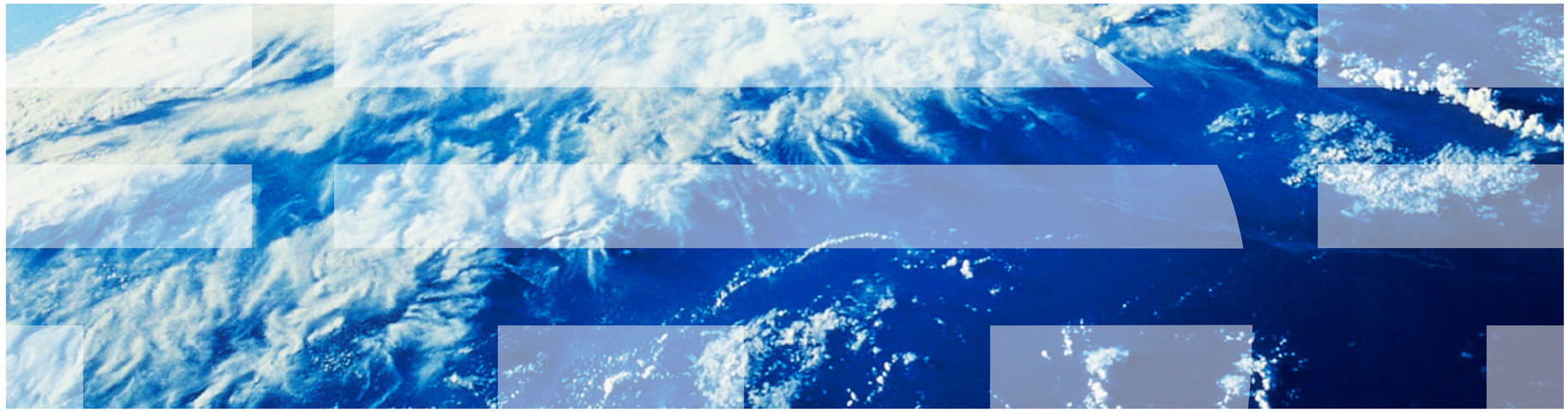
Relational Model and SQL



What we'll cover in this topic

- **Intuition**
  - Basic relational model
  - Map-filter-reduce concept
- **Intro to SQL**
  - Schemas, query structure of SELECT-FROM-WHERE, JOINs

Relational Model: Intuition



A Motivating Example

A basic Course Management System (CMS):

Entities or Relation (e.g., Students, Courses)

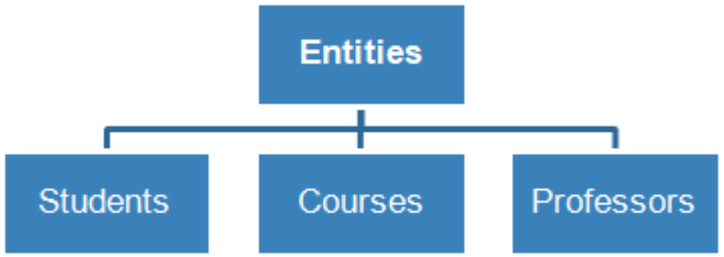
Relationships (e.g., Alice is enrolled in CSEE 4121)

A Motivating Example

A basic Course Management System (CMS):

Entities or Relation (e.g., Students, Courses)

Relationships (e.g., Alice is enrolled in CSEE 4121)

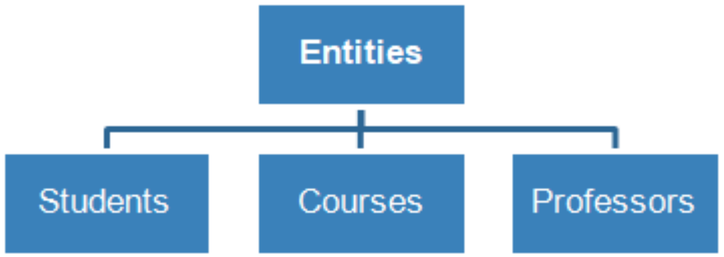


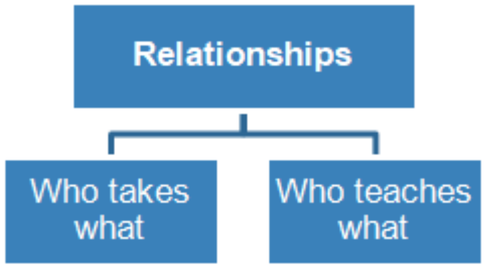
A Motivating Example

A basic Course Management System (CMS):

Entities or Relation (e.g., Students, Courses)

Relationships (e.g., Alice is enrolled in CSEE 4121)





Intuition: Spreadsheet Tables

Intuition: Spreadsheet Tables

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Intuition: Spreadsheet Tables

Logical Schema

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Intuition: Spreadsheet Tables

Logical Schema

Student(cuid: string, name: string, gpa: float)

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Intuition: Spreadsheet Tables

Logical Schema

Student(cuid: string, name: string, gpa: float)

Courses(cid: string, c-name: string, room: string)

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Intuition: Spreadsheet Tables

Logical Schema

Student(cuid: string, name: string, gpa: float)

Courses(cid: string, c-name: string, room: string)

Enrolled(cuid: string, cid: string, grade: string)

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Intuition: Spreadsheet Tables

Logical Schema

Student(cuid: string, name: string, gpa: float)

Courses(cid: string, c-name: string, room: string)

Enrolled(cuid: string, cid: string, grade: string)

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Intuition: Spreadsheet Tables

Logical Schema

Student(cuid: string, name: string, gpa: float)

Courses(cid: string, c-name: string, room: string)

Enrolled(cuid: string, cid: string, grade: string)

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Intuition: Spreadsheet Tables

Logical Schema

Student(cuid: string, name: string, gpa: float)

Courses(cid: string, c-name: string, room: string)

Enrolled(cuid: string, cid: string, grade: string)

Queries [“compute” over tables]

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Intuition: Spreadsheet Tables

Logical Schema

Student(cuid: string, name: string, gpa: float)

Courses(cid: string, c-name: string, room: string)

Enrolled(cuid: string, cid: string, grade: string)

Queries [“compute” over tables]

Alice’s GPA?

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Intuition: Spreadsheet Tables

Logical Schema

Student(cuid: string, name: string, gpa: float)

Courses(cid: string, c-name: string, room: string)

Enrolled(cuid: string, cid: string, grade: string)

Queries [“compute” over tables]

- Alice’s GPA?
- Jay’s classes?

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Intuition: Spreadsheet Tables

Logical Schema

Student(cuid: string, name: string, gpa: float)

Courses(cid: string, c-name: string, room: string)

Enrolled(cuid: string, cid: string, grade: string)

<u>Queries [“compute” over tables]</u>

- Alice’s GPA?
- Jay’s classes?
- AVG student GPA?

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Intuition: Spreadsheet Tables

Logical Schema

Student(cuid: string, name: string, gpa: float)

Courses(cid: string, c-name: string, room: string)

Enrolled(cuid: string, cid: string, grade: string)

<u>Queries [“compute” over tables]</u>

- Alice’s GPA?
- Jay’s classes?
- AVG student GPA?
- AVG student GPA in CSEE 4121?

Students				Enrolled		
CUID	**Name**	**GPA**		**CUID**	**CID**	**Grade**
as2121	Alice Smith	4.3		as2121	4121	A+
jg9999	Jay Goodwin	1.2		jg9999	4121	C
mc2312	Min Chang	2.2		mc2312	3292	A+
zb1111	Zorn Bjorn	3.8		zb1111	2999	D
Courses						
CID	**C-Name**	**Room**				
4121	Computer Sy	CEPSR				
3292	Databases	MUDD 1				
2999	Algorithms	MUDD 2				

Relational Model and Schemas

Relational Model and Schemas

- Definition: Data model

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations
- Relational model (aka tables)

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations
- Relational model (aka tables)
  - Simple and most popular

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations
- Relational model (aka tables)
  - Simple and most popular
  - Elegant algebra (E.F. Codd et al)

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations
- Relational model (aka tables)
  - Simple and most popular
  - Elegant algebra (E.F. Codd et al)

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations
- Relational model (aka tables)
  - Simple and most popular
  - Elegant algebra (E.F. Codd et al)

- Definition: Schema

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations
- Relational model (aka tables)
  - Simple and most popular
  - Elegant algebra (E.F. Codd et al)

- Definition: Schema
  - Describes blueprint of table(s)

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations
- Relational model (aka tables)
  - Simple and most popular
  - Elegant algebra (E.F. Codd et al)

- Definition: Schema
  - Describes blueprint of table(s)

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations
- Relational model (aka tables)
  - Simple and most popular
  - Elegant algebra (E.F. Codd et al)

- Definition: Schema
  - Describes blueprint of table(s)

- Every relation has a schema

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations
- Relational model (aka tables)
  - Simple and most popular
  - Elegant algebra (E.F. Codd et al)

- Definition: Schema
  - Describes blueprint of table(s)

- Every relation has a schema
  - Logical Schema: describes types, names

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations
- Relational model (aka tables)
  - Simple and most popular
  - Elegant algebra (E.F. Codd et al)

- Definition: Schema
  - Describes blueprint of table(s)

- Every relation has a schema
  - Logical Schema: describes types, names
  - Physical Schema: describes data layout

Relational Model and Schemas

- Definition: Data model
  - Organizing principle of data + operations
- Relational model (aka tables)
  - Simple and most popular
  - Elegant algebra (E.F. Codd et al)

- Definition: Schema
  - Describes blueprint of table(s)

- Every relation has a schema
  - Logical Schema: describes types, names
  - Physical Schema: describes data layout
  - Virtual Schema (Views): derived tables

Data Independence

Data Independence

1. Can we add a new column or attribute without rewriting the application?

Data Independence

1. Can we add a new column or attribute without rewriting the application?

Data Independence

1. Can we add a new column or attribute without rewriting the application?

   – **Logical Data Independence**

Data Independence

1. Can we add a new column or attribute without rewriting the application?

– **Logical Data Independence**
  - Protection from changes in the logical structure of the data

Data Independence

1. Can we add a new column or attribute without rewriting the application?

– **Logical Data Independence**
  - Protection from changes in the logical structure of the data

Data Independence

1. Can we add a new column or attribute without rewriting the application?

   – **Logical Data Independence**
   - Protection from changes in the logical structure of the data

2. Do you need to care which disks/machines are the data stored on?

Data Independence

1. Can we add a new column or attribute without rewriting the application?

   – **Logical Data Independence**
   - Protection from changes in the logical structure of the data

2. Do you need to care which disks/machines are the data stored on?

Data Independence

1. Can we add a new column or attribute without rewriting the application?

   – **Logical Data Independence**
   - Protection from changes in the logical structure of the data

2. Do you need to care which disks/machines are the data stored on?

   – **Physical Data Independence**

Data Independence

1. Can we add a new column or attribute without rewriting the application?

   – **Logical Data Independence**
   - Protection from changes in the logical structure of the data

2. Do you need to care which disks/machines are the data stored on?

   – **Physical Data Independence**
   - Protection from Physical Layout Changes

Python Operating on Lists

Python Operating on Lists

Basic types

- Int
- Long
- String

Python Operating on Lists

Basic types

- Int
- Long
- String

Map + Filter

- map(function, list)
- filter(function, list)

Map applies function to input list
Filter returns sub list that satisfies filter condition

Python Operating on Lists

Basic types
- Int
- Long
- String

Map + Filter
- map(function, list)
- filter(function, list)

Map applies function to input list
Filter returns sub list that satisfies filter condition

Reduce/Aggregate
- reduce(…)

Reduce runs a computation on a list and returns a result
E.g., SUM, AVG, MAX

SQL Queries on Tables (Lists of Rows)

SQL Queries on Tables (Lists of Rows)

Basic types

- Int32, Int64
- Char[n]
- Float32, Float64

SQL Queries on Tables (Lists of Rows)

Basic types

- Int32, Int64
- Char[n]
- Float32, Float64

Map + Filter

Single Table Query

```
SELECT c1, c2
FROM   T
WHERE condition;
```

SQL Queries on Tables (Lists of Rows)

Basic types

- Int32, Int64
- Char[n]
- Float32, Float64

Map + Filter

Single Table Query

```
SELECT c1, c2
FROM   T
WHERE condition;
```

Multi Table JOIN

```
SELECT c1, c2
FROM   T1, T2
WHERE condition;
```

SQL Queries on Tables (Lists of Rows)

Basic types

- Int32, Int64
- Char[n]
- Float32, Float64

Map + Filter

Single Table Query

```
SELECT c1, c2
FROM   T
WHERE condition;
```

Multi Table JOIN

```
SELECT c1, c2
FROM   T1, T2
WHERE condition;
```

Reduce/Aggregate

```
SELECT SUM(c1*c2)
FROM   T
WHERE condition
GROUP BY c3;
```

SQL Queries on Tables (Lists of Rows)

Basic types

- Int32, Int64
- Char[n]
- Float32, Float64

Map + Filter

Single Table Query

```
SELECT c1, c2
FROM   T
WHERE condition;
```

Multi Table JOIN

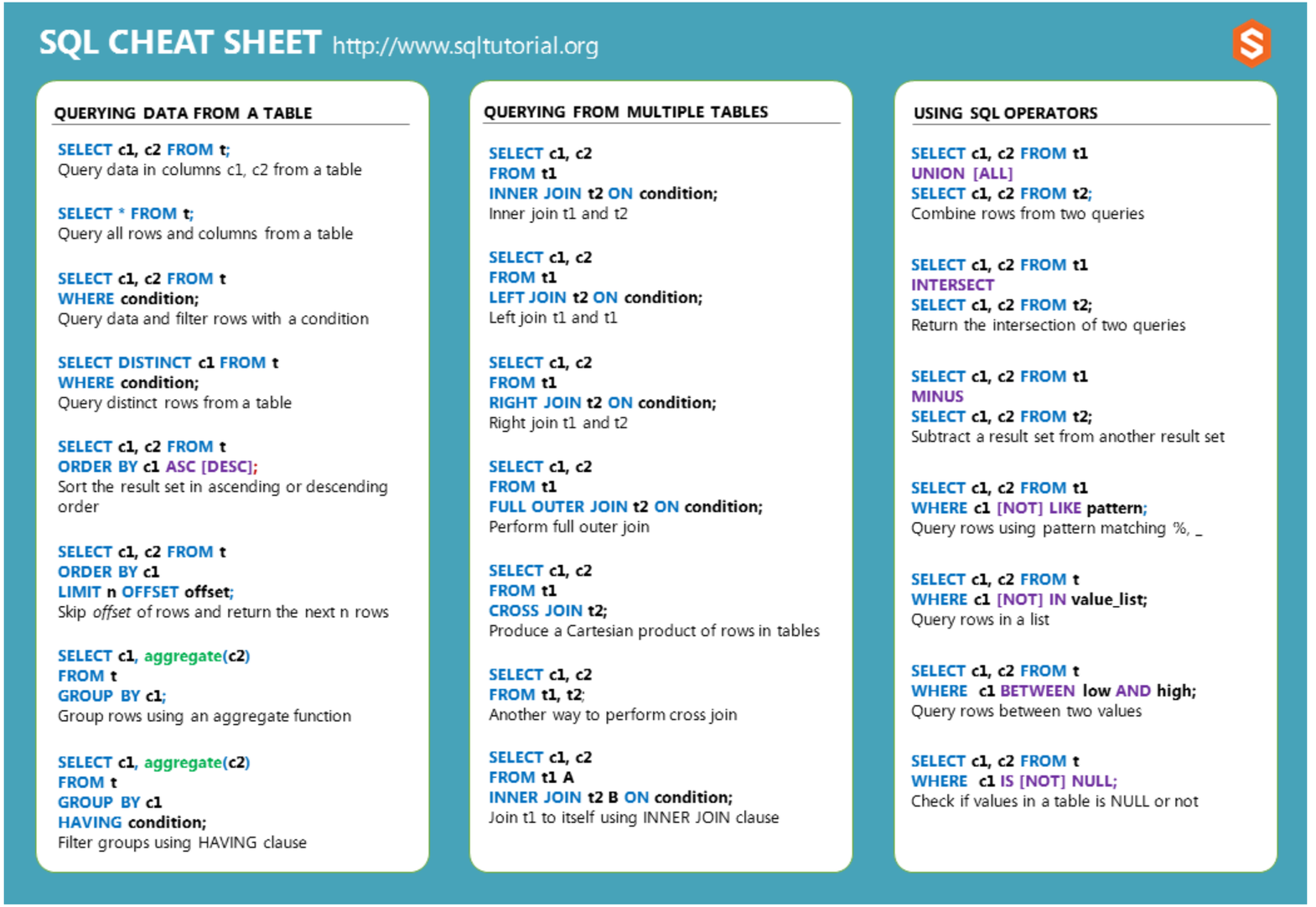
```
SELECT c1, c2
FROM   T1, T2
WHERE condition;
```

Reduce/Aggregate

```
SELECT SUM(c1*c2)
FROM   T
WHERE condition
GROUP BY c3;
```

Map-Filter-Reduce pattern: Same simple/powerful idea in MapReduce, Hadoop, Spark, etc.

SQL Cheat Sheet (www.sqltutorial.org/sql-cheat-sheet)



SQL CHEAT SHEET http://www.sqltutorial.org

QUERYING DATA FROM A TABLE

```
SELECT c1, c2 FROM t;
```
Query data in columns c1, c2 from a table

```
SELECT * FROM t;
```
Query all rows and columns from a table

```
SELECT c1, c2 FROM t
WHERE condition;
```
Query data and filter rows with a condition

```
SELECT DISTINCT c1 FROM t
WHERE condition;
```
Query distinct rows from a table

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC];
```
Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
```
Skip *offset* of rows and return the next n rows

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
```
Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
```
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

```
SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
```
Inner join t1 and t2

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
```
Left join t1 and t1

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
```
Right join t1 and t2

```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
```
Perform full outer join

```
SELECT c1, c2
FROM t1
CROSS JOIN t2;
```
Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1, t2;
```
Another way to perform cross join

```
SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;
```
Join t1 to itself using INNER JOIN clause

USING SQL OPERATORS

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2;
```
Combine rows from two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
```
Return the intersection of two queries

```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```
Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;
```
Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
```
Query rows in a list

```
SELECT c1, c2 FROM t
WHERE c1 BETWEEN low AND high;
```
Query rows between two values

```
SELECT c1, c2 FROM t
WHERE c1 IS [NOT] NULL;
```
Check if values in a table is NULL or not

Other data models

Other data models

Key-Value — The simplest model: every piece of data is stored as a key mapped to a value (which can be anything—string, JSON, binary blob). No schema, no relationships. Examples: Redis, DynamoDB, etcd. Great for caching, session storage, and simple lookups where you always know the key.

Document — Stores semi-structured documents (typically JSON or BSON) that can have nested fields and varying structures. Unlike relational tables, each document can have different fields. Examples: MongoDB, CouchDB, Firestore. Good for content management, user profiles, and applications where schema flexibility matters.

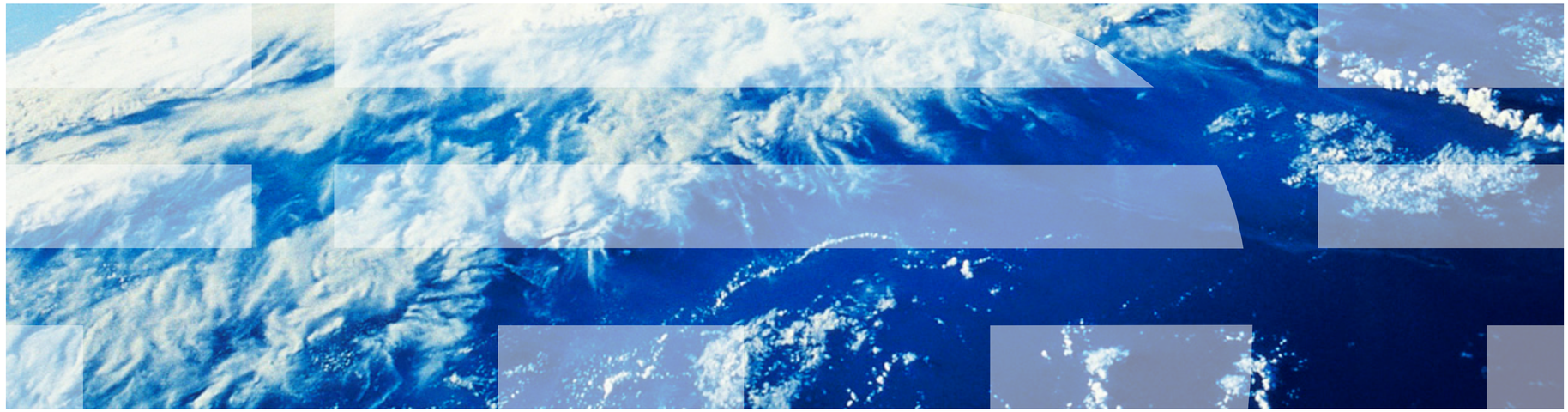
Wide-Column / Column-Family — Data is organized by columns rather than rows, with rows that can have different columns. Optimized for queries over large datasets where you typically read specific columns across many rows. Examples: Cassandra, HBase, Google Bigtable. Common for time-series data, analytics, and write-heavy workloads.

Graph — Models data as nodes (entities) and edges (relationships), with properties on both. Traversing relationships is a first-class operation rather than an expensive join. Examples: Neo4j, Amazon Neptune, TigerGraph. Ideal for social networks, fraud detection, and recommendation engines.

Time-Series — Optimized specifically for timestamped data points, with efficient compression and time-based queries. Examples: InfluxDB, TimescaleDB, Prometheus. Used for metrics, IoT sensor data, and monitoring.

Vector — Stores high-dimensional embeddings and supports similarity search (nearest neighbor queries). Examples: Pinecone, Milvus, pgvector. Essential for AI applications like semantic search and RAG systems.

Intro to SQL



SQL Introduction

- SQL is a standard language for querying and manipulating data

- SQL is a **very high-level** programming language
  - This works because it is optimized well!

SQL stands for
Structured
Query
Language

SQL is a…

- Data Manipulation Language (DML)
  - Query one or more tables
  - Insert/delete/modify tuples in tables

- Data Definition Language (DDL)
  - Define relational schemata
  - Create/alter/delete tables and their attributes

Basic Set Algebra Concepts

Basic Set Algebra Concepts

- List: [1, 1, 2, 3] Ordered, duplicates

Basic Set Algebra Concepts

- List: [1, 1, 2, 3] Ordered, duplicates
- Set: {2, 1, 3} Unordered, no duplicates

Basic Set Algebra Concepts

- List: [1, 1, 2, 3] Ordered, duplicates
- Set: {2, 1, 3} Unordered, no duplicates
- Multiset: {2, 1, 3, 1} Unordered, duplicates

Basic Set Algebra Concepts

- List: [1, 1, 2, 3] Ordered, duplicates
- Set: {2, 1, 3} Unordered, no duplicates
- Multiset: {2, 1, 3, 1} Unordered, duplicates

- Unions:
  - Set: {2, 1, 3} U {2, 3} = {2, 1, 3}
  - Multiset: {2, 1, 3} U {2, 3} = {2, 1, 3, 2, 3}

Basic Set Algebra Concepts

- List: [1, 1, 2, 3] Ordered, duplicates
- Set: {2, 1, 3} Unordered, no duplicates
- Multiset: {2, 1, 3, 1} Unordered, duplicates

- Unions:
  - Set: {2, 1, 3} U {2, 3} = {2, 1, 3}
  - Multiset: {2, 1, 3} U {2, 3} = {2, 1, 3, 2, 3}

Basic Set Algebra Concepts

- List: [1, 1, 2, 3] Ordered, duplicates
- Set: {2, 1, 3} Unordered, no duplicates
- Multiset: {2, 1, 3, 1} Unordered, duplicates

- Unions:
  - Set: {2, 1, 3} U {2, 3} = {2, 1, 3}
  - Multiset: {2, 1, 3} U {2, 3} = {2, 1, 3, 2, 3}

- Cross-product:
  - {1, 1, 2, 3} * {y, z} = {1, y}, {1, y}, {2, y}, {3, y}, {1, z}, {1, z}, {2, z}, {3, z}

Tables in SQL

Product

PName	Price	Manuf
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **relation** or **table** is a multiset of tuples/rows having the attributes specified by the schema

Tables in SQL

Product

PName	Price	Manuf
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

An **attribute** (or **column**) is a typed data entry present in each tuple in the relation

*Attributes must have an **atomic** type in standard SQL, i.e. not a list, set, etc.*

Tables in SQL

Product

PName	Price	Manuf
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **tuple** or **row** or **record** is a single entry in the table having the attributes specified by the schema

Tables in SQL

Product

PName	Price	Manuf
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

The number of tuples is the **cardinality** of the relation

The number of attributes is the **arity** of the relation

Data Types in SQL

Atomic types:

- Characters: CHAR(20), VARCHAR(50)
- Numbers: INT, BIGINT, SMALLINT, FLOAT
- Others: MONEY, DATETIME…

Every attribute must have an atomic type

Table Schemas

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

Table Schemas

The **schema** of a table is the table name, its attributes, and their types:

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

Table Schemas

The **schema** of a table is the table name, its attributes, and their types:

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

A **key** is an attribute whose values are unique; we underline a key

Table Schemas

The **schema** of a table is the table name, its attributes, and their types:

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

A **key** is an attribute whose values are unique; we underline a key

Product(<u>Pname</u>: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

Key constraints

Key constraints

A **key** is a **minimal subset of attributes** that acts as a unique identifier for tuples in a relation

- A key is an implicit constraint on which tuples can be in the relation
- i.e. if two tuples agree on the values of the key, then they must be the same tuple!

Key constraints

A **key** is a **minimal subset of attributes** that acts as a unique identifier for tuples in a relation

- A key is an implicit constraint on which tuples can be in the relation
- i.e. if two tuples agree on the values of the key, then they must be the same tuple!

Students(cuid:string, name:string, gpa: float)

Key constraints

A **key** is a **minimal subset of attributes** that acts as a unique identifier for tuples in a relation

- A key is an implicit constraint on which tuples can be in the relation
- i.e. if two tuples agree on the values of the key, then they must be the same tuple!

Students(cuid:string, name:string, gpa: float)

1. Which would you select as a key?

Key constraints

A **key** is a **minimal subset of attributes** that acts as a unique identifier for tuples in a relation

- A key is an implicit constraint on which tuples can be in the relation
- i.e. if two tuples agree on the values of the key, then they must be the same tuple!

Students(cuid:string, name:string, gpa: float)

1. Which would you select as a key?
2. Is a key always guaranteed to exist?

Declaring Schema

Students(cuid: *string,* name: *string,* gpa: *float*)

```
CREATE TABLE Students (
  cuid CHAR(20),
  name  VARCHAR(50),
  gpa  float,
  PRIMARY KEY (cuid),
)
```

NULL and NOT NULL

NULL and NOT NULL

- To say “does not have value,” we use NULL

Students(cuid:string, name:string, gpa: float)

NULL and NOT NULL

- To say “does not have value,” we use NULL

Students(cuid:string, name:string, gpa: float)

cuid	name	gpa
123	Alice	3.
143	Jim	NULL

Say, Jim just enrolled in his first class.

NULL and NOT NULL

- To say “does not have value,” we use NULL

Students(cuid:string, name:string, gpa: float)

cuid	name	gpa
123	Alice	3
143	Jim	NULL

Say, Jim just enrolled in his first class.

In SQL, we may constrain a column to be NOT NULL, e.g., “name” in this table

NULL and NOT NULL

- To say “does not have value,” we use NULL

Students(cuid:string, name:string, gpa: float)

cuid	name	gpa
123	Alice	3.
143	Jim	NULL

Say, Jim just enrolled in his first

In SQL, we may constrain a column to be NOT NULL, e.g., “name” in this table

General Constraints

General Constraints

- We can actually specify arbitrary assertions
  E.g. “*There cannot be 25 people in the DB class*”

- In practice, we don’t specify many such constraints. Why?
  Performance!

Usually we do something ugly (or avoid doing something convenient) for the sake of performance

Summary of Schema Information

- Schema and Constraints are how databases understand the semantics (meaning) of data

- SQL supports general constraints:

  Keys are most important

API Schema

- Translation layer between external clients and internal database schema
- Same logical entity might be represented differently
  - (e.g., database splits User across multiple tables, but API returns a single unified User object)
- API can hide sensitive database details or expose computed/ aggregated data not directly stored
- Database schema can change without breaking the API if the translation layer is updated accordingly

API Schema

- Translation layer between external clients and internal database schema
- Same logical entity might be represented differently
  - (e.g., database splits User across multiple tables, but API returns a single unified User object)
- API can hide sensitive database details or expose computed/ aggregated data not directly stored
- Database schema can change without breaking the API if the translation layer is updated accordingly

- Defines the structure of data in transit
  - request/response formats, endpoints, parameters
- Optimized for usability, clarity, and client needs
- External contract - publicly exposed interface that clients depend on
- Uses language-agnostic formats
  - (JSON, XML, Protocol Buffers) with type systems
- Changes must maintain backward compatibility or use versioning

Protocol Buffers

- API Schema definition language
- Provides libraries for convenience
- Comparable to JSON and XML
  - But better!
- Binary on the wire

Protocol Buffers

- API Schema definition language
- Provides libraries for convenience
- Comparable to JSON and XML
  - But better!
- Binary on the wire

```
edition = "2023";

package tutorial;

message Person {
  string name = 1;
  int32 id = 2;
  string email = 3;

  enum PhoneType {
    PHONE_TYPE_UNSPECIFIED = 0;
    PHONE_TYPE_MOBILE = 1;
    PHONE_TYPE_HOME = 2;
    PHONE_TYPE_WORK = 3;
  }

  message PhoneNumber {
    string number = 1;
    PhoneType type = 2;
  }

  repeated PhoneNumber phones = 4;
}

message AddressBook {
  repeated Person people = 1;
}
```

Uber Blog Explore EN Search

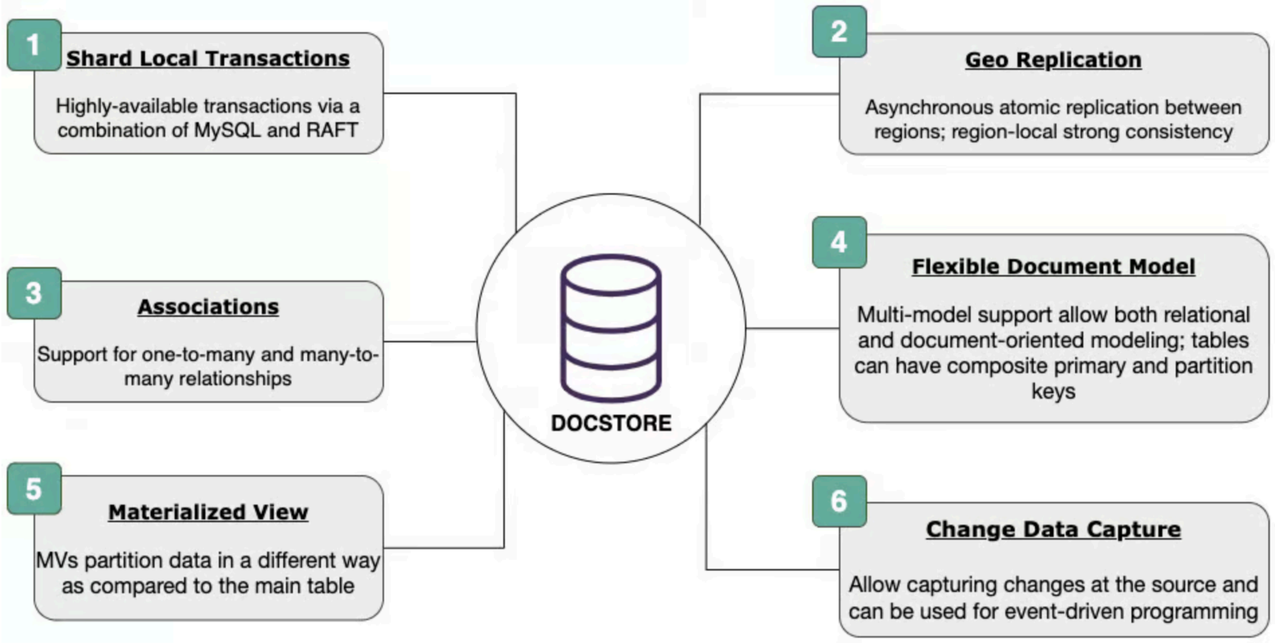
Engineering

Overview Backend Culture Data / ML Mobile Security Uber AI Web

Engineering

Evolving Schemaless into a Distributed SQL Database

February 23, 2021 / Global



- Flexible schema, not no schema!
- Centralizing the complexity of consistency
- Operational reality may not be “beautiful”

https://www.uber.com/blog/schemaless-sql-database/

Uber Blog

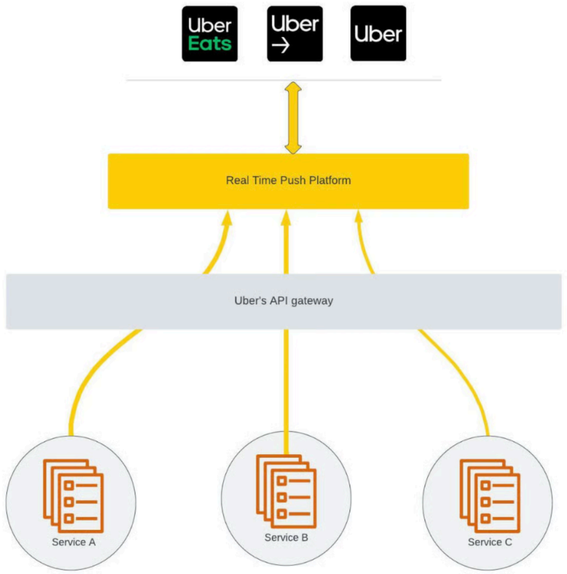
Sign up

Engineering

Engineering, Backend, Mobile

Uber's Next Gen Push Platform on gRPC

August 16, 2022 / Global



In our last blog post we talked about how we went from polling for refreshing the app to a push-based flow to build our app experience.

- Not request/response but streaming
- From REST-like to RPC-like
- Transport is as important as anything else
- Don't reinvent the wheel

https://www.uber.com/blog/ubers-next-gen-push-platform-on-grpc/