Remote Databases and Firebase

Efficient Data Handling and Integration

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#QA-Auto-FrontEnd

Table of Contents

Remote Databases

Introduction

What is Remote Database?

- Centralized **storage system accessed over a network**, enabling users to **store and retrieve data** from locations **separate** from **where** the **data is used or generated**
- **Data processing flow**:
  - User sends a **request** to the **application server**
  - **Application server interacts with the remote database**
  - Remote database **processes the request** and sends a **response back to the application server, which then forwards it to the user**

Remote Database Capabilities

- **Centralized storage**: Data is stored in a single location accessible from multiple locations or devices
- **Remote access**: Users can access the database over a network, such as the internet, from anywhere
- **Data synchronization**: Changes made to the database are synchronized across all connected devices or locations in real-time

Remote Database Capabilities

- **Scalability**: Remote databases can easily scale to accommodate growing amounts of data and users
- **Security**: Remote databases often implement robust security measures to protect data from unauthorized access or breaches
- **Collaboration**: Multiple users can access and work with the same dataset concurrently, facilitating collaboration among teams or users in different locations

Local Database vs. Remote Database

- Pros of **Local Database:**
  - Faster access
  - Greater control over privacy
  - Works well offline
  - Lower dependency on network connectivity
  - Simplified data management on a single device

- Pros of **Remote Database:**
  - Centralized access from anywhere
  - Real-time collaboration among multiple users
  - Scalability to handle large datasets and user bases
  - Automatic data synchronization across devices
  - Enhanced security measures and backup options

Local Database vs. Remote Database

- Cons of **Local Database**:
  - Limited accessibility from other devices
  - Dependency on device storage capacity
  - Difficulty in syncing data across multiple devices
  - Data loss risk if the device is damaged or lost
  - Challenges in sharing data among multiple users

- Cons of **Remote Database:**
  - Reliance on network connectivity; disruptions can affect access
  - Potential security vulnerabilities due to remote access
  - Dependency on third-party service providers
  - Costs associated with data storage and usage
  - Compliance concerns regarding data jurisdiction and regulations

Remote Databases Types

Based on Architecture, Accessibility, Data organization

By Architecture and Accessibility

- **Cloud-based databases**:
  - Hosted on cloud platforms
  - Provide scalable and flexible storage solutions
  - Accessible from anywhere with an internet connection
- **Hosted databases**:
  - Managed databases provided by third-party service providers
  - Offer remote access and administration without the need for maintaining infrastructure

By Architecture and Accessibility

- **Database as a Service (DBaaS)**:
  - Fully managed database services
  - Handle administrative tasks like backups, updates, and scaling
  - Allow users to focus on application development
- **Replicated databases**:
  - Databases with multiple synchronized copies
  - Distributed across different locations or servers
  - Ensure data availability and redundancy for disaster recovery and high availability

By Data Organization

- Remote databases can be characterized **based on their structured organization of data** as:
  - **Relational Databases**
  - **NoSQL Databases**
  - **Key-Value Stores**
  - **Document-Oriented Databases**
  - **Graph Databases**

By Data Organization

- **Relational Databases**:
  - Structured **data organized in tables** with predefined schemas
  - Utilizes **SQL for querying** and manipulation
  - Suitable for **transactional systems** and **complex queries**
- **NoSQL Databases**:
  - **Schema-less or flexible schema** structure
  - Designed for **scalability, performance, handling unstructured data**
  - Common types include **document, key-value, column-family, and graph databases**

By Data Organization

- **Key-Value Stores**:
  - Simple data model consisting of **key-value pairs**
  - Fast and scalable for **high-volume read/write operations**
  - **Ideal for caching, session management, and user preferences**
- **Document-Oriented Databases**:
  - Stores data in **flexible, JSON-like documents**
  - **Supports nested structures and dynamic schemas**
  - Well-suited for **content management, blogging platforms, and real-time analytics**

By Data Organization

- **Graph Databases**:
  - Data stored in **nodes and edges to represent relationships**
  - Enables **efficient querying of complex relationships and networks**
  - Useful for **social networks, recommendation systems, and fraud detection**

Performance and Scalability

Quick and efficiently data processing

Database Performance

- Refers to **how quickly** and **efficiently** the **system** can **respond to requests** and **process data**
- Includes factors such as:
  - **Response time**
  - **Throughput**
  - **Latency**
  - **Resource utilization**
- Achieving good performance involves **optimizing** various aspects of the database system, including **hardware**, **software**, and **data model**

Optimizing Performance

- To **improve database performance**, consider implementing these strategies:
  - **Optimized Queries**: Write **efficient queries** that **retrieve only the necessary data**. Use **appropriate indexes to speed up data retrieval**
  - **Data Modeling**: Design an **efficient data model** that minimizes redundant data and maximizes query performance
  - **Indexing**: Create **indexes on frequently queried columns** to speed up data retrieval operations

Optimizing Performance

- **Caching**: Implement **caching mechanisms** to **store frequently accessed data in memory**, reducing the need for repeated database queries
- **Hardware Optimization**: Use **high-performance hardware components** such as **fast storage devices, sufficient memory,** and **powerful processors** to improve database performance
- **Query Optimization**: Analyze **query execution plans** and optimize them using techniques such as **query rewriting, join optimization,** and **parallel processing**

Database Scalability

- Refers to **the ability of a system to handle increasing workload by adding resources or expanding its capacity** without significantly impacting performance
- **Two main types** of scalability:
  - **Vertical Scalability**
  - **Horizontal Scalability**

Vertical Scalability

- Known as "**Scaling up**"
- Involves **increasing the capacity of individual hardware components**
- For example, upgrading to a more powerful server with:
  - **Higher CPU**
  - **Memory**
  - **Storage capacity**
- Vertical scalability **has its limits and can become expensive or impractical beyond a certain point**

Horizontal Scalability

- Also known as "**Scaling out**"
- Involves **adding more instances of the database system across multiple servers or nodes**
- Horizontal scalability allows **the workload to be distributed across multiple machines**, thereby **increasing overall capacity and performance**
- Often **involves techniques** such as**:**
  - **Sharding**
  - **Replication**
  - **Load balancing**

DB Scalability Strategies

Software University

- Designing the database system to **support scalability** involves **structuring** it for **both horizontal** and **vertical expansion**:
  - **Horizontal Scaling**
    - Design the database system to **support distributed architectures** (sharding data across multiple nodes, implementing replication for fault tolerance and load balancing)
  - **Vertical Scaling**
    - Choose **hardware components that can be easily upgraded (servers with expandable memory** and **storage capacity**)
    - Use technologies that support vertical scalability (**database clustering and partitioning)**

Firebase

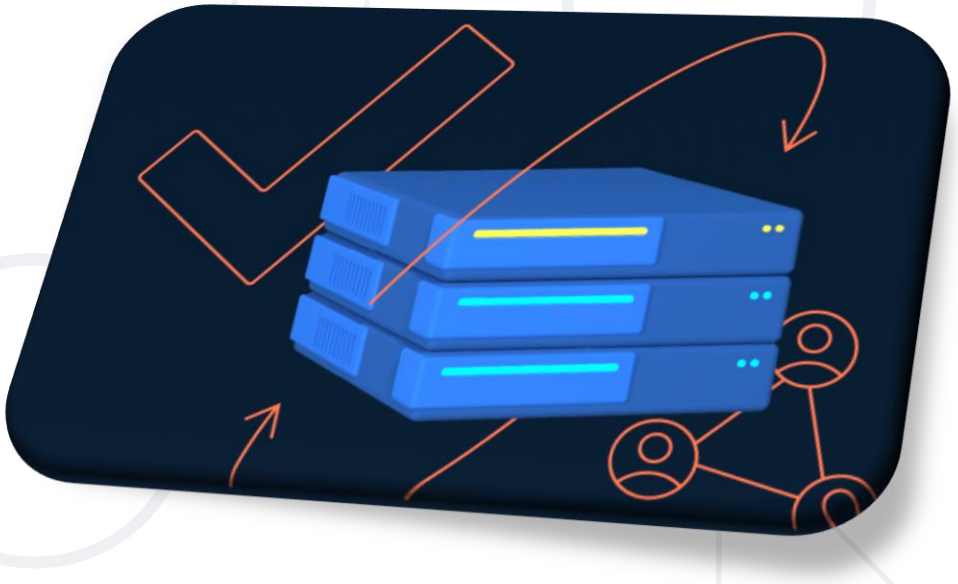
Overview

Firebase Overview

- **Comprehensive platform** developed by Google for building web and mobile applications. Provides a **wide range of tools and services**
- Offers **features** as:
  - Real-time database
  - Authentication
  - Cloud storage
  - Hosting and Analytics
- Firebase Platform

Firebase Realtime Database

- The Firebase **Realtime Database** is a **cloud-hosted**, **NoSQL database**, that:
  - Store data as JSON
  - Synchronize data in real time
  - Remains available when app goes offline



- Firebase Realtime DB - Introduction

Firebase Authentication

- Provide a **service for user sign-in authentication**
- **Two types** for its implementation:
  - **FirebaseUI Auth** - complete drop-in auth solution
  - **Firebase SDK Authentication** as:
    - Email and password based authentication
    - Federated identity provider integration
    - Phone number authentication
    - Custom auth system integration
    - Anonymous auth
- Firebase Authentication - Introduction

Cloud Storage for Firebase

- Build on **Google Cloud infrastructure** for app development
- Provide **object storage service** for:
  - Storing **images**, **audio**, **video**, or other **user-generated content**
  - **Access controls for files** using **declarative security language**
  - **Integrate** with **Firebase Authentication** to identify users

- Cloud Storage - Introduction

Firebase Hosting and Analytics

Software University

- Provides **production-grade web content hosting**
- Allows **quick deploy** of web application and serve **both static** and **dynamic content** to a global CDN (content delivery network)
- **Google Analytics integrates across Firebase features** and provides **unlimited reporting** for up to 500 distinct events that can be defined using the Firebase SDK
- Analytics reports help **understand clearly how users behave**, which enables informed decisions regarding app marketing and performance optimization
- Google Analytics for Firebase – Introduction
- Firebase Hosting – Introduction

Cloud Firestore

Strategies and Tools for Efficient Software Testing

Firestore introduction

Software University

- Cloud Firestore is a **flexible, scalable database** for mobile, web, and server development from Google
- **Part of the Firebase platform**, offering a **NoSQL, document-oriented database**
- Designed to **store** and **synchronize data** for **client-** and **server-side** development, making it **suitable for real-time applications**
- Cloud Firesore – Introduction
- Cloud Firestore – link

Cloud Firestore Key Features:

Software University

- **Scalability**: **Automatically scales** based on your application's needs, **handling large amounts of data** and **traffic without manual intervention**
- **Real-time updates**: Offers **real-time synchronization**, changes to **data on one device are immediately propagated to other connected devices**
- **Offline support**: Apps can continue to function even **when offline, syncing data automatically when connectivity is restored**
- **Security**: P**rovides security rules** that allow fine-grained control over who can access specific data and under what conditions
- **Querying**: Supports **querying and indexing data**, allowing for **efficient retrieval of information**
- **Integration**: Seamlessly **integrates with other Firebase products** (Firebase Authentication, Cloud Functions, and Cloud Storage)

Cloud Firestore vs. Realtime Database

- **Cloud Firestore:**
  - Stores data as collections of documents
  - Offline support for Apple, Android, and web clients
  - Indexed queries with compound sorting and filtering
  - Advanced write and transaction operations

- **Firebase Realtime DB:**
  - Stores data as one large JSON tree
  - Offline support for Apple and Android clients
  - Deep queries with limited sorting and filtering features
  - Basic write and transaction operations

Cloud Firestore vs. Realtime Database

- **Cloud Firestore:**
  - Regional and multi-region solution that scales automatically
  - Extremely high uptime performance
  - Scaling is automatic
  - Non-cascading rules that combine authorization and validation

- **Firebase Realtime DB:**
  - Realtime Database is a regional solution
  - High uptime performance
  - Scaling requires sharding
  - Cascading rules language that separates authorization and validation

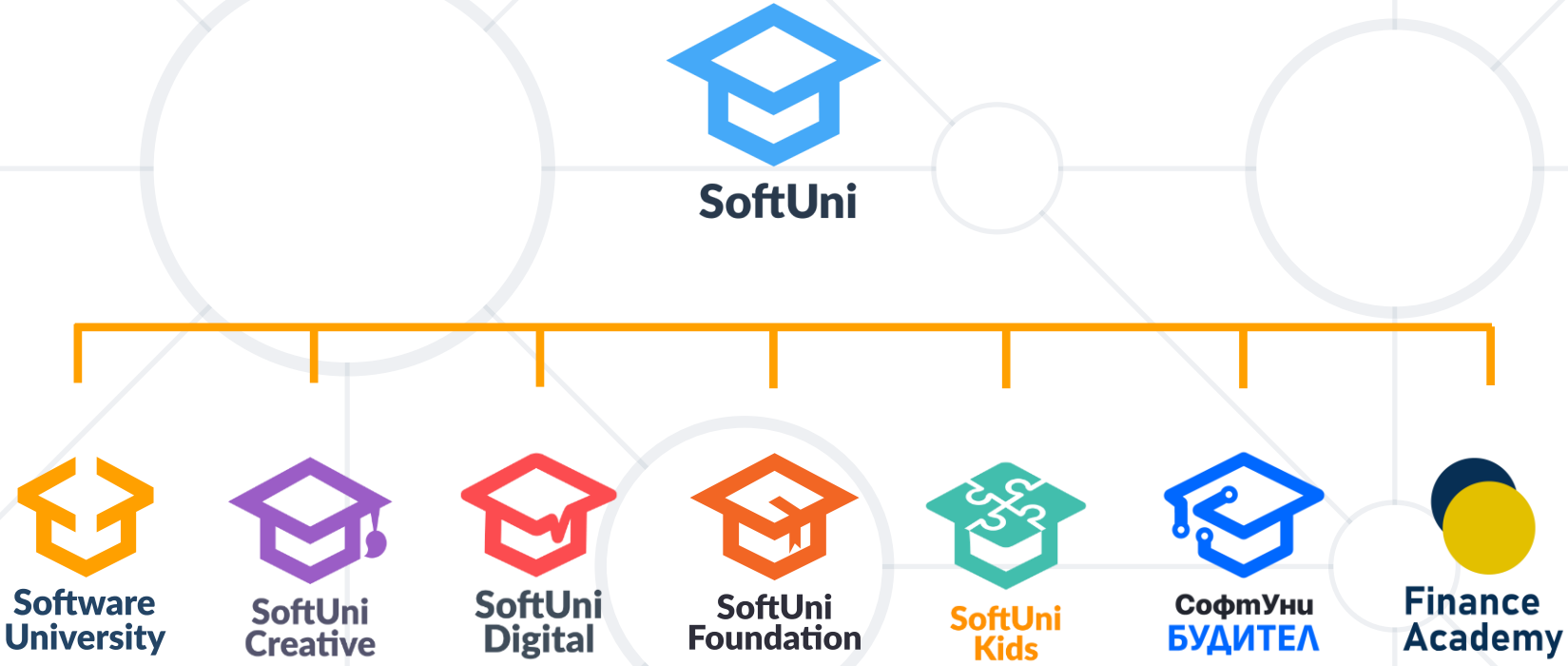
Live Demo

Firebase and Remote databases (Lab)

Summary

- What is Remote Database?
- Remote Databases capabilities
- Local DB vs. Remote DB
- What is Performance and Scalability?
- DB Performance realization
- Horizontal and Vertical Scalabilities
- Firebase platform and tools
- Cloud Firestore

Questions?



SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

Решения за твоето утре

DRAFT KINGS

VIVACOM

AMBITIONED

INDEAVR

Serving the high achievers

SOFTWARE GROUP

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg