Database Fundamentals

Databases Primer

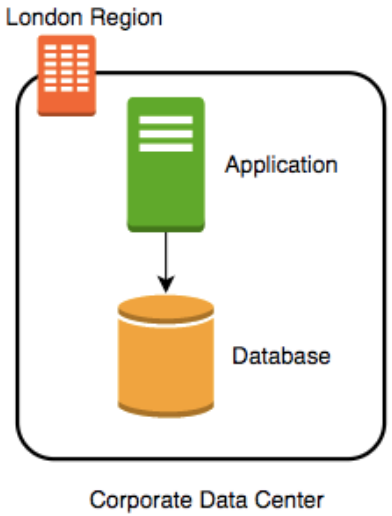
In28 Minutes

- Databases provide **organized** and **persistent** storage for your data
- To **choose between different database types**, we would need to understand:
  - Availability
  - Durability
  - RTO
  - RPO
  - Consistency
  - Transactions etc
- Let's get started on a **simple journey** to understand these

Database

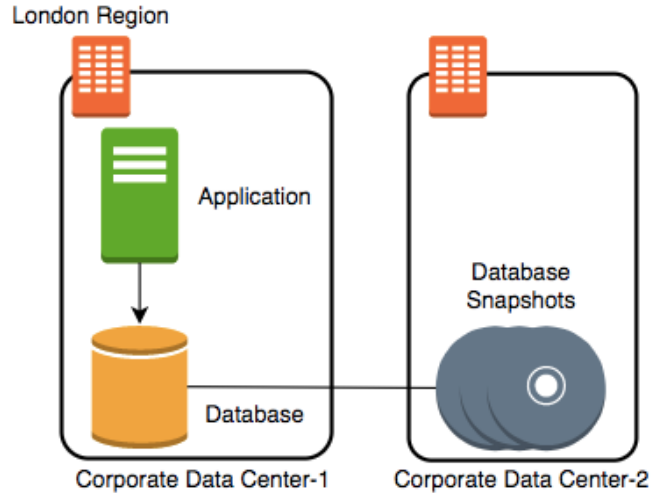
Database - Getting Started

- Imagine a database deployed **in a data center in London**
- Let's consider some challenges:
  - **Challenge 1**: Your database will go down if the data center crashes or the server storage fails
  - **Challenge 2**: You will lose data if the database crashes



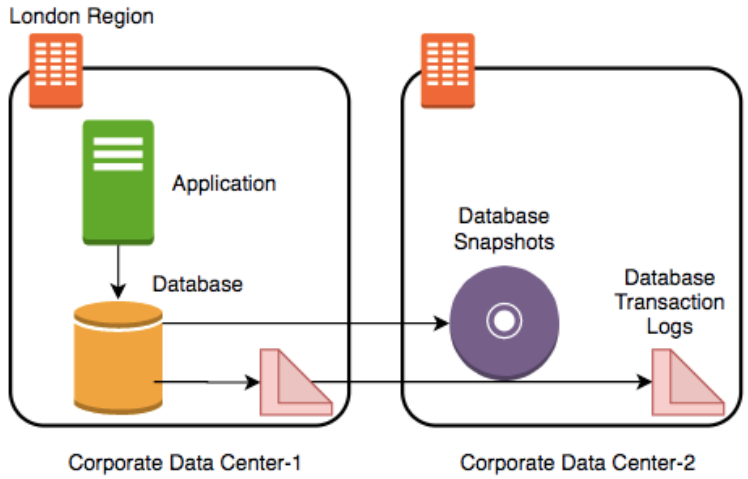
Database - Snapshots

- Let's automate taking copy of the database (**take a snapshot**) every hour to another data center in London
- Let's consider some challenges:
  - **Challenge 1**: Your database will go down if the data center crashes
  - **Challenge 2** (PARTIALLY SOLVED): You will lose data if the database crashes
    - You can setup database from latest snapshot. But depending on when failure occurs you can lose up to an hour of data
  - **Challenge 3**(NEW): Database will be slow when you take snapshots



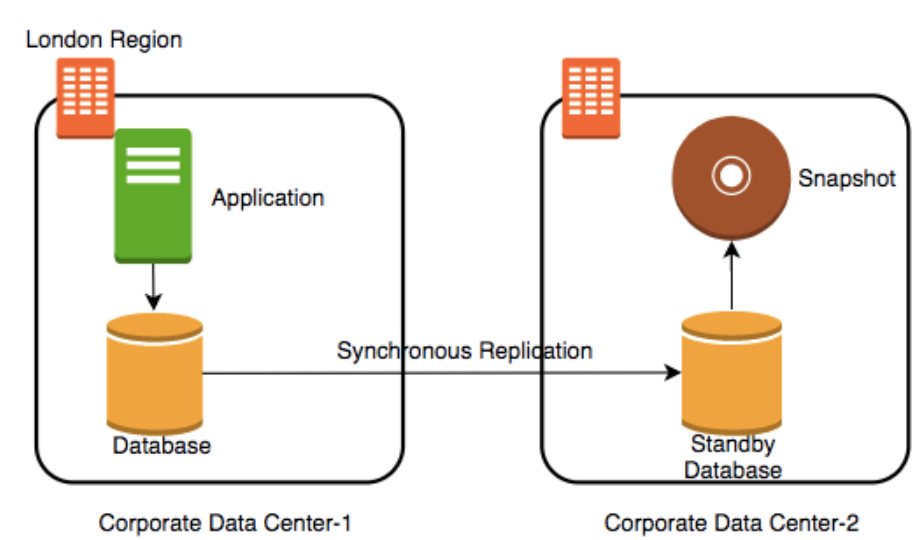
Database - Transaction Logs

- Let's add **transaction logs** to database and create a **process to copy it over** to the second data center
- Let's consider some challenges:
  - **Challenge 1**: Your database will go down if the data center crashes
  - **Challenge 2** (SOLVED): You will lose data if the database crashes
    - You can setup database from latest snapshot and apply transaction logs
  - **Challenge 3**: Database will be slow when you take snapshots



Database - Add a Standby

- Let's add a **standby database** in the second data center with replication
- Let's consider some challenges:
  - **Challenge 1** (SOLVED): Your database will go down if the data center crashes
    - You can switch to the standby database
  - **Challenge 2** (SOLVED): You will lose data if the database crashes
  - **Challenge 3** (SOLVED): Database will be slow when you take snapshots
    - Take snapshots from standby.
    - Applications connecting to master will get good performance always



Availability and Durability

In 28 Minutes

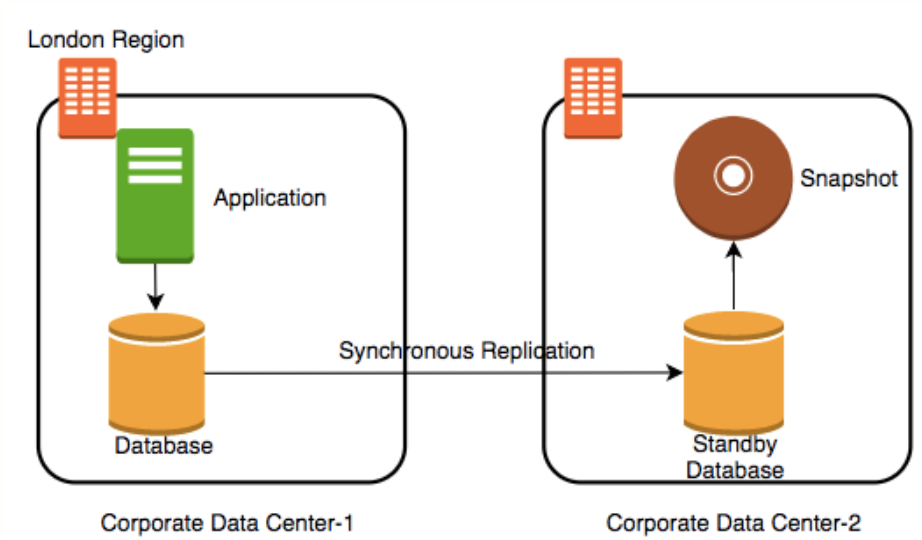
- **Availability**
  - Will I be able to access my data now and when I need it?
  - Percentage of time an application provides the operations expected of it
- **Durability**
  - Will my data be available after 10 or 100 or 1000 years?
- Examples of measuring availability and durability:
  - 4 9's - 99.99
  - 11 9's - 99.999999999
- Typically, an **availability of four 9's** is considered very good
- Typically, a **durability of eleven 9's** is considered very good

Availability

Availability	Downtime (in a month)	Comment
99.95%	22 minutes	
99.99% (4 9's)	4 and 1/2 minutes	Typically online apps aim for 99.99% (4 9's) availability
99.999% (5 9's)	26 seconds	Achieving 5 9's availability is tough

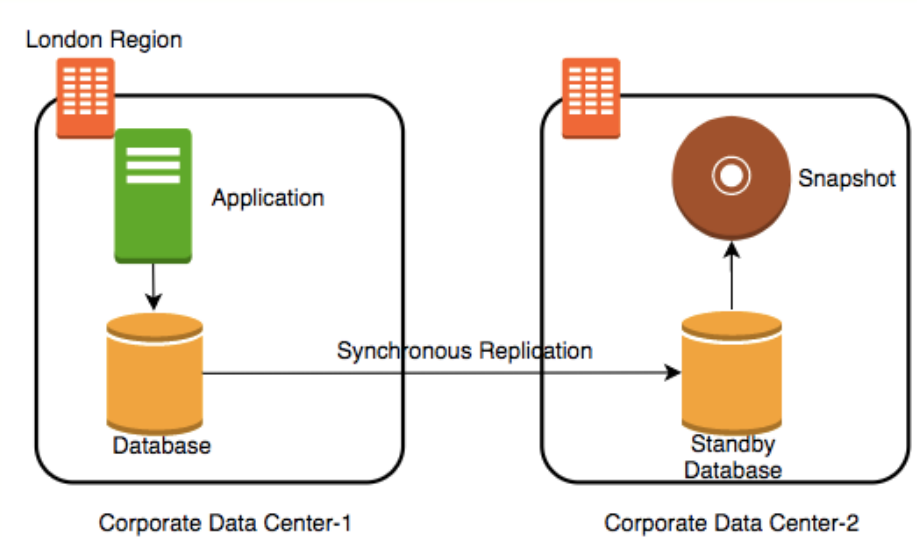
Durability

- What does a **durability of 11 9's** mean?
  - If you **store one million files for ten million years**, you would expect to **lose one file**
- Why should durability be high?
  - Because **we hate losing data**
  - Once we lose data, it is gone



Increasing Availability and Durability of Databases

- **Increasing Availability**:
  - Have multiple standbys available OR distribute the database
    - in multiple Zones
    - in multiple Regions
- **Increasing Durability**:
  - Multiple copies of data (standbys, snapshots, transaction logs and replicas)
    - in multiple Zones
    - in multiple Regions
- **Replicating data** comes with its own challenges!
  - We will talk about them a little later



Database Terminology : RTO and RPO

In 28 Minutes

- Imagine a **financial transaction being lost**
- Imagine a **trade being lost**
- Imagine a **stock exchange going down** for an hour
- **Typically** businesses are fine with some downtime but they hate losing data
- Availability and Durability are technical measures
- How do we measure **how quickly we can recover from failure?**
  - **RPO (Recovery Point Objective)**: Maximum acceptable period of data loss
  - **RTO (Recovery Time Objective)**: Maximum acceptable downtime
- Achieving **minimum RTO and RPO is expensive**
- **Trade-off** based on the criticality of the data

Database

Question - RTO and RPO

In 28 Minutes

- You are running an application in VM instance storing its data on a persistent data storage. You are taking snapshots every 48 hours. If the VM instance crashes, you can manually bring it back up in 45 minutes from the snapshot. What is your RTO and RPO?
  - RTO - 45 minutes
  - RPO - 48 hours

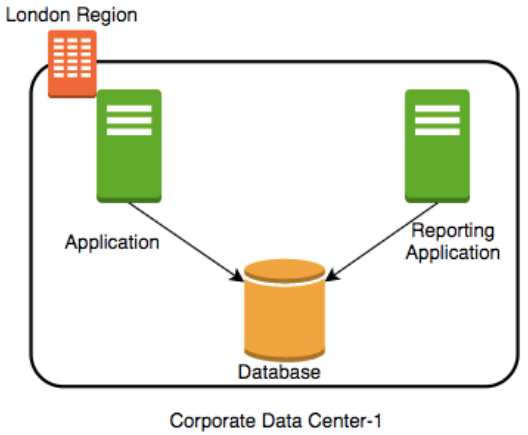
Achieving RTO and RPO - Failover Examples

In 28 Minutes

Scenario	Solution
Very small data loss (RPO - 1 minute) **Very small downtime (RTO - 5 minutes)**	**Hot standby -** Automatically synchronize data Have a standby ready to pick up load Use automatic failover from master to standby
Very small data loss (RPO - 1 minute) **BUT I can tolerate some downtimes (RTO - 15 minutes)**	**Warm standby** - Automatically synchronize data Have a standby with minimum infrastructure Scale it up when a failure happens
Data is critical (RPO - 1 minute) but I can tolerate downtime of a few hours (RTO - few hours)	Create regular data **snapshots and transaction logs** Create database from snapshots and transactions logs when a failure happens
Data can be lost without a problem (for example: cached data)	Failover to a completely new server

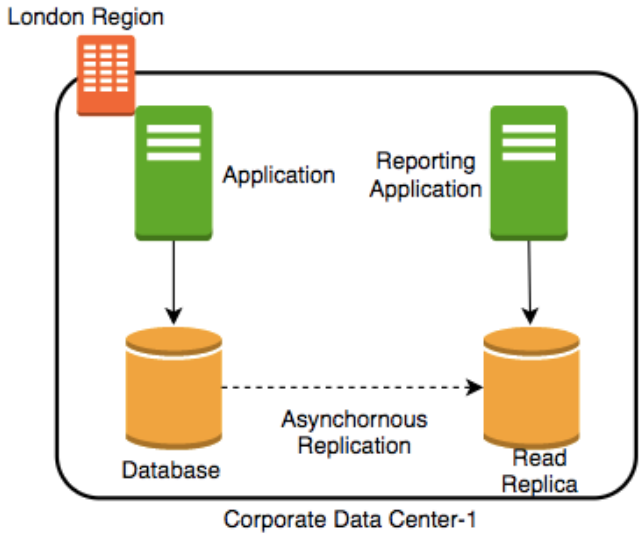
(New Scenario) Reporting and Analytics Applications

- New reporting and analytics applications are being launched using the same database
  - These applications will ONLY read data
- Within a few days you see that the database performance is impacted
- How can we fix the problem?
  - **Vertically scale the database** - increase CPU and memory
  - **Create a database cluster (Distribute the database)** - Typically database clusters are expensive to setup
  - **Create read replicas** - Run read only applications against read replicas



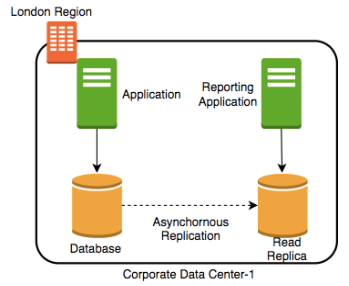
Database - Read Replicas

- Add **read replica**
- Connect reporting and analytics applications to **read replica**
- Reduces load on the master databases
- Upgrade read replica to master database (supported by some databases)
- Create read replicas **in multiple regions**
- **Take snapshots** from read replicas



Consistency

- How do you ensure that data in multiple database instances (standbys and replicas) is updated simultaneously?
- **Strong consistency** - Synchronous replication to all replicas
  - Will be slow if you have multiple replicas or standbys
- **Eventual consistency** - Asynchronous replication. A little lag - few seconds - before the change is available in all replicas
  - In the intermediate period, different replicas might return different values
  - Used when scalability is more important than data integrity
  - Examples : Social Media Posts - Facebook status messages, Twitter tweets, Linked in posts etc
- **Read-after-Write consistency** - Inserts are immediately available
  - However, updates would have eventual consistency



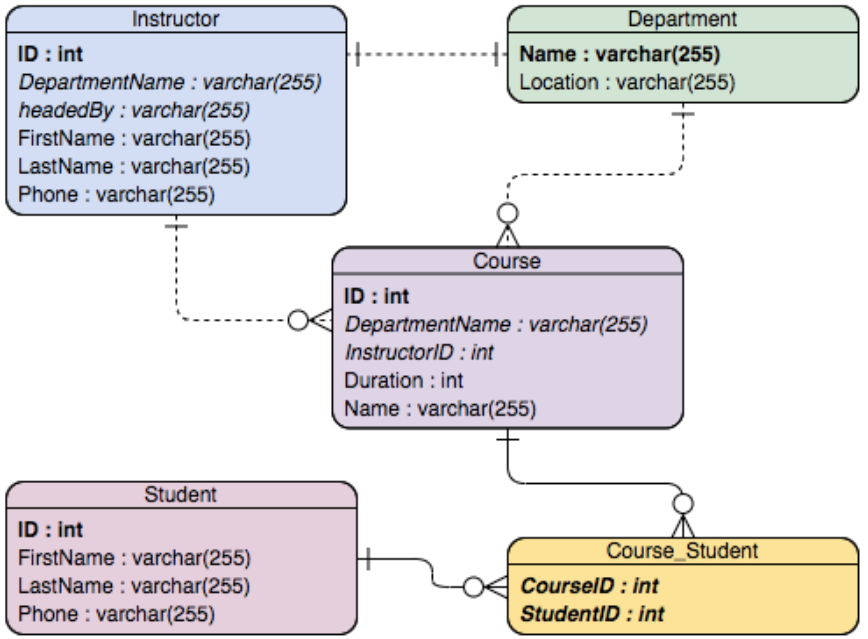
Database Categories

In 28 Minutes

- There are **several categories** of databases:
  - Relational (OLTP and OLAP), Document, Key Value, Graph, In Memory among others
- **Choosing type of database** for your use case is not easy. A few factors:
  - Do you want a **fixed schema**?
    - Do you want flexibility in defining and changing your schema? (schemaless)
  - What level of **transaction properties** do you need? (atomicity and consistency)
  - What kind of **latency** do you want? (seconds, milliseconds or microseconds)
  - **How many transactions** do you expect? (hundreds or thousands or millions of transactions per second)
  - **How much data** will be stored? (MBs or GBs or TBs or PBs)
  - and a lot more...

Relational Databases

- This was the **only option** until a decade back!
- Most **popular (or unpopular)** type of databases
- **Predefined schema** with tables and relationships
- Very **strong transactional** capabilities
- Used for
  - OLTP (Online Transaction Processing) use cases and
  - OLAP (Online Analytics Processing) use cases



Relational Database - OLTP (Online Transaction Processing)

In 28 Minutes

- Applications where **large number of users make large number of small transactions**
  - small data reads, updates and deletes
- **Use cases:**
  - Most traditional applications, ERP, CRM, e-commerce, banking applications
- **Popular databases:**
  - MySQL, Oracle, SQL Server etc
- Recommended Google Managed Services:
  - **Cloud SQL** : Supports PostgreSQL, MySQL, and SQL Server for regional relational databases (upto a few TBs)
  - **Cloud Spanner**: Unlimited scale (multiple PBs) and 99.999% availability for global applications with horizontal scaling

Cloud SQL

Cloud Spanner

Relational Database - OLAP (Online Analytics Processing)

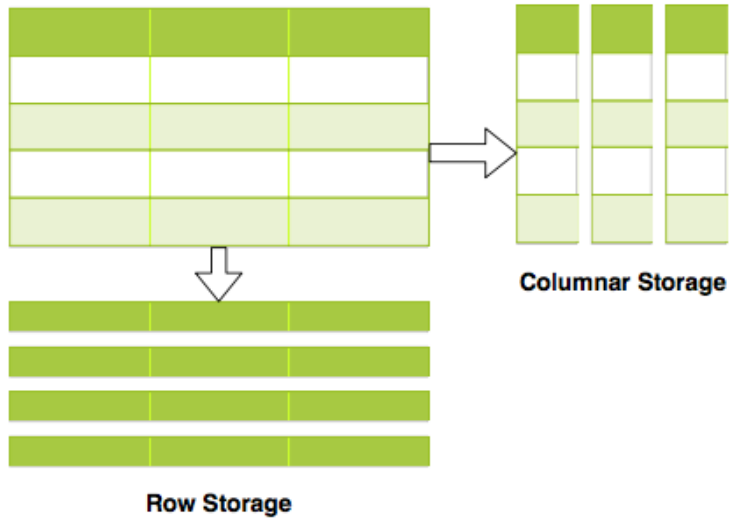
In 28 Minutes

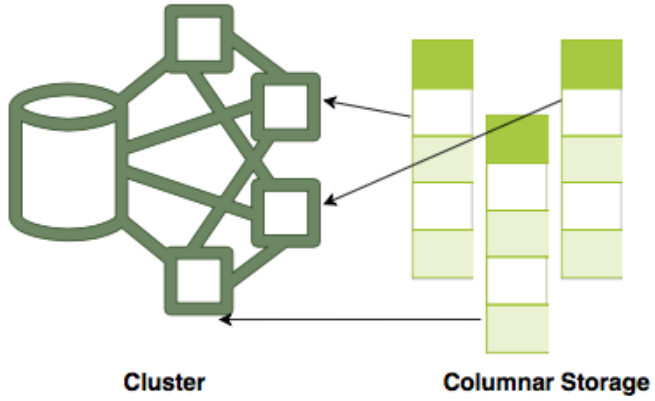
- Applications allowing users to **analyze petabytes of data**
  - **Examples** : Reporting applications, Data ware houses, Business intelligence applications, Analytics systems
  - **Sample application** : Decide insurance premiums analyzing data from last hundred years
  - Data is consolidated from multiple (transactional) databases
- Recommended GCP Managed Service
  - **BigQuery**: **Petabyte-scale** distributed data ware house

Relational Databases - OLAP vs OLTP

In 28 Minutes

- OLAP and OLTP use **similar data structures**
- BUT **very different approach in how data is stored**
- **OLTP databases** use row storage
  - Each table row is stored together
  - Efficient for processing small transactions
- **OLAP databases** use columnar storage
  - Each table column is stored together
  - **High compression** - store petabytes of data efficiently
  - **Distribute data** - one table in multiple cluster nodes
  - **Execute single query across multiple nodes** - Complex queries can be executed efficiently





NoSQL Databases

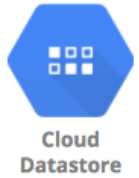
In28 Minutes

- **New approach** (actually NOT so new!) to building your databases
  - NoSQL = not only SQL
  - Flexible schema
    - Structure data **the way your application needs it**
    - Let the schema evolve with time
  - Horizontally scale to petabytes of data with millions of TPS
- **NOT a 100% accurate generalization** but a great starting point:
  - Typical NoSQL databases trade-off "Strong consistency and SQL features" to achieve "scalability and high-performance"
- **Google Managed Services**:
  - Cloud Firestore (Datastore)
  - Cloud BigTable

Cloud Firestore (Datastore) vs Cloud BigTable

In 28 Minutes

- **Cloud Datastore** - Managed serverless NoSQL document database
  - Provides ACID transactions, SQL-like queries, indexes
    - Designed for transactional mobile and web applications
  - Firestore (next version of Datastore) adds:
    - Strong consistency
    - Mobile and Web client libraries
  - Recommended for small to medium databases (0 to a few Terabytes)
- **Cloud BigTable** - Managed, scalable NoSQL wide column database
  - NOT serverless (You need to create instances)
  - Recommend for data size > 10 Terabytes to several Petabytes
  - Recommended for large analytical and operational workloads:
    - NOT recommended for transactional workloads (Does NOT support multi row transactions - supports ONLY Single-row transactions)



In-memory Databases

In 28 Minutes

- **Retrieving data from memory is much faster than retrieving data from disk**
- In-memory databases like Redis deliver microsecond latency by storing **persistent data in memory**
- Recommended GCP Managed Service
  - **Memory Store**
- **Use cases** : Caching, session management, gaming leader boards, geospatial applications

Memorystore

Databases - Summary

In 28 Minutes

Database Type	GCP Services	Description
Relational OLTP databases	Cloud SQL, Cloud Spanner	Transactional usecases needing **predefined schema** and very **strong transactional** capabilities (Row storage) **Cloud SQL**: MySQL, PostgreSQL, SQL server DBs **Cloud Spanner**: Unlimited scale and 99.999% availability for global applications with horizontal scaling
Relational OLAP databases	BigQuery	Columnar storage with predefined schema. Datawarehousing & BigData workloads
NoSQL Databases	Cloud Firestore (Datastore) , Cloud BigTable	Apps needing **quickly evolving** structure (**schema-less**) **Cloud Firestore** - Serverless transactional document DB supporting mobile & web apps. Small to medium DBs (0 - few TBs) **Cloud BigTable** - Large databases(10 TB - PBs). Streaming (IOT), analytical & operational workloads. NOT serverless.
In memory	Cloud Memorystore	Applications needing **microsecond** responses

Databases - Scenarios

In 28 Minutes

Scenario	Solution
A start up with quickly evolving schema (table structure)	Cloud Datastore/Firestore
Non relational db with less storage (10 GB)	Cloud Datastore
Transactional global database with predefined schema needing to process million of transactions per second	CloudSpanner
Transactional local database processing thousands of transactions per second	Cloud SQL
Cache data (from database) for a web application	MemoryStore
Database for analytics processing of petabytes of data	BigQuery
Database for storing huge volumes stream data from IOT devices	BigTable
Database for storing huge streams of time series data	BigTable