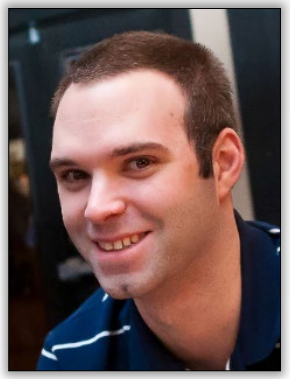
Relational or Not?

Choosing the Best Database for the Task

Bob Pusateri

sqlbits
2024

About Bob Pusateri



Microsoft CERTIFIED Master

SQL Server 2008

@bobp.bsky.social
@sqlbob
bob@bobpusateri.com
bobpusateri.com
bobpusateri

MVP Microsoft® Most Valuable Professional

Passions:

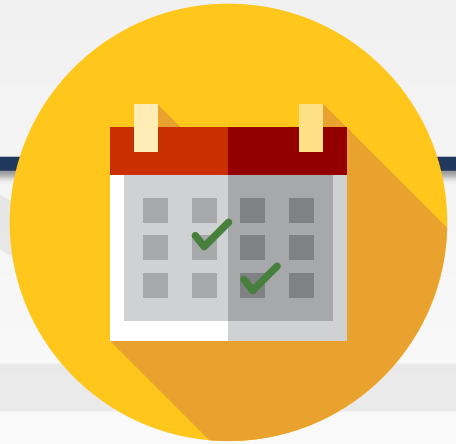
- Performance Tuning & Troubleshooting
- Very Large Databases
- Storage Engine Internals
- Big Data
- Cloud Architecture
- Teaching & Helping
- #BobFacts

ChiSQL

FRIEND OF redgate 2023

sqlbits ***2024***

Agenda



- Relational Databases
- Non-Relational Databases
- Pros and cons of each

Relational vs. Non-Relational

Relational

- Based on the Relational Model
- Data stored in tables
- “Normalized” data works best
- Database enforces schema & constraints
- When data violates relational rules (normal forms) bad things can happen

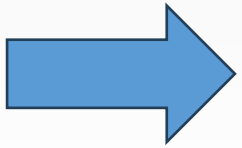
Non-Relational / NoSQL

- Based on *other* models
- Data stored in non-table structures
- No rigid schema present
- Constraints typically enforced by application
- Typically used with scale-out deployments

Normal Forms

- **First Normal Form (1NF):** Each table cell contains a single value

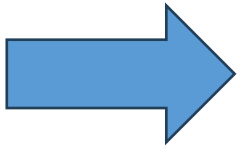
Class ID	Names
1	Amy, Brian, Cathy



Class ID	Name
1	Amy
1	Brian
1	Cathy

- **2NF:** Each non-key attribute is dependent on the primary key (PK)

Make	Model	Country
Audi	Q5	Germany
Audi	R8	Germany
Tesla	Y	USA
Land Rover	Discovery	UK



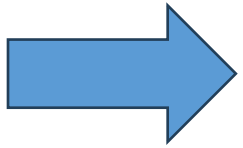
Make	Country
Audi	Germany
Tesla	USA
Land Rover	UK

Make	Model
Audi	Q5
Audi	R8
Tesla	Y
Land Rover	Discovery

Normal Forms

- **3NF:** Each column is directly related to PK, not to any other column

Award	Year	Winner	DOB
Best Actress	1939	Vivien Leigh	5 Nov 1913
Best Actress	1951	Vivien Leigh	5 Nov 1913
Best Actor	1944	Bing Crosby	3 May 1903
Best Actor	1956	Yul Brynner	11 Jul 1920



Award	Year	Winner
Best Actress	1939	Vivien Leigh
Best Actress	1951	Vivien Leigh
Best Actor	1944	Bing Crosby
Best Actor	1956	Yul Brynner

Winner	DOB
Vivien Leigh	5 Nov 1913
Bing Crosby	3 May 1903
Yul Brynner	11 Jul 1920

Normal Forms

- There are others
  - Boyce-Codd Normal Form (aka 3.5NF)
    - All redundancy based on “functional dependency” has been removed
  - 4NF
  - 5NF

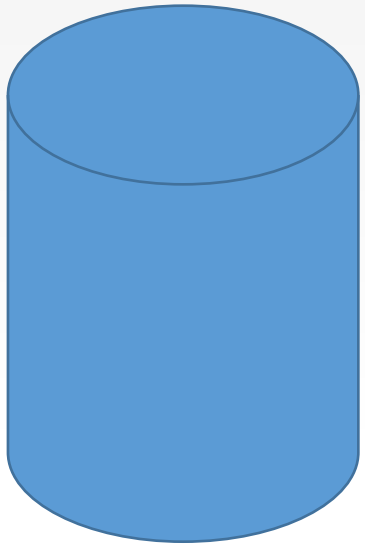
Thoughts on NoSQL

- Some say NoSQL == No Schema == No Design
  - Not True
- **GENERALLY** NoSQL schemas
  - Do Exist
  - Are somewhat enforced by the database
  - Are fully enforced by the application
    - And this has some nice advantages!
- There are still design decisions that need to happen early on
  - (And if they're wrong you will pay for them later)

3 DBAS WALKED INTO A NOSQL BAR….

A WHILE LATER THEY WALKED OUT BECAUSE THEY COULDN'T FIND A TABLE

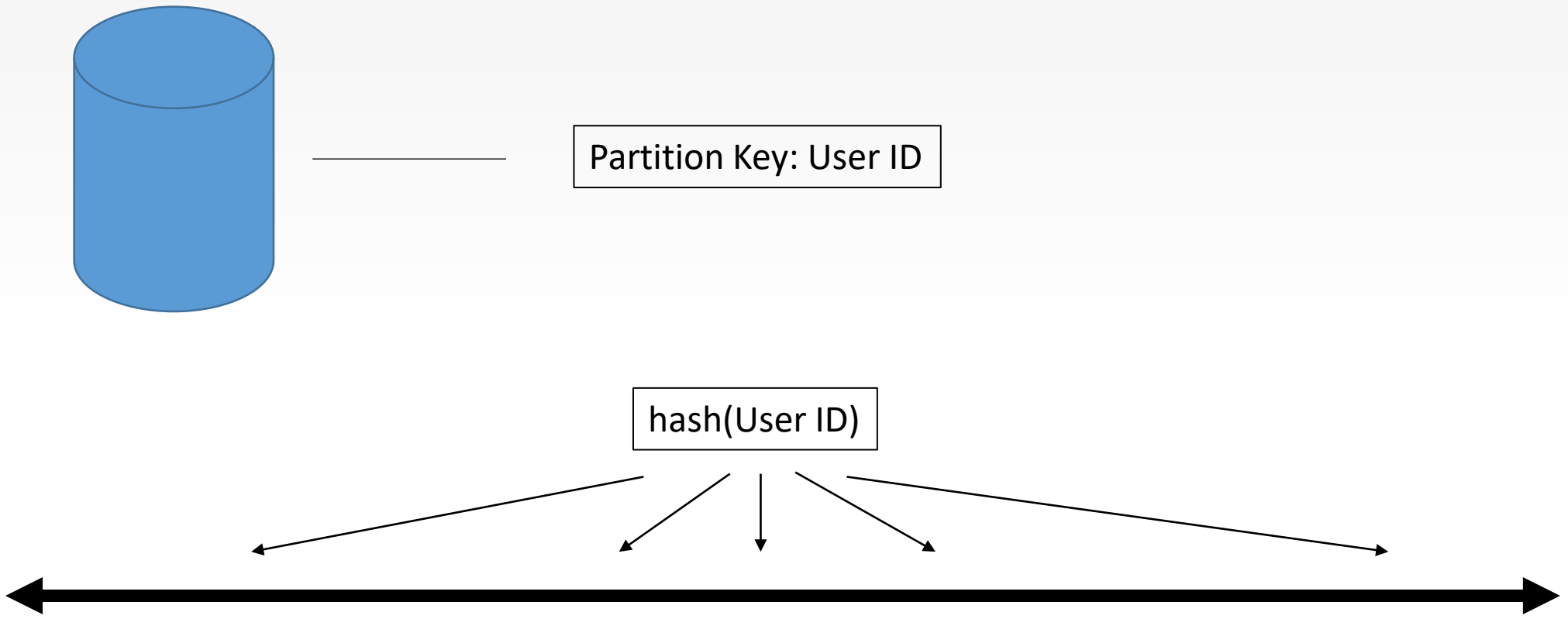
Partitioning



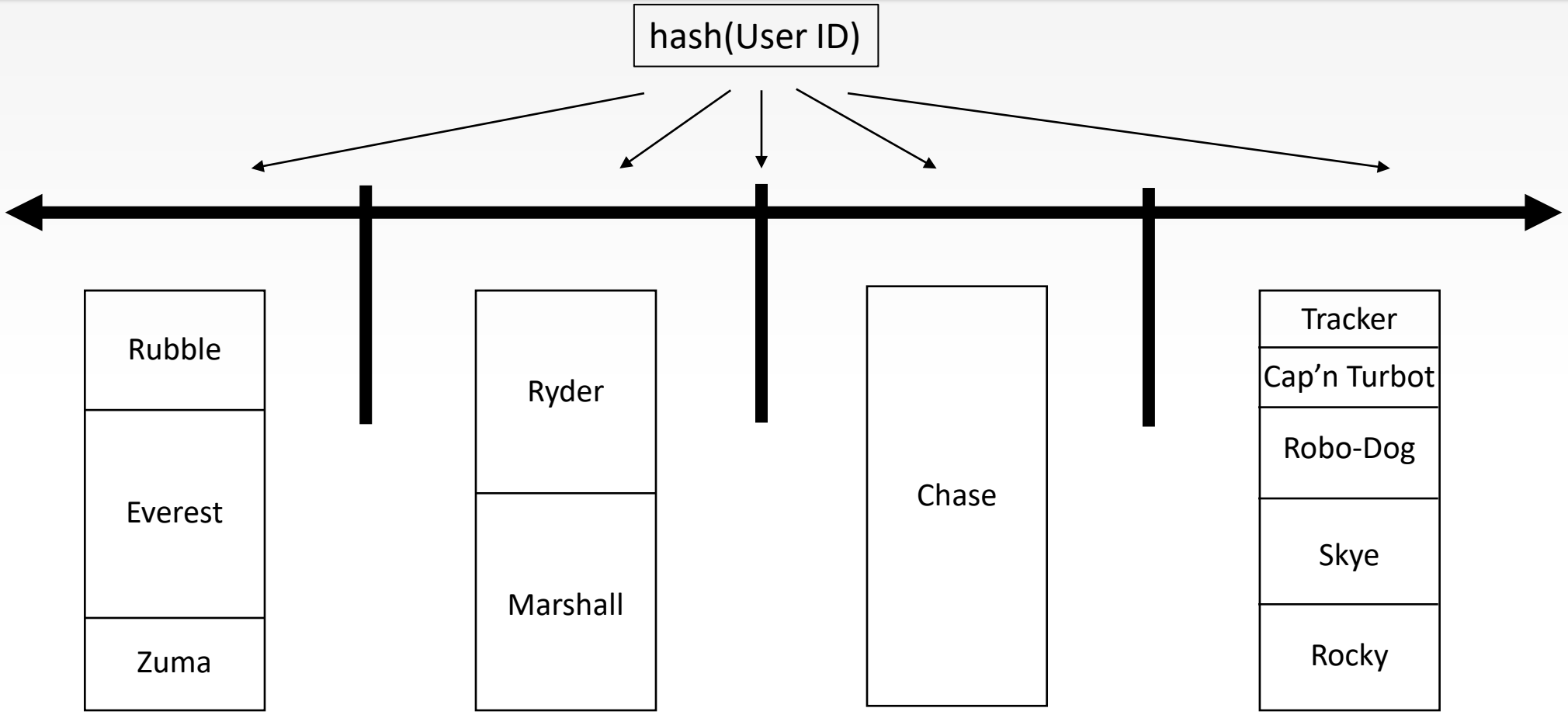
Cosmos DB Container

Partition Key: User ID

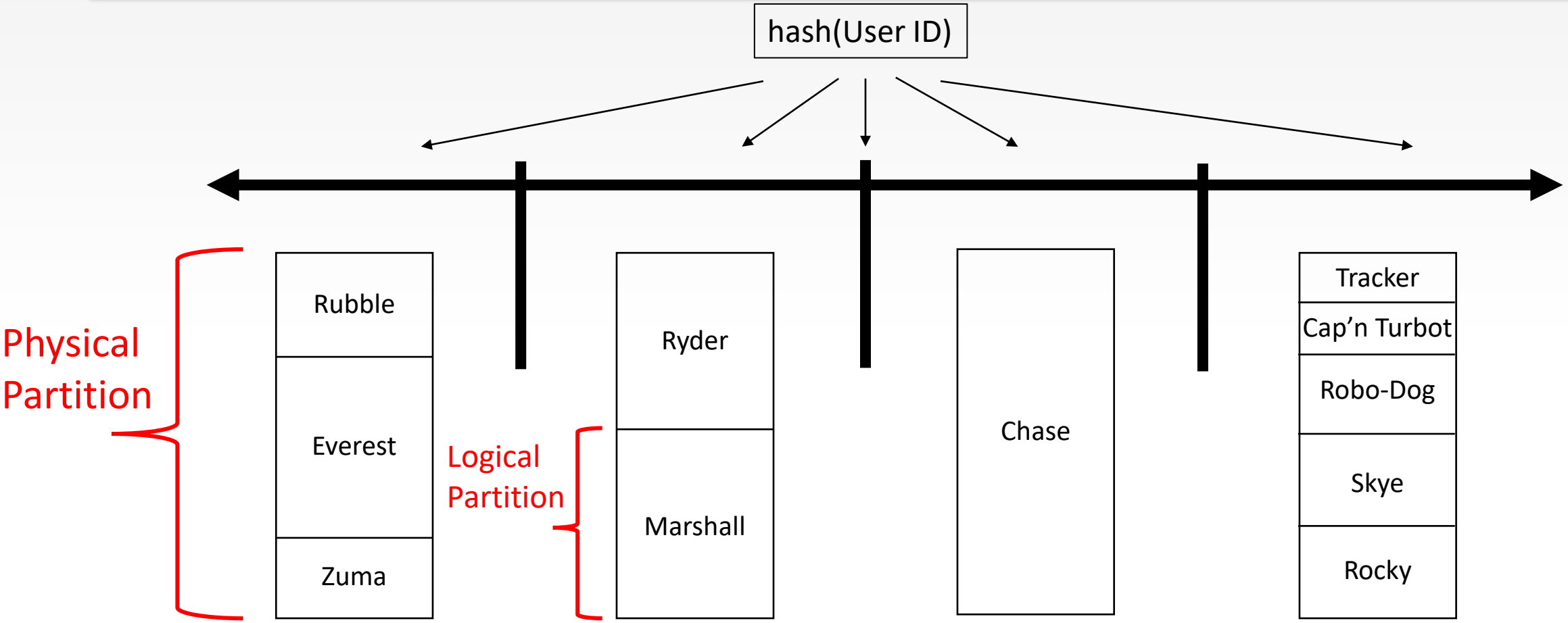
Partitioning



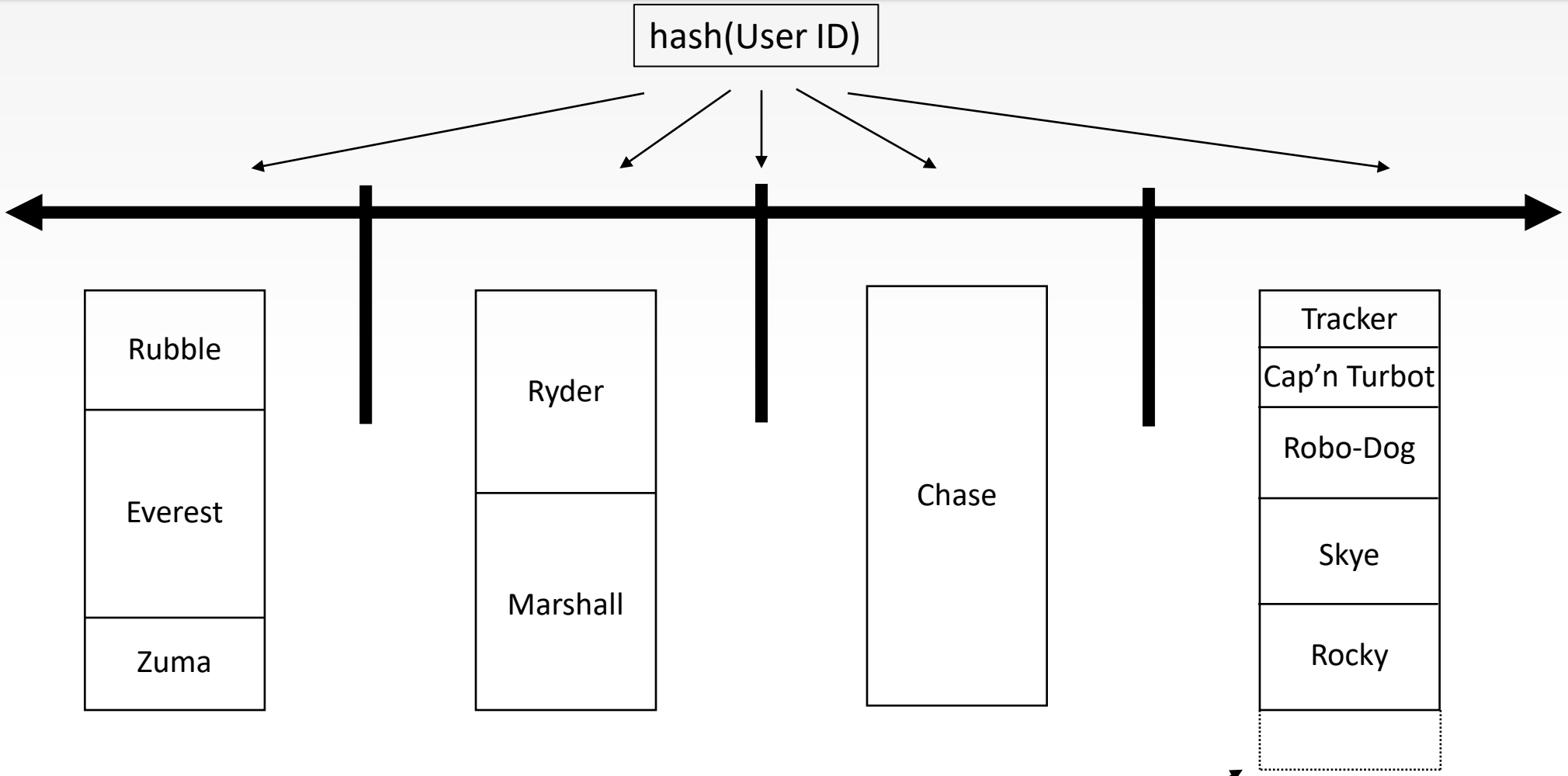
Partitioning (Physical Partition Sets)



Partitioning (Physical Partition Sets)

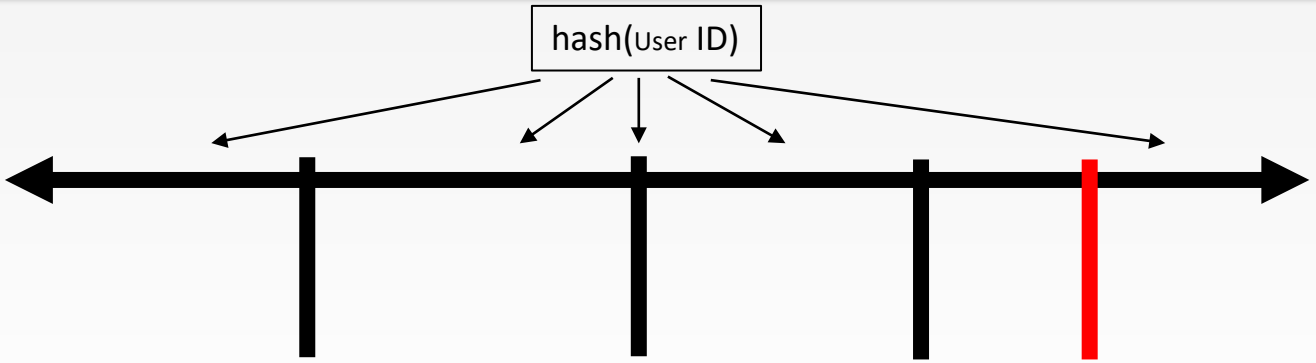


Partitioning (Physical Partition Sets)



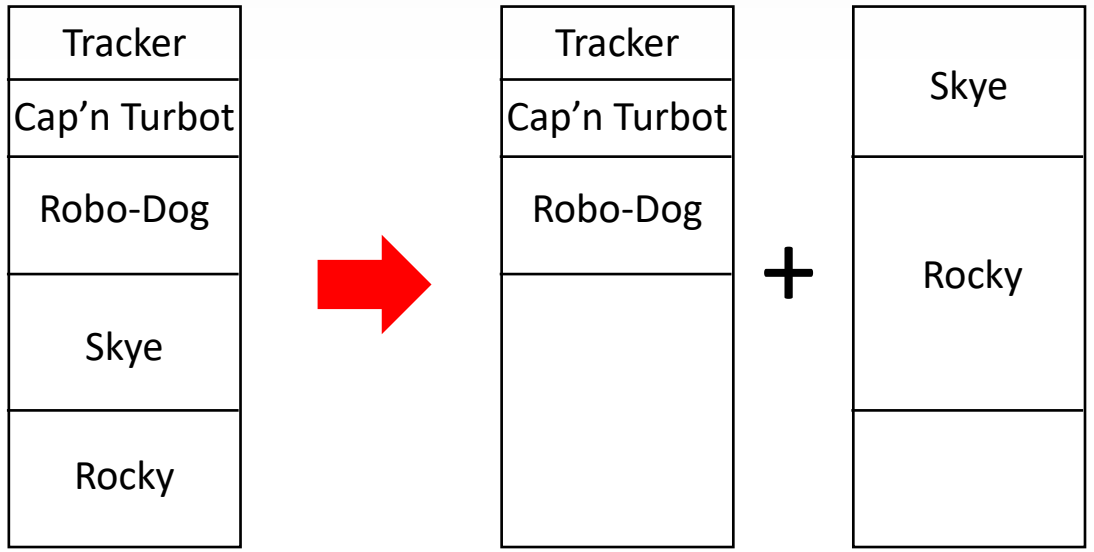
What happens when it needs to grow?

Partitioning (Physical Partition Sets)



Partitions can be dynamically subdivided to grow the database without affecting availability

This is done automatically.



Choosing A Good Partition Key

- Plan to distribute both request and storage volume
  - Remember the 20GB limit
  - Adding dates after partition values can help with this
- For greatest efficiency, queries should eliminate partitions
- Queries can be routed/filtered via partition key
- “Fan-Out” is something to try to avoid where possible

Choosing A Good Partition Key

- Understand your workload!
- Understand the most frequent/expensive queries
- Understand insert vs update ratios
- Remember partition keys are logical!
  - Don’t be afraid of having too many
  - More key values = better scalability

Relational Databases: ACID

- **A**tomic
  - Each transaction either completes or rolls back
- **C**onsistent
  - Any change made to the DB is consistent with database constraints. If state is violated, the whole transaction fails
- **I**solated
  - Each transaction runs in an isolated environment & can't interfere with others
- **D**urable
  - A transaction's changes are persisted once it commits

Non-Relational Databases: BASE

- **B**asically **A**vailable
  - Ensure availability of data via replication
- **S**oft state
  - Instead of the database enforcing consistency, developers are responsible
- **E**ventually consistent
  - Until consistency is achieved, reads are still possible but with older data

Distributed Databases: Brewer's CAP Theorem

- **C**onsistency
  - Each Read is of the most recent write (or an error)
- **A**vailability
  - Each request receives a non-error response, no guarantee it's of the most recent write
- **P**artition tolerance
  - System can operate despite messages being dropped/delayed between nodes
- [PICK TWO]

Consistency: ACID vs. CAP

- Problem: There's no consistent definition of "consistent" in CS
- Transactions (ACID)
  - Each transaction moves from a single valid state to another
- Replication (CAP)
  - Getting a consistent view across replicated copies of data
  - And CAP doesn't even cover all cases….

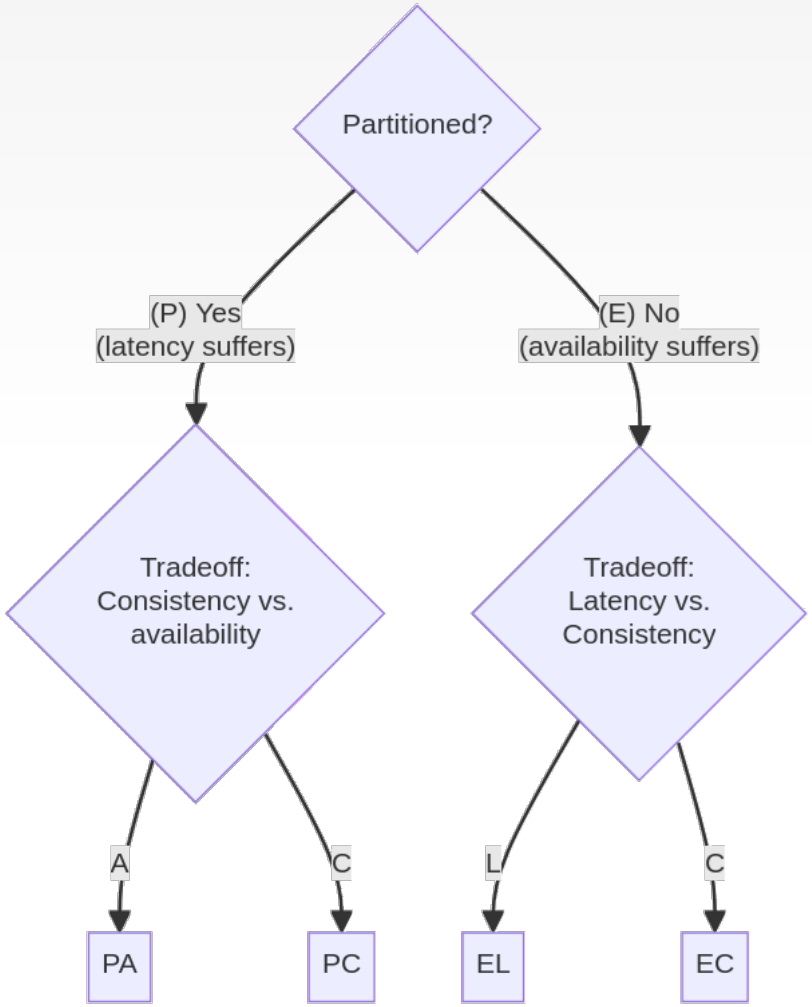
https://www.pexels.com/photo/brown-glass-bottle-with-liquid-and-pipette-4021773/

Consistency: PACELC Theorem

- An extension of the CAP Theorem
- **P**artitioning: **A**vailability vs. **C**onsistency **E**LSE **L**atency vs. **C**onsistency
- When partitioning a distributed system you have to choose between availability and consistency, but also when not partitioning one must choose between latency and consistency.

Consistency: PACELC Theorem

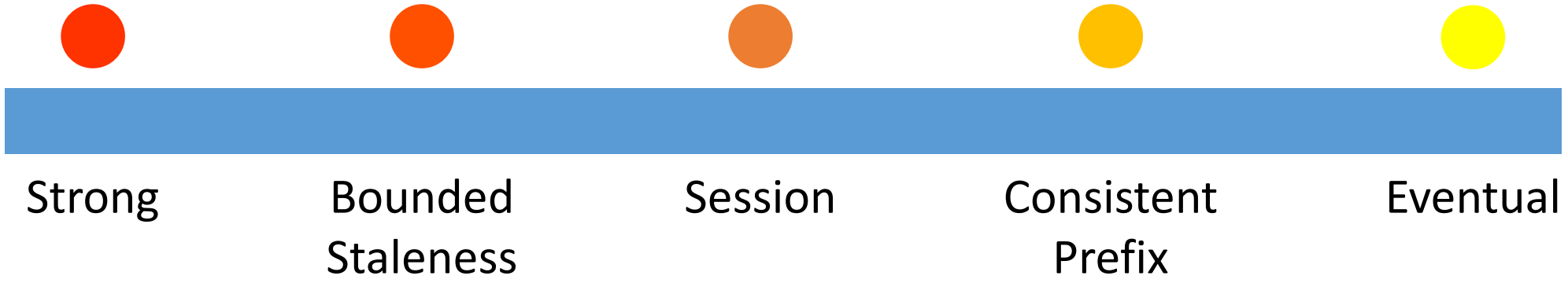
- Reader is far away from writer
- Value gets updated by writer
- Should the reader:
  - See the old value? (prioritize latency)
  - See the same result as the master?
  - Wait for the new value (prioritize consistency)



https://commons.wikimedia.org/wiki/File:PACELC_theorem.png

Azure Cosmos DB Consistency Models

- Azure Cosmos DB has 5 of them
- You can choose what gets prioritized
- Can be overridden on a per-request basis



So Which DB Should I Use When?

Relational Bad Ideas: Logs

- Logs are mostly written and rarely read
  - Reading from a log probably means troubleshooting an issue
  - Otherwise probably never reading
- Logs are best written synchronously
- Reads are usually sequential (and not random)
- Kept for a period of time and then deleted

Relational Bad Ideas: Logs

- Logs aren't normalized; not 1NF compliant
  - We usually only care about parts of the message
  - Leads to lots of wildcard searches
- Logs aren't transactional
- If database failures occur, we can't log anything
- In short – log to a text file
  - And maybe load them into a service for fast querying

https://www.pexels.com/photo/brown-chop-logs-on-outdoor-296333/

Relational Bad Ideas: Images

- Binary data is a horrible idea in a database
- Turns the database into a file system
- Usually written once, and read in its entirety
- Makes the DB (and backups, and log) larger
- Probably doesn’t compress well
- Not transactional
- Database adds zero value
- Use a file system instead!

Relational Bad Ideas: Lots of Text

- Relational DBs don’t do much for large amounts of text
- Like log data, lots of wildcard searches (eg ‘%foo%’)
- Like images, will probably be reading it all
- The worst of both worlds!
- Document-based DB may be a better fit here

NoSQL Bad Ideas: Strict ACID Required

- If this is a requirement, stick with an RDBMS
- Financial information
- Operations requiring transactions to ensure consistency

NoSQL Bad Ideas: Already/Easily Relational Data

- Data is already coming from a RDBMS
- Data can easily be arranged into relational entities
- No need to re-invent the wheel!

NoSQL Great Ideas

- Schemaless / Unstructured Data
- Denormalized Data
- Massive scale-out requirements

Questions?

@bobp.bsky.social

@sqlbob

bob@bobpusateri.com

bobpusateri.com

bobpusateri