AWS Purpose-Built Databases

for Architects

Andreas Juffinger

Senior Solutions Architect,
Amazon Web Services , Austria

Data almost always outlives the system.

Access and Preservation of Data is Key

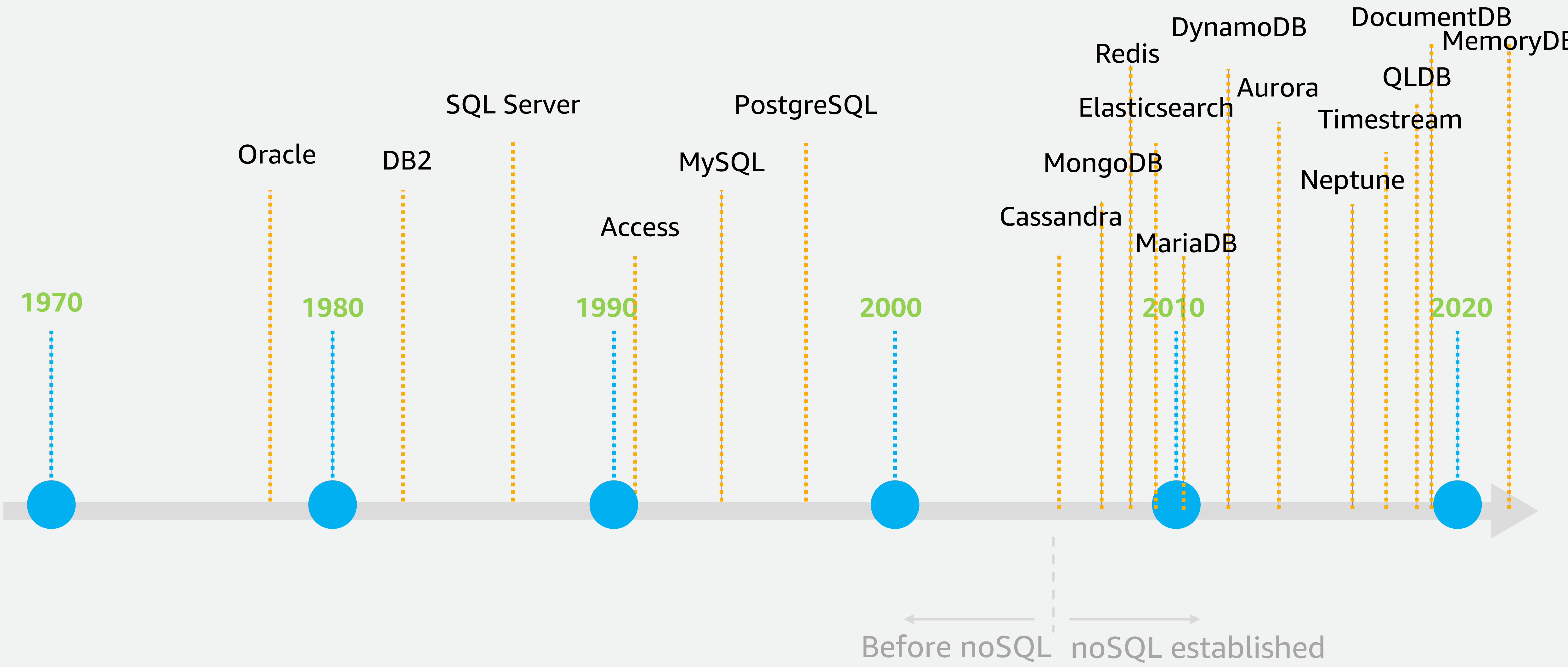
Database (DB)

Primary job of any database is to reliably store data and make it available for users.

Def.: *"A database is a logical collection of data which is managed by a specific software – the database management system (DBMS)*

A database includes not only user data but also the objects necessary for its management (e.g. indexes or logfiles)."

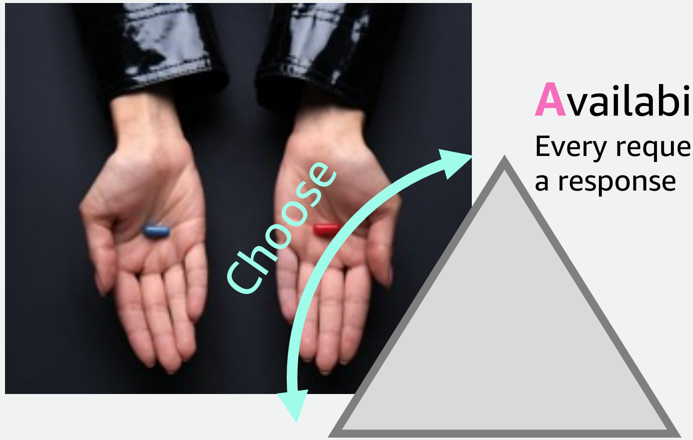
https://db-engines.com/en/article/Database


1970
Oracle
1980
DB2
SQL Server
1990
Access
MySQL
PostgreSQL
2000
Cassandra
MongoDB
Redis
Elasticsearch
2010
MariaDB
DynamoDB
Aurora
Neptune
Timestream
QLDB
DocumentDB
2020
MemoryD
Before noSQL
noSQL established

Database Design Theorems

CAP: Iron Triangle of Data

You cannot have both - choose between availability and consistency if you want to support partition tolerance (separation of replicas).



Availability:
Every request receives a response

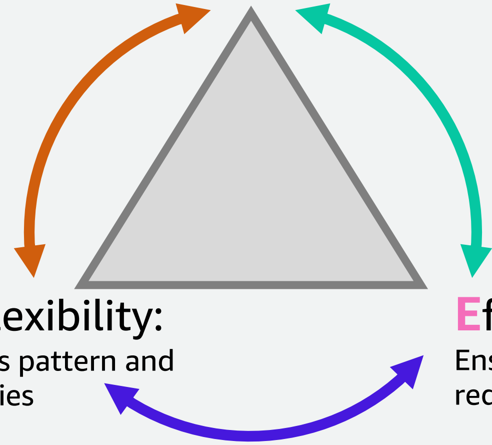
Consistency:
All replicas of same data have same value

Partition Tolerant:
System operates when connectivity between replicas is lost.

PIE: Iron Triangle of Purpose

You cannot have all – choose two out of pattern flexibility, infinite scale, and efficiency.

Infinite Scale:
Scale gracefully without practical limits.



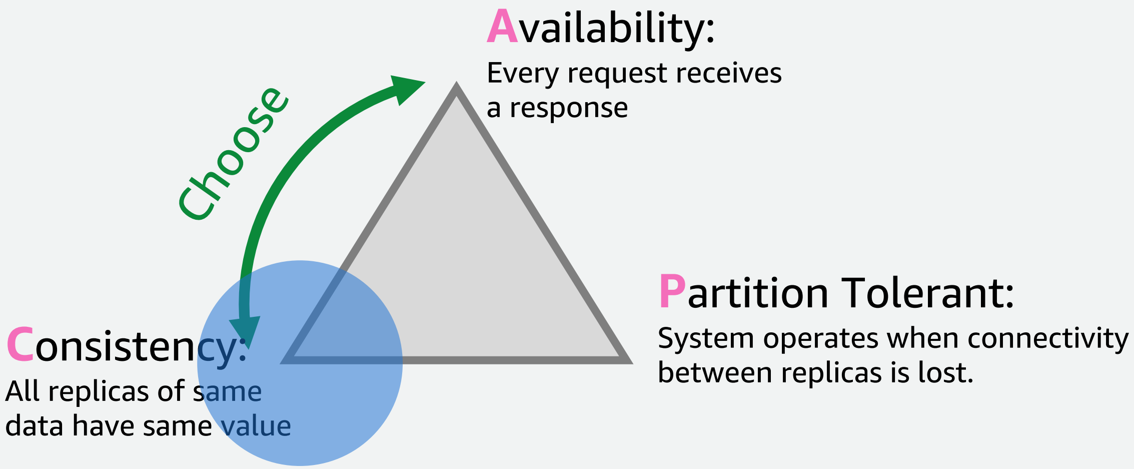
Pattern Flexibility:
Random access pattern and one-time queries

Efficiency:
Ensure system provides required latency at all times.

Relational Database

Storage (on disk and in memory) used to be a driver for database design.

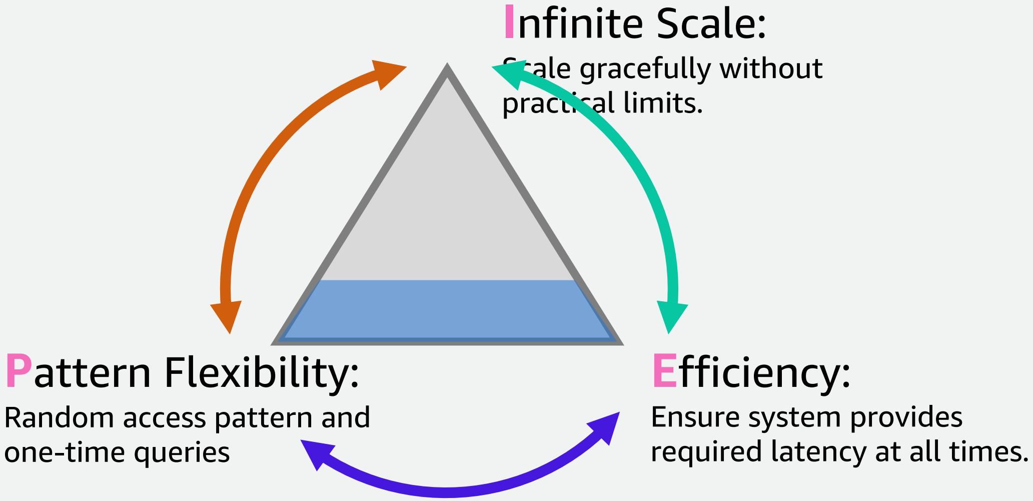
The relational algebra and database normalization aims for removing redundancy to avoid anomalies.



Availability:
Every request receives a response

Partition Tolerant:
System operates when connectivity between replicas is lost.

Consistency:
All replicas of same data have same value



Infinite Scale:
Scale gracefully without practical limits.

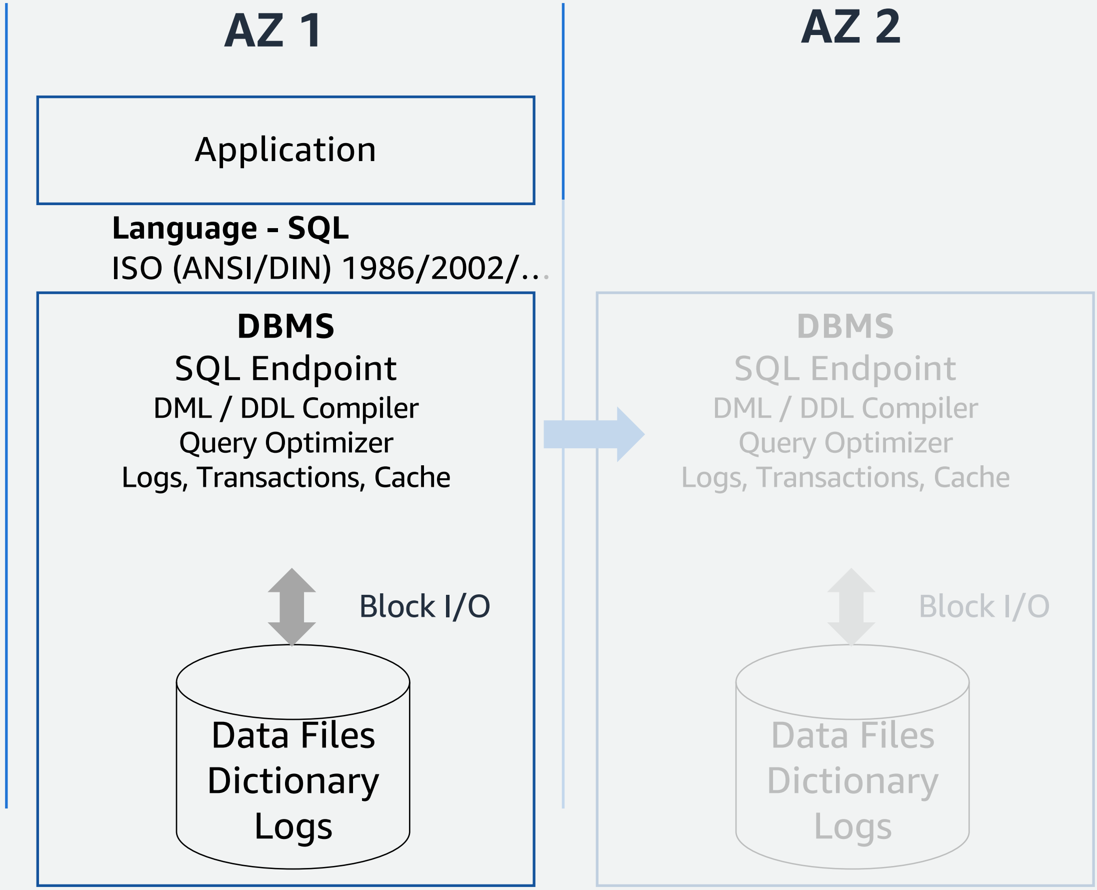
Pattern Flexibility:
Random access pattern and one-time queries

Efficiency:
Ensure system provides required latency at all times.

Systems where infinite scale is sacrifieced in favor of a flexible and efficient application

Cloud Native Relational Databases

Traditional RDBMS



Cloud Native Principles

Compute and storage have different lifetimes, compute requires higher agility

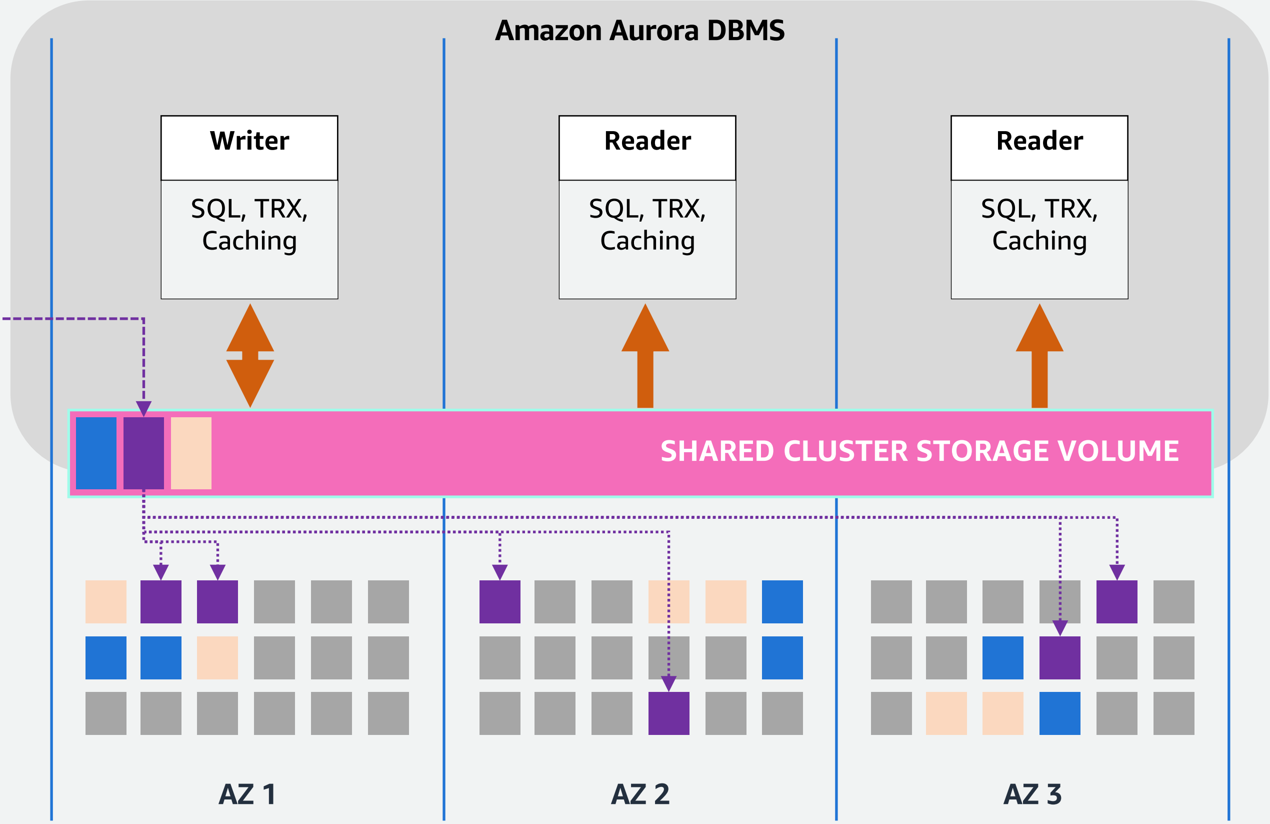
- Instances are scaled up/down
- Instances are added to cluster to scale out
- Instances are shut down
- Instances fail and may be replaced

Compute and storage are best **decoupled** for scalability, availability and durability

Highly Durable, Fault Tolerant, Cluster Storage

Decoupled Compute and Storage

- Purpose-build log-structured distributed storage provided as cluster storage volume with a continuous backup to S3.
- Storage volume segmented in 10 GiB **protection groups (PG)**
- Six copies of data, two in each Availability Zone to protect against AZ+1 failure modes



Transactional vs Analytical Processing

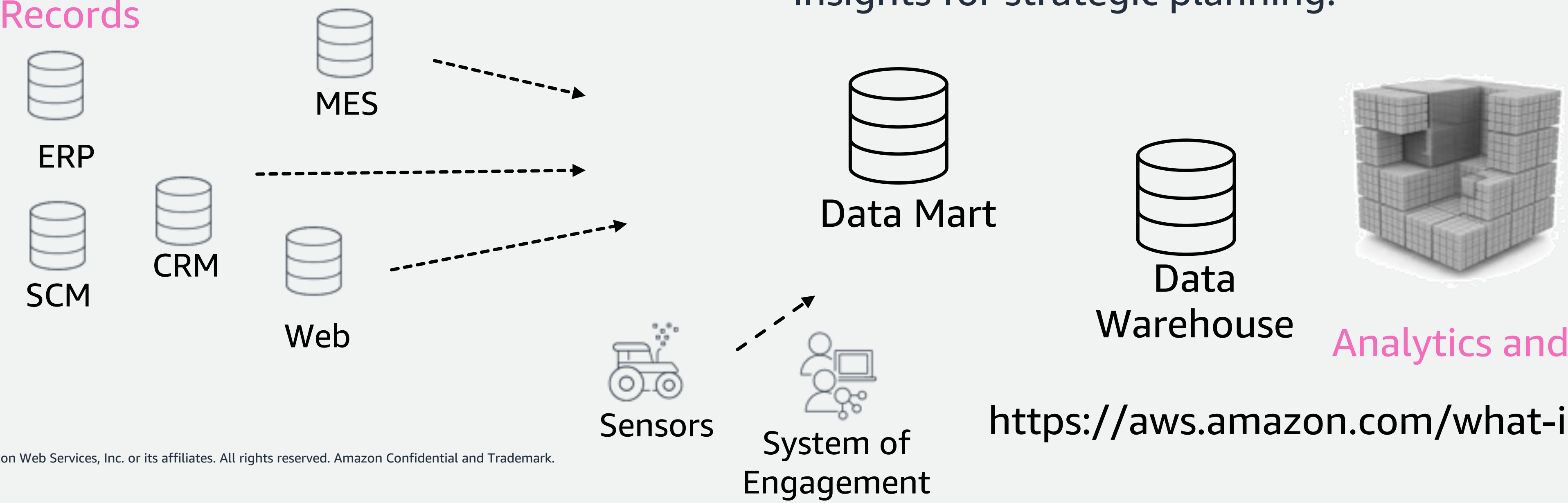
OLTP

Online transaction processing (OLTP) is a data technology that stores information quickly and reliably in a database. OLTP database management systems focus on creating, updating, and deleting records in transactions.

OLAP

Online analytical processing (OLAP) is software technology you can use to analyse business data from different points of view. These databases store historical data from multiple sources. OLAP databases allow users to view different summaries of multidimensional data to provide actionable insights for strategic planning.

System of Records



ERP

MES

SCM

CRM

Web

Sensors

System of Engagement

Data Mart

Data Warehouse

Analytics and Reporting

https://aws.amazon.com/what-is/olap/

Row and Column Oriented DBMS

Table:

ID	NAME	SAL	DEP
1	Smith	800	11
2	Jones	900	13
3	Martin	901	22
4	Scott	777	11

◎ Row oriented DBMS stores this table on disc:

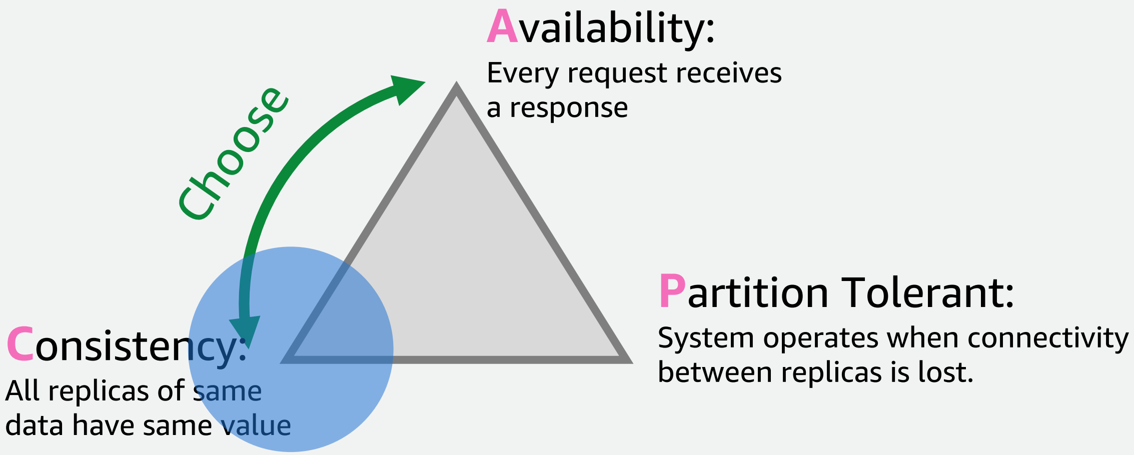
```
1 ; Smith ; 800 ; 11
2 ; Jones ; 900 ; 13
3 ; Martin ; 901 ; 22
4 ; Scott ; 777 ; 11
```

◎ Columnar DBMS stores this table on disc:

```
NAME 1:Smith; 2:Jones; 3:Martin; 4:Scott
SAL  1:800; 2:900; 3:901; 4:777
DEP  1:11; 2:13; 3:22; 4:11
```

Analytical Databases

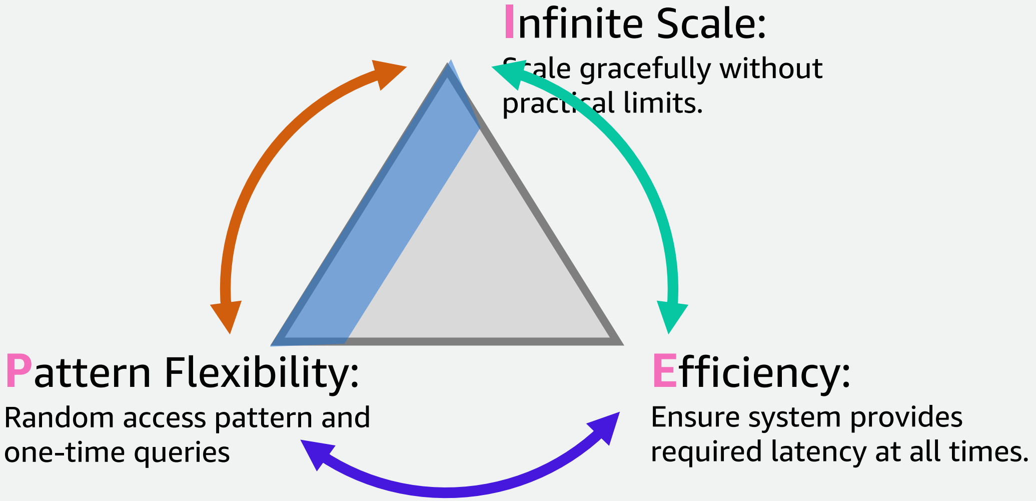
Analytical databases are designed to support aggregations and analytic queries with window functions on a large scale



Availability: Every request receives a response

Consistency: All replicas of same data have same value

Partition Tolerant: System operates when connectivity between replicas is lost.



Infinite Scale: Scale gracefully without practical limits.

Pattern Flexibility: Random access pattern and one-time queries

Efficiency: Ensure system provides required latency at all times.

Systems where efficiency is sacrifieced in favor of pattern flexibility and scalable application

Amazon Redshift: Cloud Data Warehouse

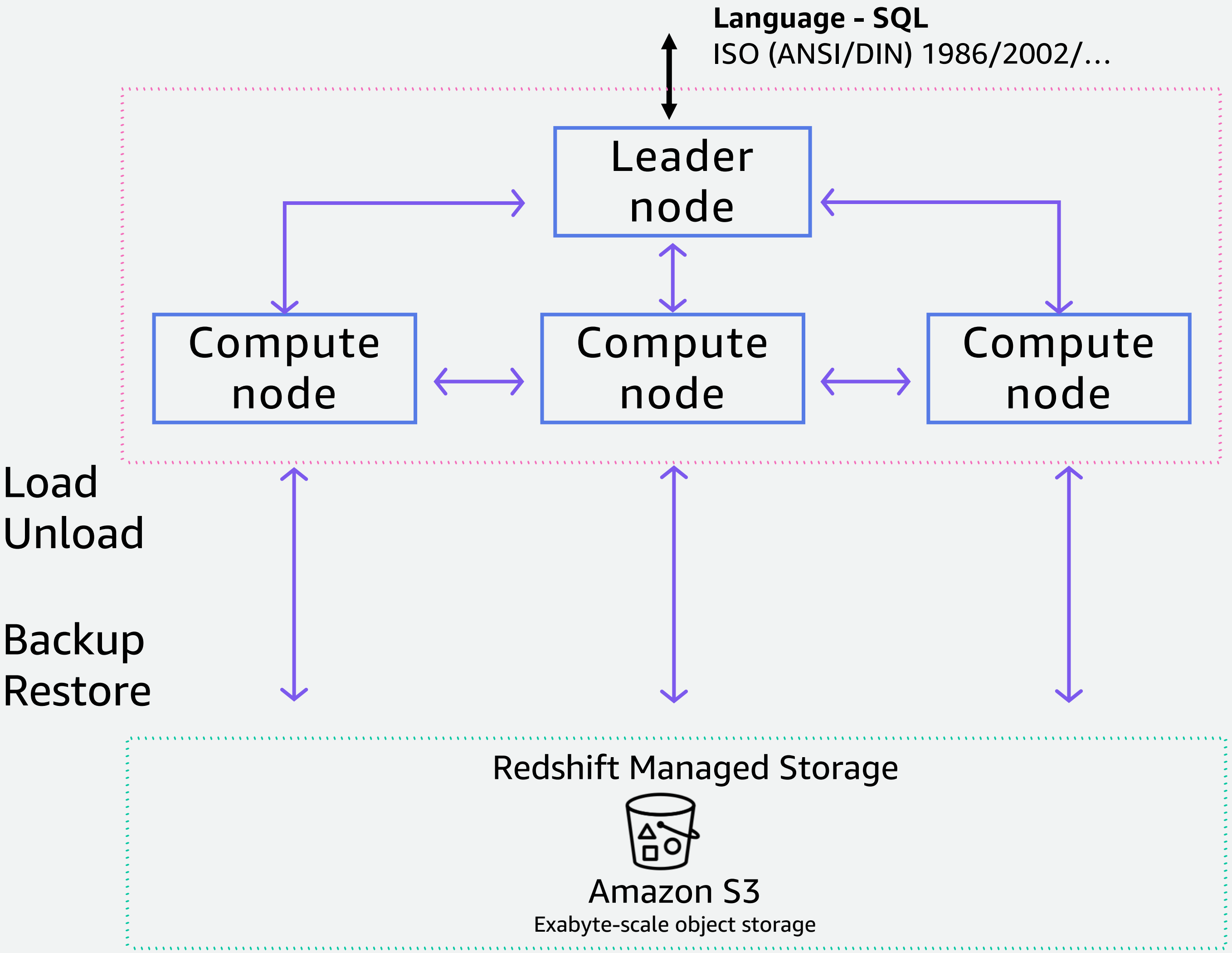
Amazon Redshift is an analytical database a Data Warehouse built for the cloud.

Leader Node

- SQL endpoint
- Stores metadata
- Coordinates parallel processing

Compute Nodes

- Local, columnar storage
- Executes queries in parallel
- Load, unload, backup, restore from S3



Consistency Model

ACID is an acronym

*for **A**tomicity, **C**onsistency, **I**solation, and **D**urability. It is a model to define and maintain consistency and integrity in a structured database.*

Atomic: All components of a transaction are treated as a single action. All are completed or none are; if one part of a transaction fails, the database's state is unchanged.

Consistent: Transactions must follow the defined rules and restrictions of the database, e.g., constraints, cascades, and triggers. Thus, any data written to the database must be valid and any transaction that completes will change the state of the database. No transaction can create an invalid data state. Note that this is different from "consistency" as it's defined in the CAP theorem.

Isolated: Fundamental to achieving concurrency control, isolation ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially, i.e., one after the other. With isolation, an incomplete transaction cannot affect another incomplete transaction.

Durable: Once a transaction is committed, it will persist and will not be undone to accommodate conflicts with other operations. Many argue that this implies the transaction is on disk as well; most formal definitions aren't specific.

BASE is an acronym

*for **B**asically **A**vailable **S**oft state **E**ventually consistent. It is a method for maintaining consistency and integrity in a structured or semi-structured database.*

Basically Available: BA allows for one instance to receive a change request and make that change available immediately. The system will always guarantee a response for every request. However, it is possible that the response may be a failure or stale data, if the change has not been replicated to all nodes. In an ACID system, the change would not become available until all instances were consistent. Consistency in a BASE model is traded for availability.

Soft state: In a BASE system, there are allowances for partial consistency across distributed instances. For this reason, BASE systems are considered to be in a soft state, also known as a changeable state. In an ACID system, the database is considered to be in a hard state because users cannot access data that is not fully consistent.

Eventual consistency: This reinforces the other letters in the acronym. The data will be eventually consistent. In other words, a change will eventually be made to every copy. However, the data will be available in whatever state it is during propagation of the change.

Query Languages

SQL (/ˈsiːkwəl/ "sequel”)

Structured query language includes query, manipulation and schema definition for databases following the relational model.

- 1986 – Initial ANSI Standard : SQL 86
- …
- 2003 – Window functions, sequences and autogenerated values
- 2016 – Row pattern matching
- 2019 – Added data type multidimensional array

noSQL

The acronym NoSQL is often understood as "Not Only SQL", implying that relational systems are a proven technology but not necessarily the optimal choice for each kind of intended use.

NoSQL systems are a heterogenous group of very different database systems. https://db-engines.com/en/article/NoSQL

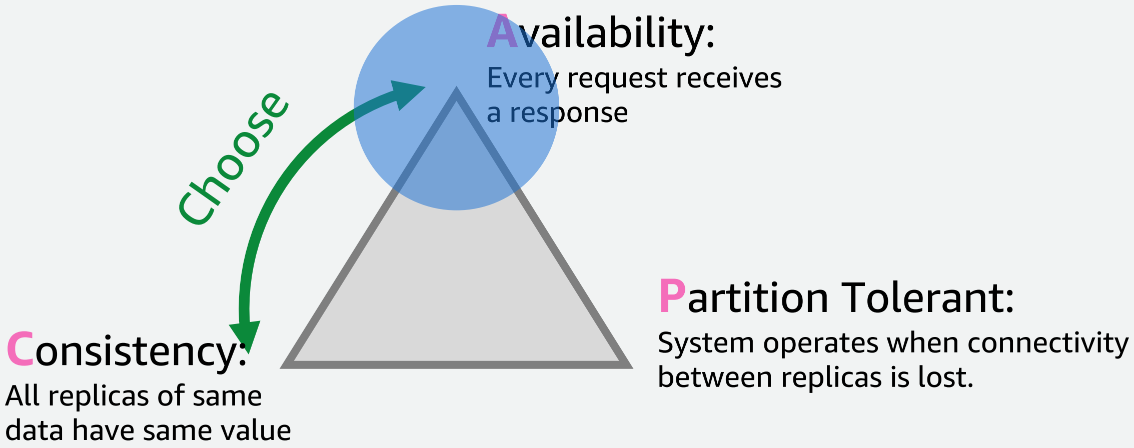
The following categories are well accepted:

- Key-Value Stores
- Graph DBMS
- Document Stores
- Timeseries DBMS
- Wide Column Stores

a.o. Native XML DBMS, Content Stores, Search Engines

Key-Value Stores

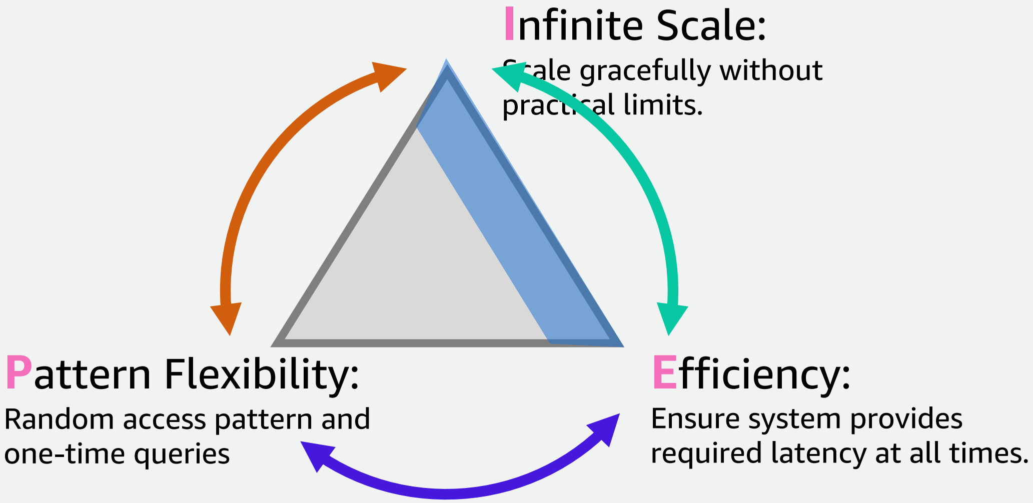
Internet-scale applications have now requirements to databases which can scale for concurrency, data volume and handle peaks in traffic elastic (scale-out and scale-in)



Availability:
Every request receives a response

Consistency:
All replicas of same data have same value

Partition Tolerant:
System operates when connectivity between replicas is lost.



Infinite Scale:
Scale gracefully without practical limits.

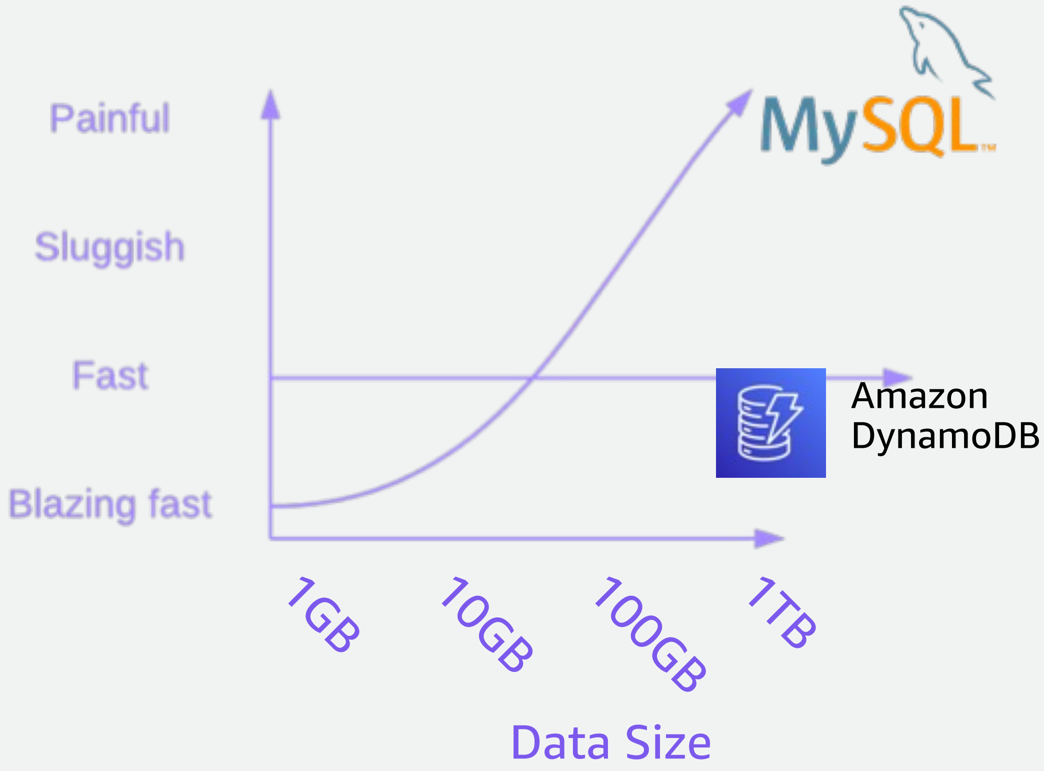
Pattern Flexibility:
Random access pattern and one-time queries

Efficiency:
Ensure system provides required latency at all times.

Systems where pattern flexibility is sacrifieced in favor of highly scalable and efficient application

Amazon DynamoDB

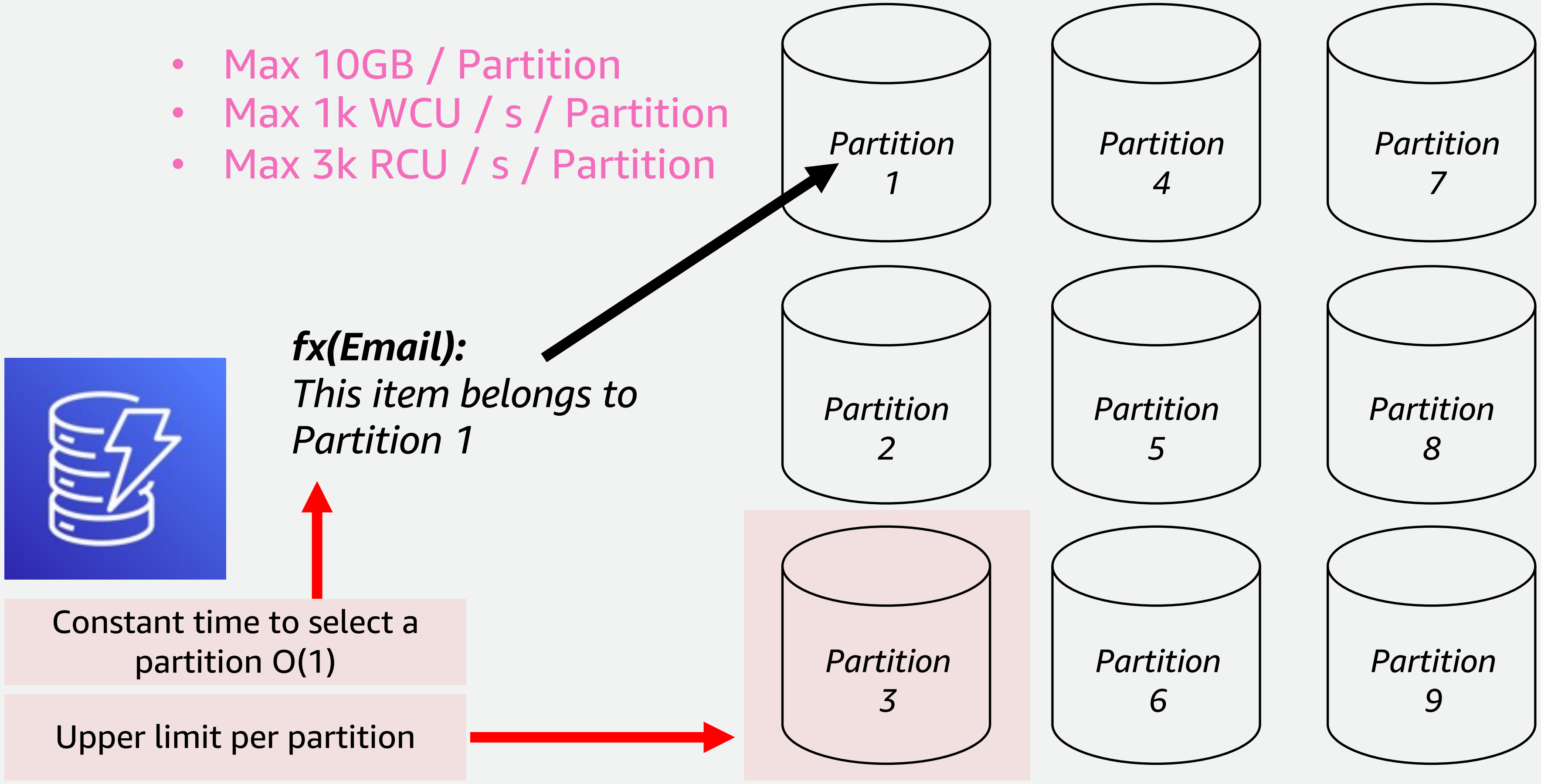
- Consistent response time, regardless of:
  - Database size
  - Concurrent queries
- Reduce need for expertise
- Reliable fully managed service



Amazon DynamoDB – Consistent Response Time

- Max 10GB / Partition
- Max 1k WCU / s / Partition
- Max 3k RCU / s / Partition

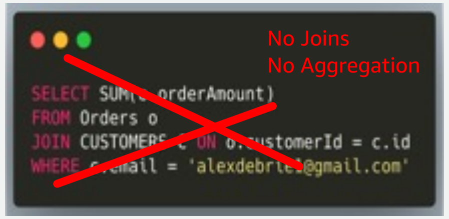
PutItem:
Email: “alexdebrie1..”
OrderId:“01F37K…”



fx(Email):
This item belongs to Partition 1

Constant time to select a partition O(1)

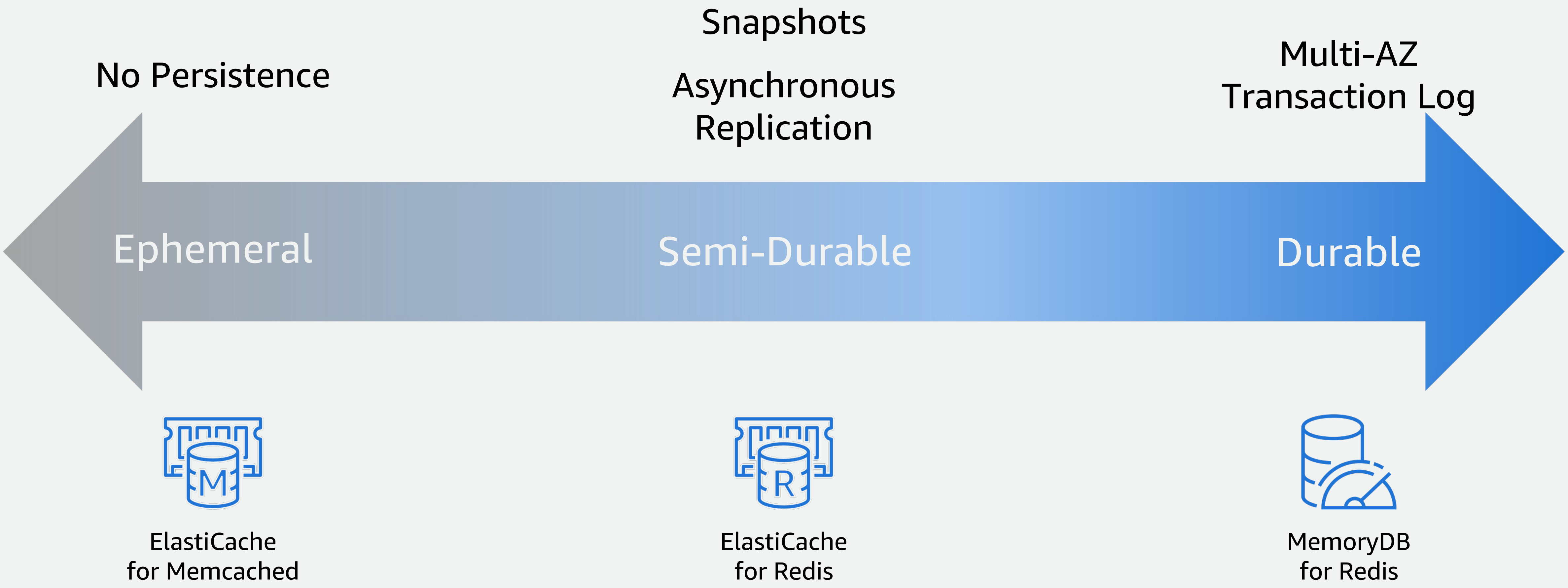
Upper limit per partition



No Joins
No Aggregation

```
SELECT SUM(o.orderAmount)
FROM Orders o
JOIN CUSTOMERS c ON o.customerId = c.id
WHERE c.email = 'alexdebrie1@gmail.com'
```

Memory Based Key-Value Stores



No Persistence

Snapshots

Asynchronous Replication

Multi-AZ Transaction Log

ElastiCache for Memcached

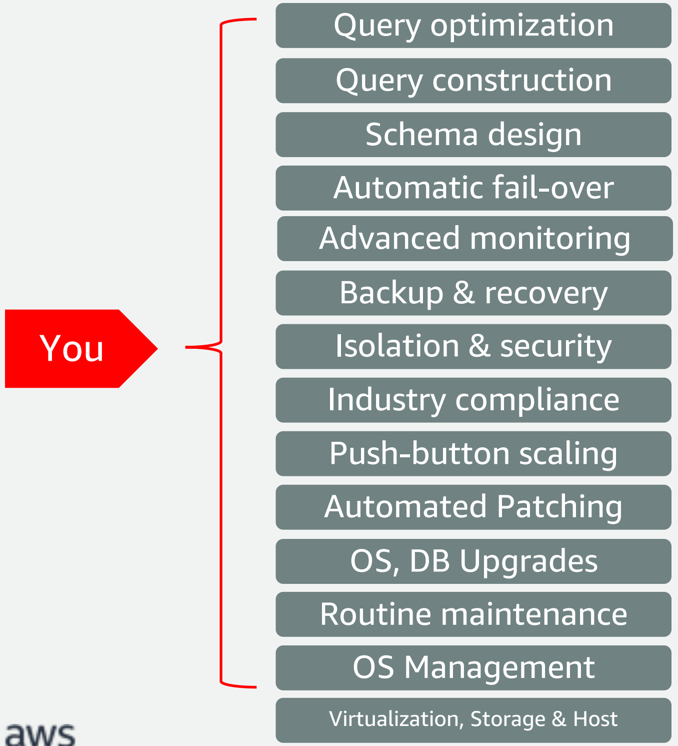
ElastiCache for Redis

MemoryDB for Redis

Operational Responsibilities

Customer Managed

Highest degree of flexibility and complexity. Lots of operational effort.

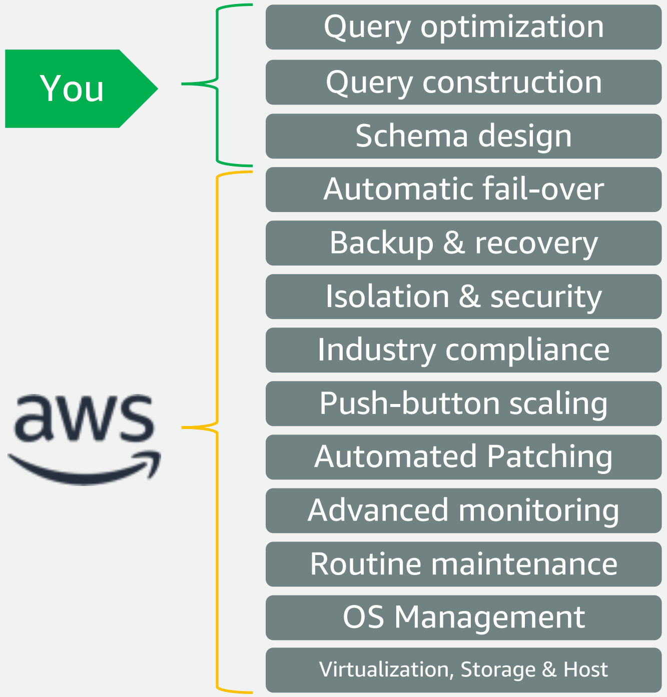


You

- Query optimization
- Query construction
- Schema design
- Automatic fail-over
- Advanced monitoring
- Backup & recovery
- Isolation & security
- Industry compliance
- Push-button scaling
- Automated Patching
- OS, DB Upgrades
- Routine maintenance
- OS Management
- Virtualization, Storage & Host

Managed Services

Free your teams from time-consuming database tasks like server provisioning, patching, and backups.



You

- Query optimization
- Query construction
- Schema design

aws

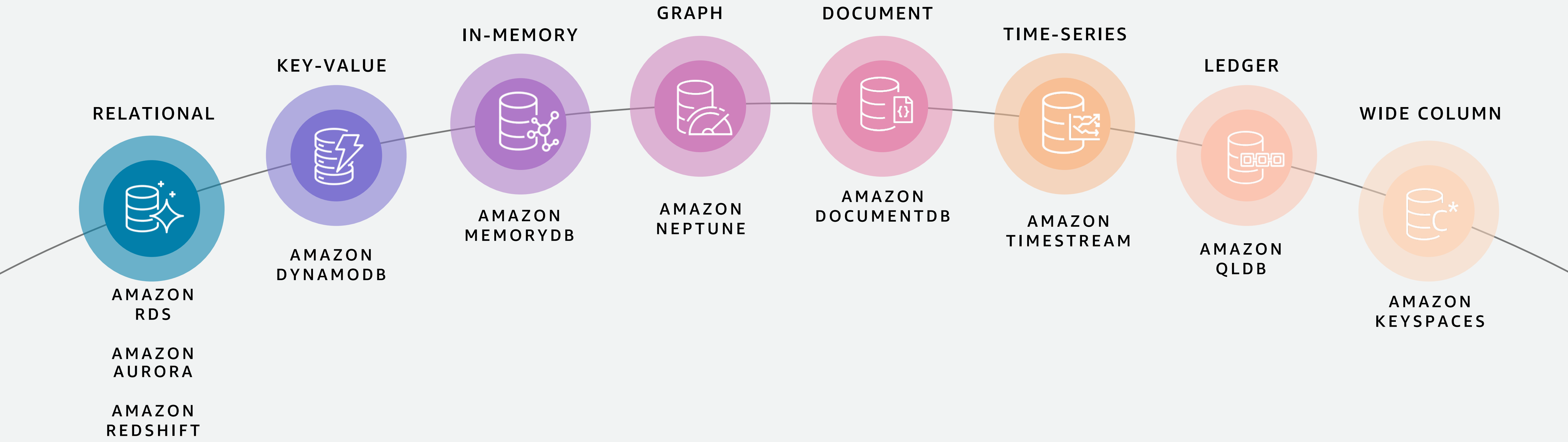
- Automatic fail-over
- Backup & recovery
- Isolation & security
- Industry compliance
- Push-button scaling
- Automated Patching
- Advanced monitoring
- Routine maintenance
- OS Management
- Virtualization, Storage & Host

Serverless

Are managed services which allow you to build and run applications without thinking about servers, nodes or clusters.

Serverless technologies feature automatic scaling, built-in high availability supporting highest degree of agility.

AWS Portfolio of Purpose-Built Databases



RELATIONAL

AMAZON RDS

AMAZON AURORA

AMAZON REDSHIFT

KEY-VALUE

AMAZON DYNAMODB

IN-MEMORY

AMAZON MEMORYDB

GRAPH

AMAZON NEPTUNE

DOCUMENT

AMAZON DOCUMENTDB

TIME-SERIES

AMAZON TIMESTREAM

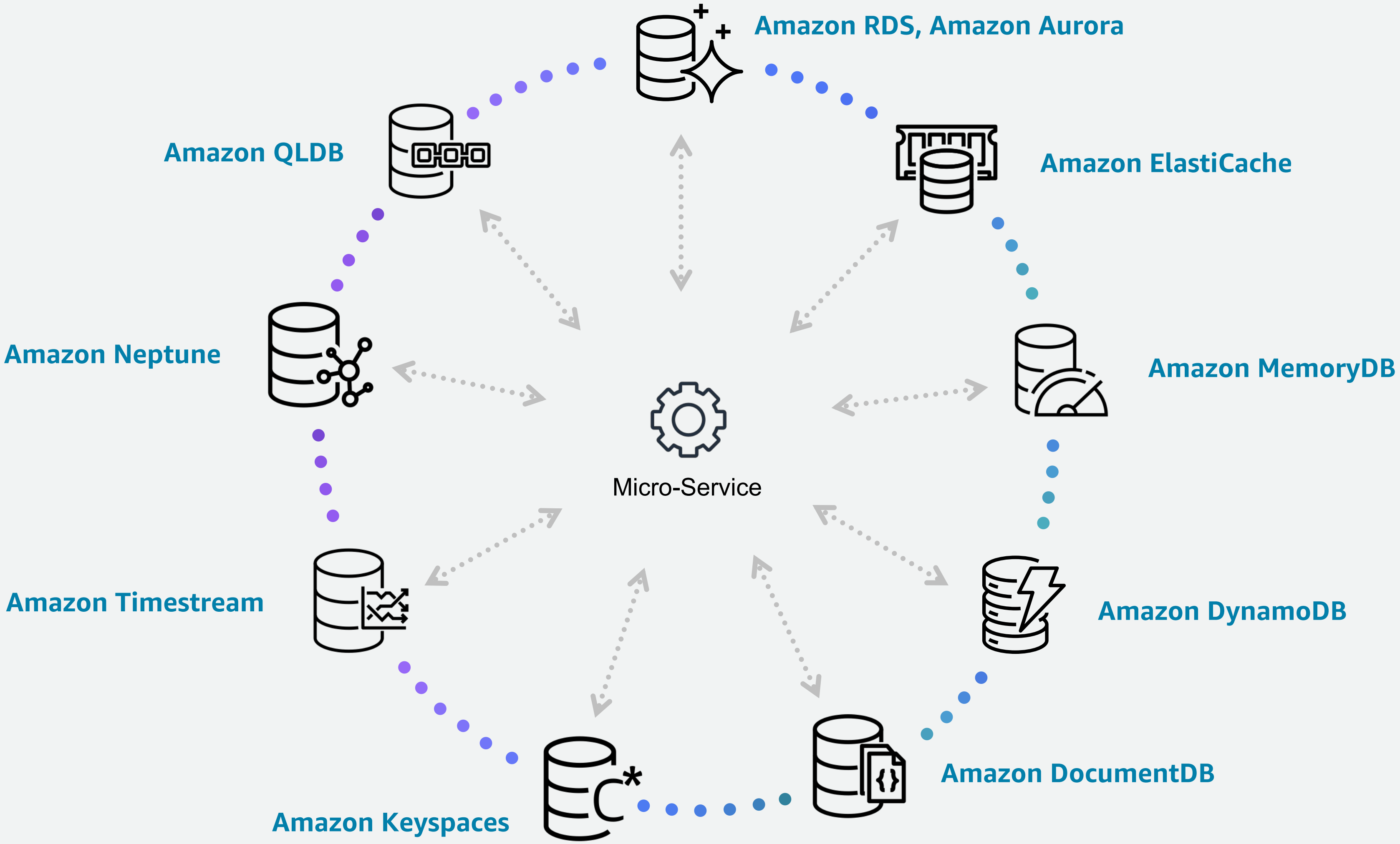
LEDGER

AMAZON QLDB

WIDE COLUMN

AMAZON KEYSPACES

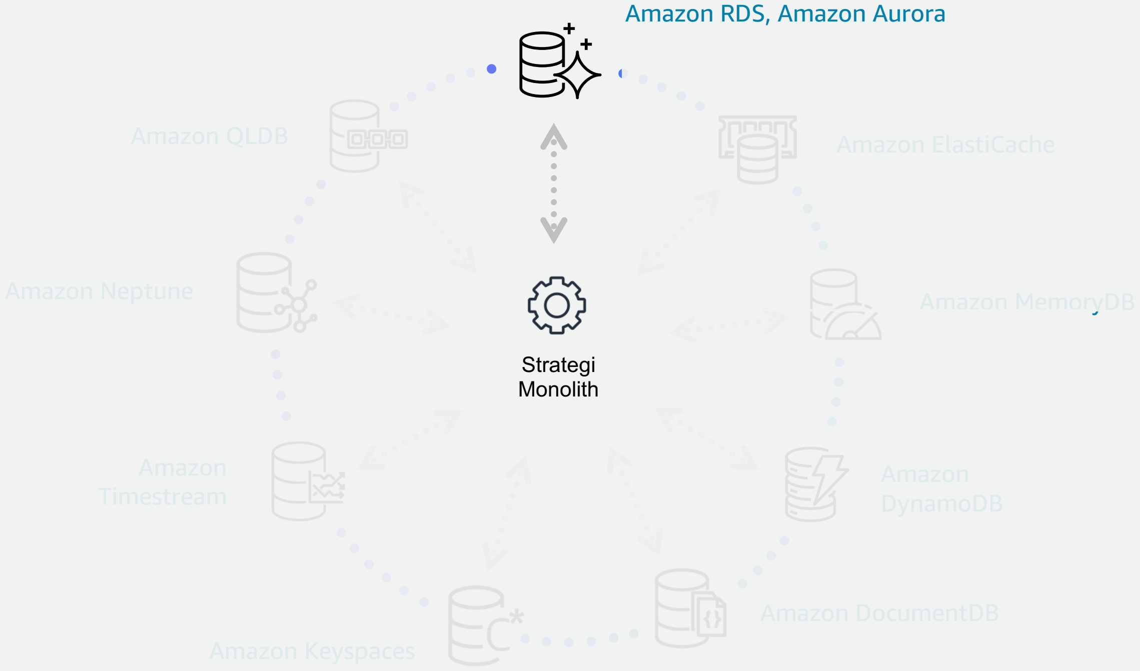
AWS Purpose-Build are Options*) for Architects



*) Options like in stock market (by G. Hohpe)

Starting with the Strategic Monolith

IN AN EARLY STAGE WITH A LOT OF UNCERTAINTY AND UNCLEAR ACCESS PATTERNS



Architectural Decision Record (ADR):

Context: We need for our custom built web shop service a database. The solution is designed as a strategic monolith. Access and query patterns are unknown.

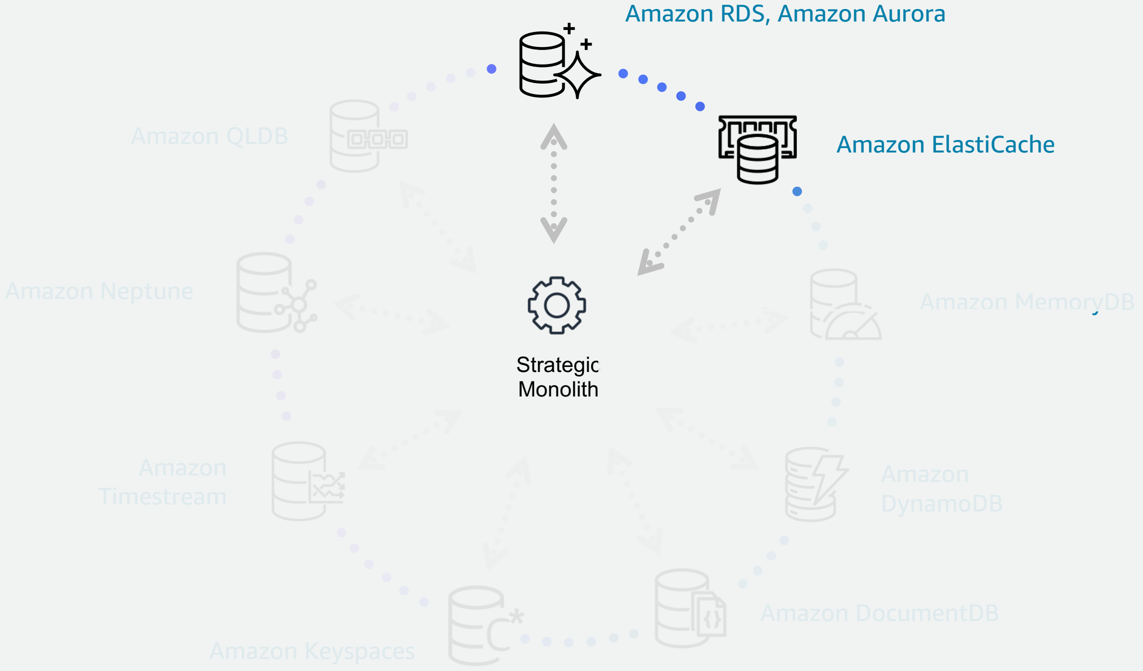
Alternatives: MySQL on EC2, RDS for Oracle, RDS for MySQL, Aurora (Serverless), DynamoDB

Decision: We use Amazon Aurora MySQL as database, as it allows arbitrary query patterns, the team has the experience and tools. A fully managed service is favourable as it reduces workload on the team. The serverless option will make dealing with spiky traffic easier.

Consequences: We expect evolution on the long run, therefore the implementation must follow a strict layering [ADR 03] and only the data layer ‘speaks’ with the DB.

The Startup is Growing

PERFORMANCE ISSUES ARE SEEN ON FREQUENTLY USED DATA



Architectural Decision Record (ADR):

Context: The web shop displays for every user some hot products (configured in DB) retrieved via the /hot API. We see the same request for every visitor.

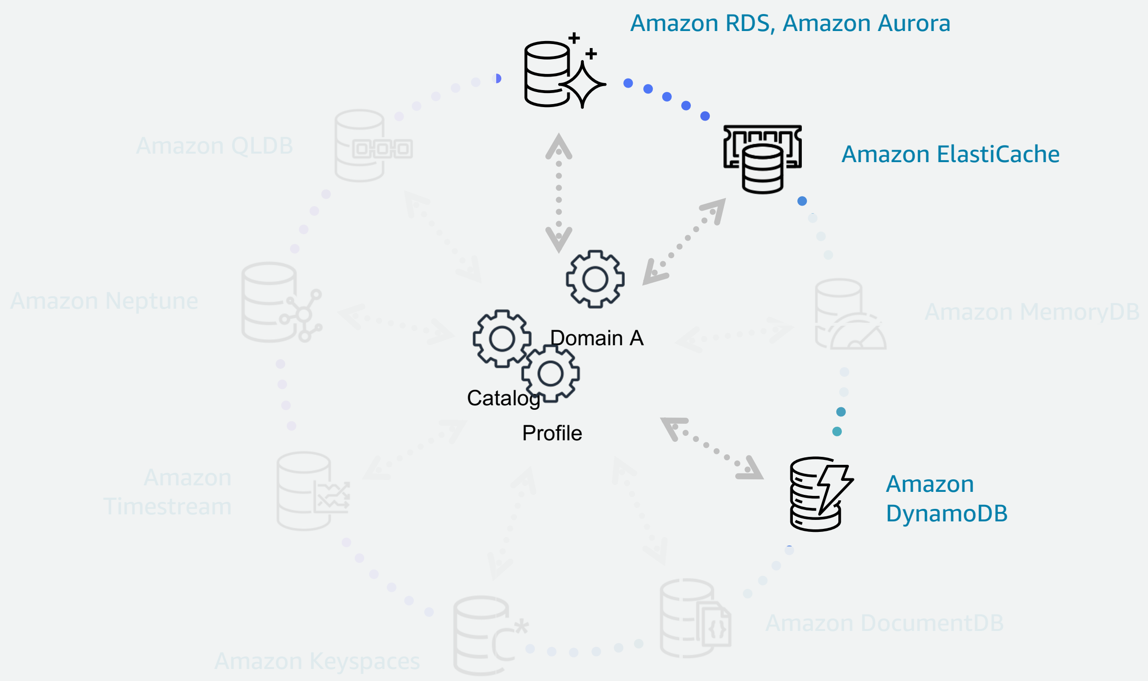
Alternatives: Self Managed Redis, AWS Elasticache, InMemory Data Structure

Decision: We use Amazon Elasticache for REDIS to cache hot products (and other frequently fired queries on slow changing, not visitor specific data). EC supports replication during scale-out and up.

Consequences: During the TTL (15 min) of the cache we wont see the updated hot product and potential stale information

Wow, the second team

ITS TIME TO SPLIT THE MONOLITH, SO THAT THE TWO TEAMS CAN WORK EFFICIENTLY



Architectural Decision Record (ADR):

Context: The product catalog service and profile service are separated and need to be built for massive scale (10mill products, 100mill profiles,)

Alternatives: DynamoDB, MemoryDB

Decision: A clear key/value access pattern has been identified. DynamoDB will be chosen, following a single table approach will support the required query patterns.

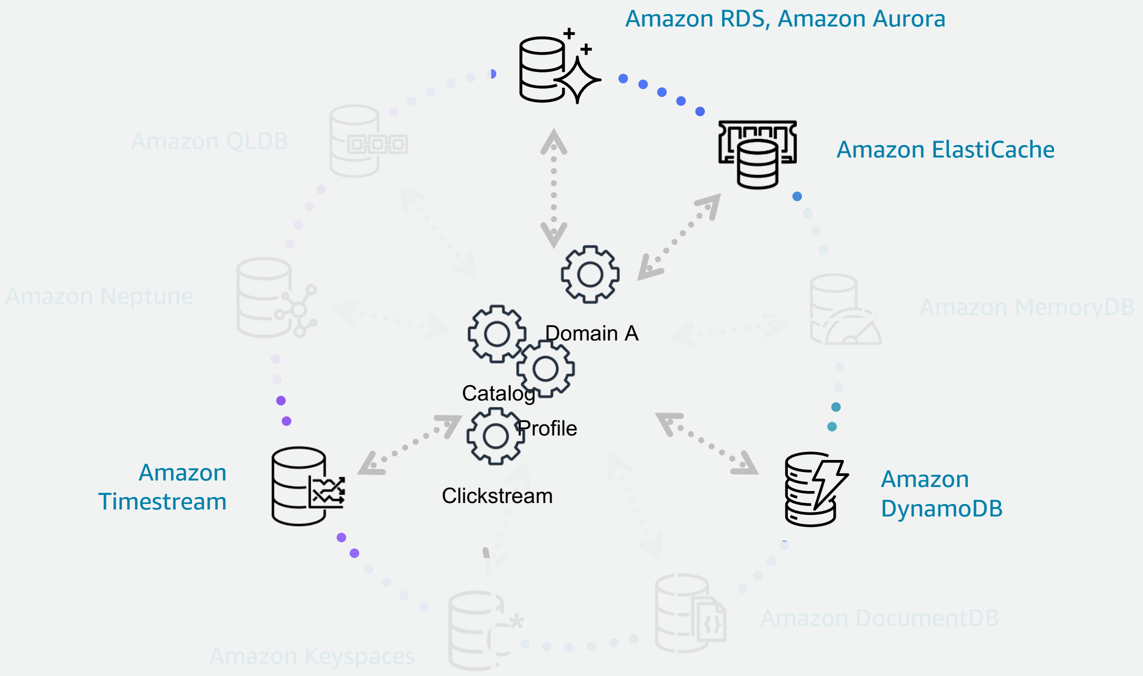
Consequences: Complex joins and aggregations are not possible. Access patterns need to be explicit modeled as LSI / GSI.

All computations need to be done in application layer.

For notifications around updates we will use DynamoDB Streams to emit events.

Collect and Analyze Clickstream…

MARKETING REQUIRES ANALYTICS ON USER BEHAVIOR



Architectural Decision Record (ADR):

Context: We will collect clickstream data from the web shop to map, we need many different ways of aggregating the data. The ingest rate needs to be able to handle very large amount of records per seconds.

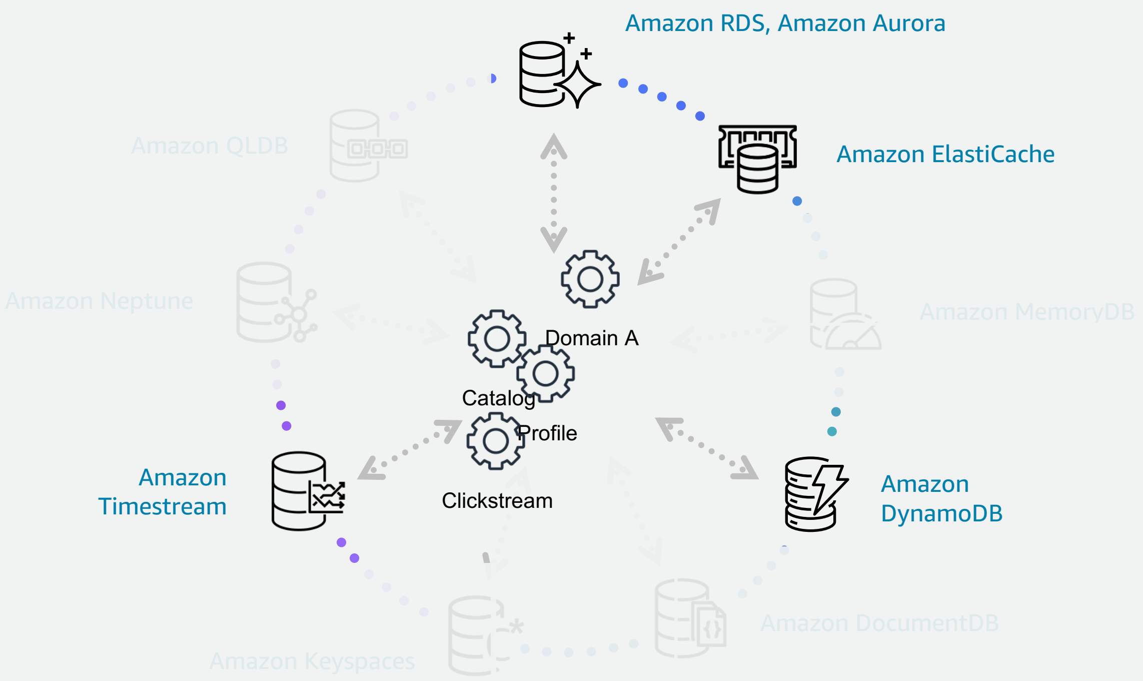
Alternatives: Timestream, DynamoDB

Decision: Analytical access patterns with aggregations are required. Timestream will be chosen and the clickstream will be stored as time series.

Consequences: The dimensionality is limited in timestream to 128 per table.

Parcel tracking …

SHOW HISTORY OF GPS COORDINATES AND STAGES OF THE DELIVERY



Architectural Decision Record (ADR):

Context: As our logistic partner can provide tracking information on a per parcel level we can now show a much more accurate delivery tracking to customers. Simple put/get on the whole timeseries are required with low latency

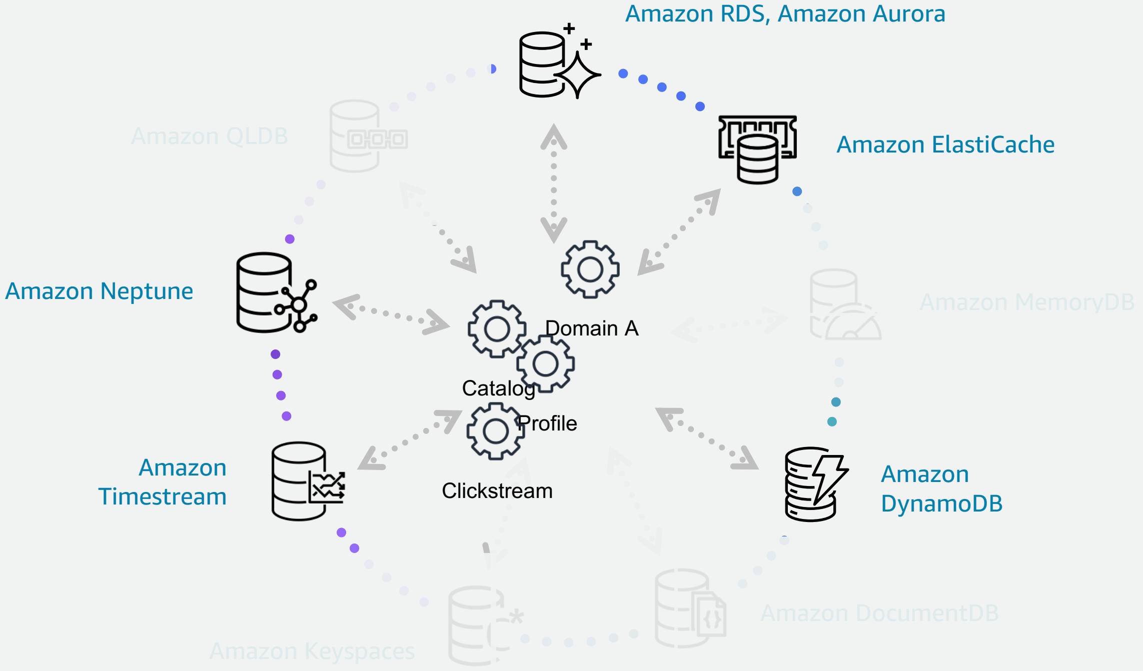
Alternatives: Timestream, DynamoDB

Decision: No analytical access patterns, but low latency get operations are required. DynamoDB will be chosen to write and read the time series as an item collection.

Consequences: DynamoDB will not provide aggregate functions within time series functions, we have to use streams to maintain a “total time” and “total steps” item.

Recomender System – Customer 360 …

FOR A ML BASED RECOMMENDER SYSTEM A GRAPH NEEDS TO BE MAINTAINED



Architectural Decision Record (ADR):

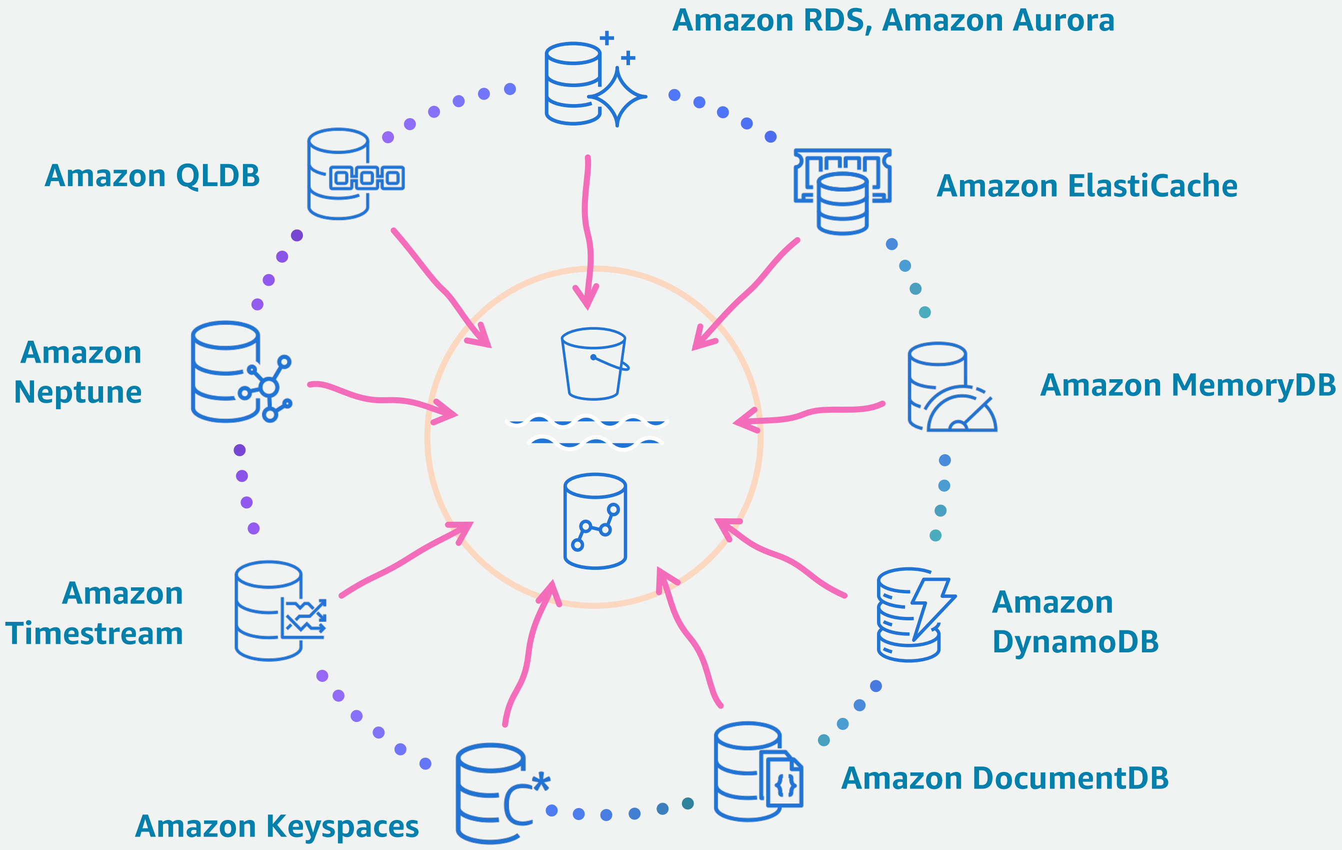
Context: The AI/ML system requires a identity graph which allows to traverse the relationships between

Alternatives: Amazon Neptune, Amazon Aurora

Decision: Neptune is purpose built for graphs. The provided property graph model is compatibel with the AI/ML system. Neptune treats relationships as “first class citizens” and is a better fit than a complex, hard to scale relational datamodel.

Consequences: Integration with DynamoDB based on streaming modifications to profile and catalog can be used to populate part of the information.

A Modern Data Strategy



Crosscutting Concern

Modern Data Strategy implies that each and every services provides its “siloed” data also as a Data Product (see Data Mesh).

While its an anti pattern to integrate on database level services must consider to have an interface including the right quality requirements (accuracy, completeness, reliability, relevance, timeliness) to provide data to the corporate Data Lake / Data Warehouse

Purpose-built databases

ACCESS PATTERN DRIVE DECISIONS

	Relational	**Key-value**	**In-memory**	**Graph**	**Document**	**Time-series**	**Ledger**	**Wide Column**
	Referential integrity, ACID transactions, schema-on-write	High throughput, Low latency reads and writes, endless scale	Query by key with microsecond latency	Quickly and easily create and navigate relationships between data	Store documents and quickly access querying on any attribute	Collect, store, and process data sequenced by time	Complete, immutable, and verifiable history of all changes to application data	Scalable, highly available, and managed Apache Cassandra-compatible service
AWS Service(s)	Aurora RDS	DynamoDB	ElastiCache	Neptune	DocumentDB	Timestream	QLDB	Keyspaces Managed Cassandra
Common Use Cases	ERP, CRM, Finance, DevOps	Shopping cart, product catalog, customer preferences	Leaderboards, real-time analytics, caching	Fraud detection, social networking, recommendation engine	Content management, personalization, mobile	IoT applications, event tracking, analytics on measurements	supply chain, health care, registrations, financial	Build low-latency applications, leverage open source, migrate Cassandra to the cloud

Thank you!

Andreas Juffinger

in https://www.linkedin.com/in/andreasjuffinger/