

Database Group Project

PostgreSQL to Google Firebase

Team Weed Detector



Parth Kalkar
BS19-02



Kamil Sabbagh
BS19-02



Rafik Hachana
BS19-04



Igor Mpose
BS19-06

AGENDA

01

Introduction

What is Google firebase and its Pros and Cons.

02

Installation

How to install it and how it operates in general

03

Migration

Scripts, components and bulking time (UML Diagram included)

04

Automatization

How migration can be automatized continuously

05

Queries

Comparing queries and performance

06

CONCLUSION

Conclusion and Questions from the audience will be WELCOME

01

Introduction

What is Google firebase and its Pros and Cons.

1.1 WHAT IS FIREBASE

The **Firebase Realtime Database** is a cloud-hosted NoSQL database that lets you store and sync data between your users in real-time. Data is stored as JSON and synchronized in real-time to every connected client. When you build cross-platform apps with iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.



1.2 KEY CAPABILITIES



Realtime

It uses data synchronization



Offline

It remains responsive even when offline



Accessibility

It can be accessed directly from a mobile device or web browser



Scalability

It can split your data across multiple database instances in the same Firebase project



Google indexing

Easy Google indexing by search



AdMob

It has focused in-application promoting

1.3 FIREBASE PROS

Authentication

Easy Hosting

Ability to
Messaging

Offers Analytics

Storage facility

Crash reporting



1.4 FIREBASE CONS

Downloads all subtrees on load

Inconsistencies

Data migration

Queues

Complex Queries

Data Export



02

Installation

How to install it and how it operates in general.

2.1 Environment Preparation

As talked about in the introduction, Firebase is a **cloud-hosted NoSQL database** and it's **GUI is a web based dashboard/portal** hence, as an administrator, you don't need to install anything locally on your machine to be able to access this tool. In this part, I'll take you through a series of steps to prepare your environment.



Step 1

The screenshot shows the Firebase homepage at firebase.google.com. The main headline reads "Firebase helps you build and run successful apps". Below it, a subtext states "Backed by Google and loved by app development teams - from startups to global enterprises". At the bottom left, there are three buttons: "Get started", "Try demo", and "Watch video". A hand icon with a yellow sleeve is pointing towards the "Get started" button, which is highlighted with a dashed orange border. The background features a blue gradient with a stylized illustration of two people working on a laptop, surrounded by arrows and a rocket ship icon.

Save the date - Google I/O returns May 18-20. Register to get the most out of the digital experience: Build your schedule, reserve space, participate in Q&As, earn Google Developer profile badges, and more. [Register now](#)

Firebase

Products Use Cases Pricing Docs Community Support

Search English Go to console

Get started Try demo Watch video

Backed by Google and loved by app development teams - from startups to global enterprises

25%

Step 2

The screenshot shows the Firebase console homepage. At the top, there's a navigation bar with various links like 'Apps', 'Basic Operato...', 'User profile', etc. Below the navigation is a banner featuring a woman working on a laptop. The main area is titled 'Your Firebase projects' and shows two projects: 'Testing' (with ID 'test-8d0da') and 'Assignment-2' (with ID 'assignment-2-e7286'). A large orange hand icon points to the 'Add project' button, which is located inside a dashed box. Below the projects, there are two cards: 'Explore a demo project' (iOS) and 'Firebase projects are containers for your apps' (with a note about sharing features like Realtime Database and Analytics). At the bottom, there are language settings ('English (United Kingdom)'), support links ('Support · Terms · Privacy policy'), and a hexagonal pattern on the sides.

console.firebaseio.google.com/u/0/

Apps Best Melodic... SelfishNet win... Basic Operato... User profile Python Websi... Teach Python... About - AnkiW... Java Tutorial... Java Tutorial|... Hot Coubs - Th... Spotify - Sund...

Firebase

Your Firebase projects

+ Add project

Testing test-8d0da

Assignment-2 assignment-2-e7286

Explore a demo project

iOS

Firebase projects are containers for your apps

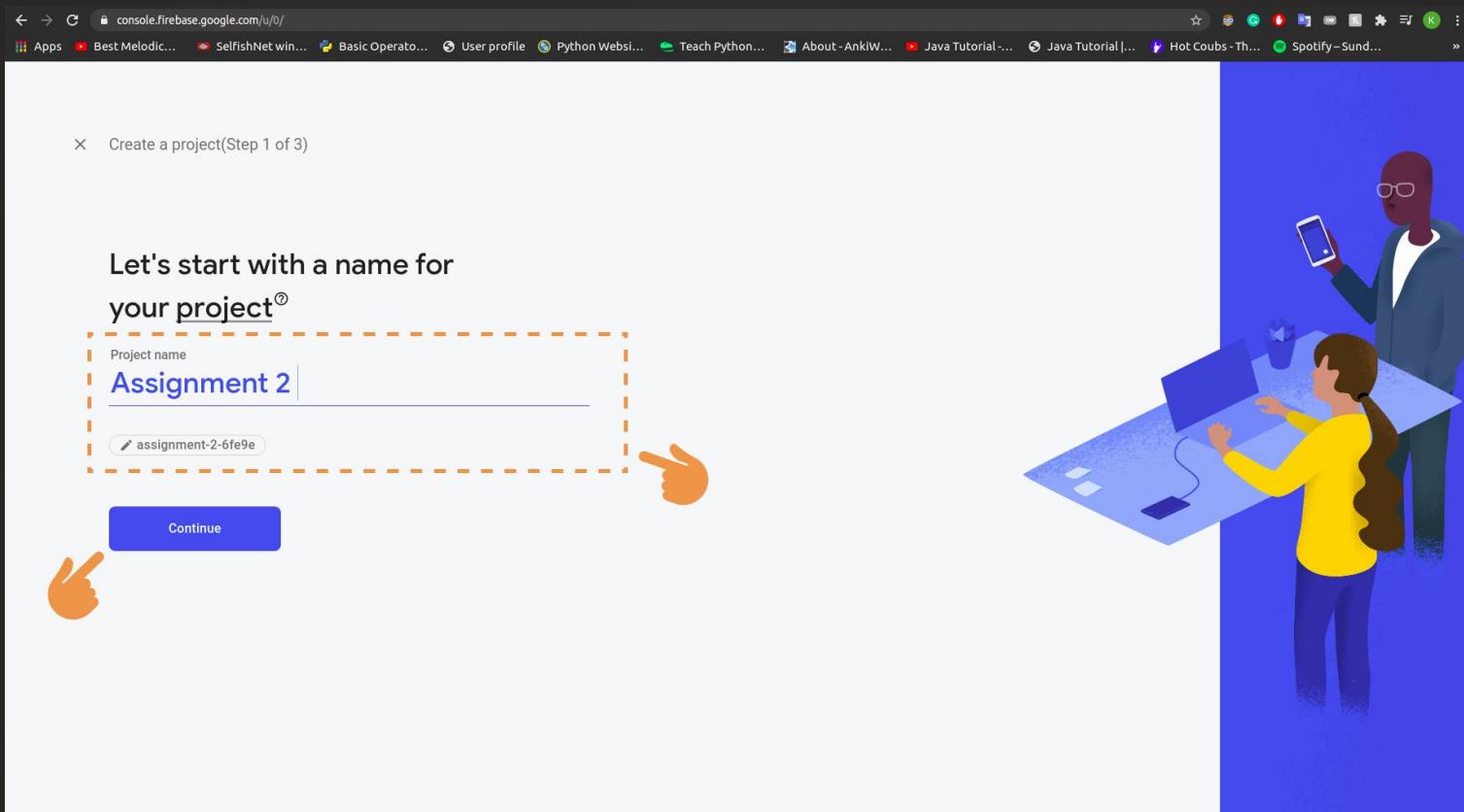
Realtime Database and Analytics

Learn more

Support · Terms · Privacy policy

Language English (United Kingdom)

Step 3



The screenshot shows a browser window for the Firebase console at `console.firebaseio.google.com/u/0/`. The title bar says "Create a project (Step 1 of 3)". The main content asks "Let's start with a name for your project®". A dashed orange border surrounds a text input field containing "Assignment 2". Below the input field is a smaller button labeled "assignment-2-6fe9e". At the bottom is a blue "Continue" button. Two orange hand icons are pointing towards the "Continue" button. To the right of the form is a stylized illustration of two people working on a computer.

× Create a project (Step 1 of 3)

Let's start with a name for your project®

Project name

Assignment 2

assignment-2-6fe9e

Continue



Step 4

← → ⌛ 🔒 console.firebaseio.google.com/u/0/

Apps Best Melodic... SelfNet win... Basic Operato... User profile Python Websi... Teach Python... About - AnkiW... Java Tutorial -... Java Tutorial |... Hot Coubs - Th... Spotify - sund...

×

Create a project(Step 2 of 3)

Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions and Cloud Functions.

Google Analytics enables:

- A/B testing ⓘ
- User segmentation and targeting ⓘ across Firebase products
- Predicting user behaviour ⓘ
- Crash-free users ⓘ
- Event-based Cloud Functions triggers
- Free unlimited reporting ⓘ

Enable Google Analytics for this project
Recommended

Previous

Continue



Step 5

← → C 🔒 console.firebaseio.google.com/u/0/

Apps Best Melodic... SelfishNet win... Basic Operato... User profile Python Websi... Teach Python... About - AnkiW... Java Tutorial -... Java Tutorial |... Hot Coubs - Th... Spotify - Sund...

X Create a project(Step 3 of 3)

Configure Google Analytics

Choose or create a Google Analytics account ⓘ

Default Account for Firebase

Automatically create a new property in this account ✎

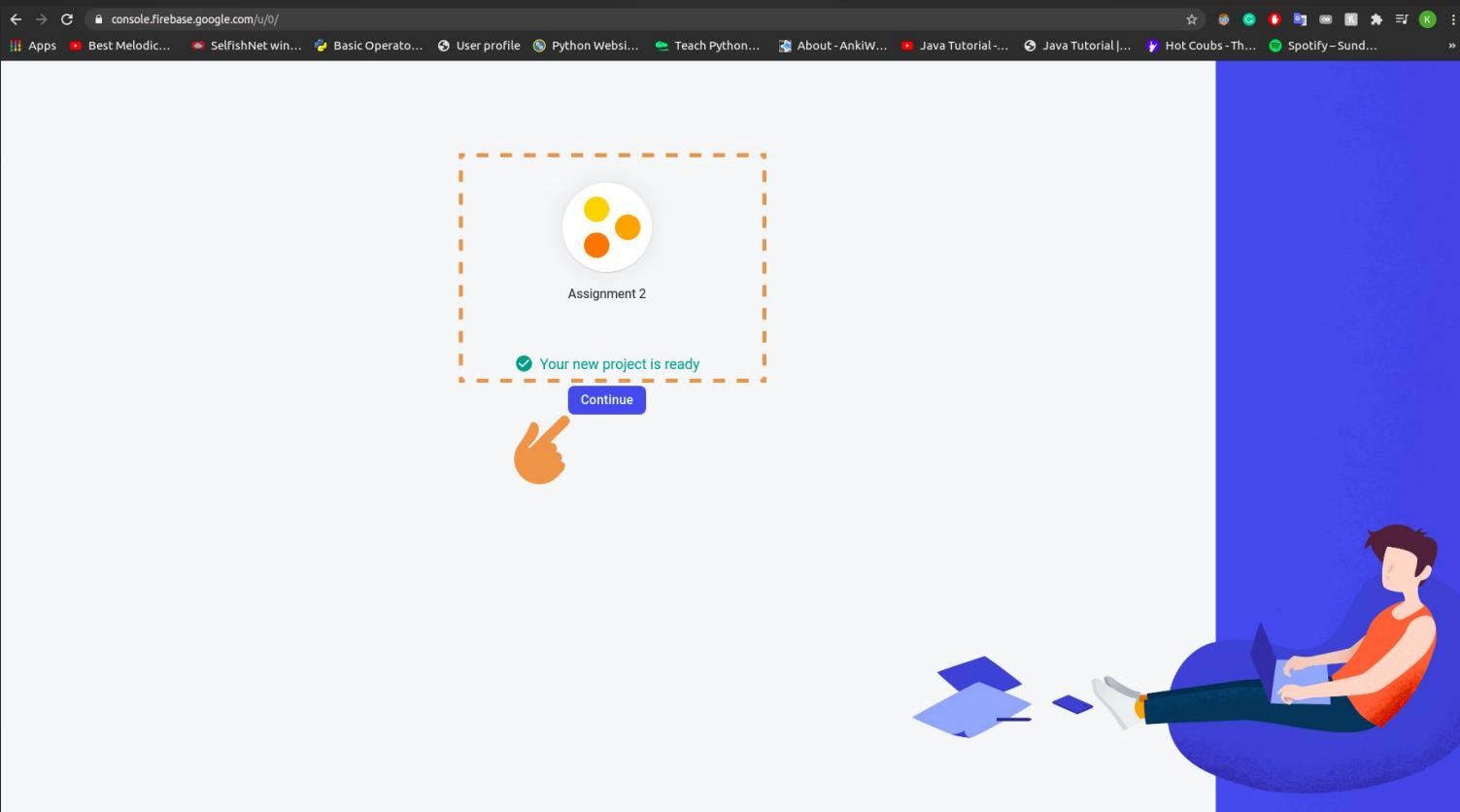
Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more.](#)

Previous Create project

A hand icon points to the 'Default Account for Firebase' dropdown. Another hand icon points to the 'Create project' button. The URL in the browser is 'console.firebaseio.google.com/u/0/'.



Step 6



Step 7

The screenshot shows the Firebase console interface. On the left, a sidebar menu titled "Build" contains sections for Authentication, Firestore, Realtime Database, Storage, Hosting, Functions, and Machine Learning. An orange hand cursor is pointing at the "Realtime Database" item, which is highlighted with a dashed orange border. The main content area is titled "Assignment 2" and features a "Spark plan". It includes a call-to-action "Get started by adding Firebase to your app" with icons for iOS, Android, web, and Cloud Functions. Below this is a button "Add an app to get started" and a large illustration of two people interacting with a smartphone. A modal window titled "Store and sync app data in milliseconds" is open at the bottom, showing two cards: "Authentication" (purple background) and "Cloud Firestore" (orange background). The "Authentication" card has the subtext "Authenticate and manage users". The "Cloud Firestore" card has the subtext "Real-time updates, powerful queries and automatic scaling".

Step 8

The screenshot shows the Firebase Realtime Database creation interface. On the left, a sidebar lists various services: Authentication, Firestore, Realtime Database (which is selected), Storage, Hosting, Functions, and Machine Learning. The main area is titled "Realtime Database" with the subtitle "Store and sync data in real time". A prominent "Create Database" button is centered within a dashed orange box. To the right, there's a diagram illustrating data synchronization between multiple servers. Below the main area, a callout box asks "Is Realtime Database right for you?" with a "Compare Databases" link. At the bottom, there are sections for "Learn more" (including links to "How do I get started?", "View the docs", and "How much will Realtime Database cost? View pricing") and a video thumbnail titled "Introducing Firebase Realtime Database" with options to "Watch lat..." and "Share".

Step 9

The screenshot shows the Firebase Realtime Database setup interface. On the left, the Firebase navigation bar includes Project Overview, Authentication, Firestore, Realtime Database (selected), Storage, Hosting, Functions, and Machine Learning. Below these are sections for Release and monitor, Analytics, Engage, and Extensions. The main area is titled "Realtime Database" with the subtitle "Store and sync data in real time". A "Create Database" button is visible. A modal window titled "Set up database" is open, showing "Database options" (selected) and "Security rules". It displays the location setting: "Your location setting is where your Realtime Database data will be stored." A dropdown menu under "Realtime Database location" is set to "United States (us-central1)". A hand cursor is hovering over the "Next" button at the bottom right of the modal. The background features a dark theme with a network diagram of interconnected nodes.

Step 10

The screenshot shows the Firebase Realtime Database setup process. The main title is "Realtime Database" with the subtitle "Store and sync data in real time". A blue header bar indicates the current step: "Set up database" (step 2 of 2). The sub-step is "Security rules".

Text: "Once you have defined your data structure you will have to write rules to secure your data." followed by a "Learn more" link.

Two options are shown:

- Start in **locked mode**: "Your data is private by default. Client read/write access will only be granted as specified by your security rules."
- Start in **Test Mode**: "Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access."

A code snippet for Test Mode security rules is displayed:

```
{  
  "rules": {  
    ".read": "now < 1621630800000", // 2021-5-22  
    ".write": "now < 1621630800000", // 2021-5-22  
  }  
}
```

An orange callout box contains the note: "The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days".

At the bottom right of the dialog is a button labeled "Enable" with a blue background and white text, surrounded by a dashed orange border. A hand icon is pointing at this button.

Below the dialog, there is a section titled "How much will Realtime Database cost?" with a "View pricing" link. To the right of this is a "Watch on YouTube" button.

The overall background of the Firebase interface shows a network of servers connected by lines, and the left sidebar lists various Firebase services like Authentication, Firestore, and Storage.

Step 11

The screenshot shows the Firebase Realtime Database console. On the left, the Project Overview sidebar is visible, featuring a navigation menu with options like Authentication, Firestore, Realtime Database (which is selected and highlighted with a dashed orange border), Storage, Hosting, Functions, Machine Learning, Release and monitor, Analytics, Engage, and Extensions. A hand cursor is pointing at the Realtime Database icon in the sidebar. The main content area is titled "Realtime Database" and displays the URL <https://assignment-2-6fe9e-default.firebaseio.com/>. Below the URL, the database structure is shown as a single node named "assignment-2-6fe9e-default-rtdb" with a value of "null". At the bottom of the main area, it says "Database location: United States (us-central1)". The browser's address bar also shows the full URL.

Step 13

The screenshot shows the Firebase console interface for a project named "Assignment 2". The left sidebar contains navigation links for Project Overview, Build (Authentication, Firestore, Realtime Database, Storage, Hosting, Functions, Machine Learning), Release and monitor (Crashlytics, Performance, ...), Analytics (Dashboard, Realtime, Events), Engage (Predictions, A/B Testing, C...), and Extensions (Spark, Free \$0/month, Upgrade). The main content area displays the "Assignment 2" page with a "Spark plan" button. The central message reads: "Get started by adding Firebase to your app" and "Add an app to get started", accompanied by icons for iOS, Android, web, and Cloud Functions. Below this, a section titled "Store and sync app data in milliseconds" highlights "Authentication" (Authenticate and manage users) and "Cloud Firestore" (Real-time updates, powerful queries and automatic scaling). The URL in the browser bar is <https://console.firebaseio.google.com/u/0/project/assignment-2-6fe9e/overview>.

Step 14

x Add Firebase to your web app

1 Register app

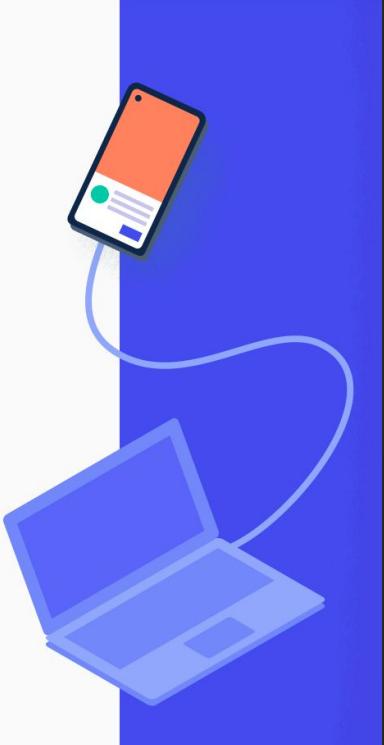
App nickname: 

Also set up **Firebase Hosting** for this app. [Learn more](#) 

Hosting can also be set up later. It's free to get started at any time.

Register app 

2 Add Firebase SDK



console.firebaseio.google.com/u/0/project/assignment-2-6f9e/overview

Apps Best Melodic... SelfishNet win... Basic Operato... User profile Python Websi... Teach Python... About - AnkiW... Java Tutorial... Java Tutorial... Hot Coubs - Th... Spotify - Sund...

Step 15

console.firebaseio.google.com/u/0/project/assignment-2-6fe9e/overview

Add Firebase to your web app

Register app

2 Add Firebase SDK

Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#available-libraries -->
<script src="https://www.gstatic.com/firebasejs/8.4.1/firebase-analytics.js"></script>

<script>
  // Your web app's Firebase configuration
  // For Firebase JS SDK v7.20.0 and later, measurementId is optional
  var firebaseConfig = {
    apiKey: "AIzaSyByqhTvVV32naMHD4W-GgE20MVpRePPzVM",
    authDomain: "assignment-2-6fe9e.firebaseio.com",
    databaseURL: "https://assignment-2-6fe9e-default-rtdb.firebaseio.com",
    projectId: "assignment-2-6fe9e",
    storageBucket: "assignment-2-6fe9e.appspot.com",
    messagingSenderId: "325906578909",
    appId: "1:325906578909:web:d0a946bcfc464562278a34",
    measurementId: "G-W6M83N53JY"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
  firebase.analytics();
</script>
```

Learn more about Firebase for web: [Get started](#), [Web SDK API reference](#), [Samples](#)

Continue to the console



2.2 Connecting with Python

"import pyrebase"

```
43 #connecting to FireBase DataBase
44 firebaseConfig = {
45     'apiKey': "AIzaSyAfSnRHC913I0_zR5LdUZCk-xZs_YFio54",
46     'authDomain': "assignment-2-e7286.firebaseioapp.com",
47     'databaseURL': "https://assignment-2-e7286-default-rtdb.firebaseio.com",
48     'projectId': "assignment-2-e7286",
49     'storageBucket': "assignment-2-e7286.appspot.com",
50     'messagingSenderId': "926718650475",
51     'appId': "1:926718650475:web:14adeb6a5d847e4594292e",
52     'measurementId': "G-GWLPMN10JZ" }
```

03

Migration

Scripts, components and bulking time (UML Diagram included)

3. MIGRATION APPROACHES

Throughout this assignment, we found out that there are two approaches of migrating data from Postgresql to Google Firebase. Those two approaches are:

1.

Naive
approach

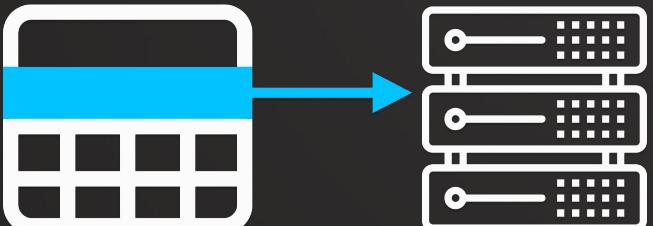
2.

Optimized
approach

3.1 NAIVE APPROACH

Installing each component (the tables in this example) from the original Database on PGadmin and uploading each row from each table in the original DB to the new one. Any entry to a new table will automatically create a new table in FireBase.

Uploading row by row



Bulking time + installation time

40 min

3.2 OPTIMIZED APPROACH

First, we install all the data from PGadmin. Each row will be represented as a dictionary that contains names of the columns and the matching data. Then all these rows will be added together forming a table also represented as a dictionary with the row index and its data. Finally, the complete table will be represented as a dictionary containing the table names and all their columns.

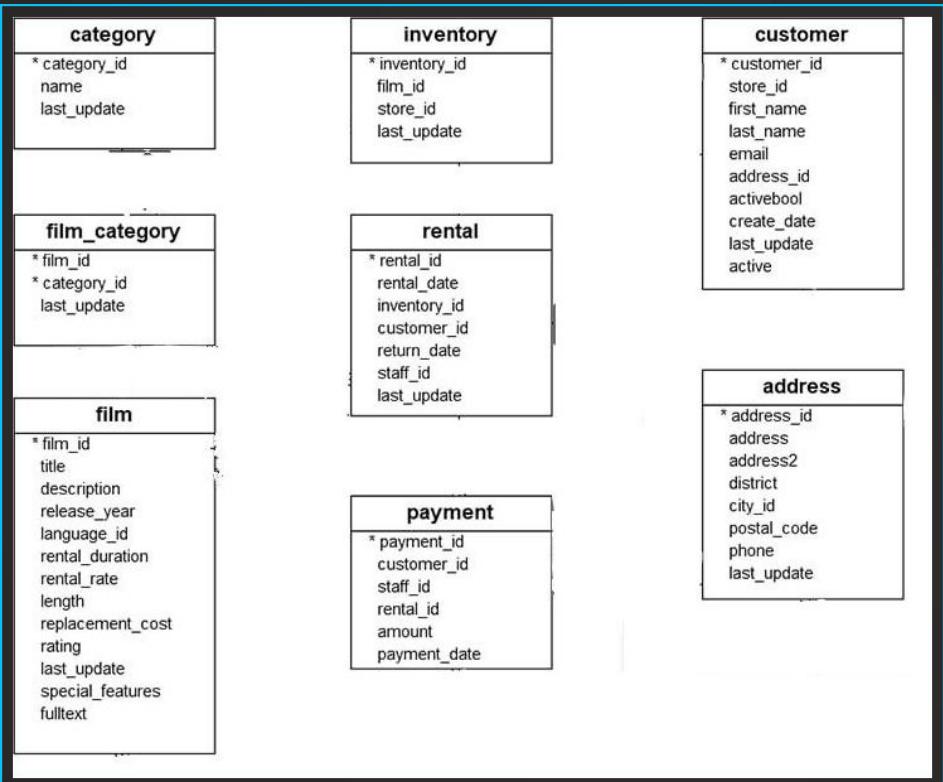
Uploading DB as a dictionary

```
DataBases{  
    table :{  
        index :{  
            column : data }}}}
```

Bulking time + installation time

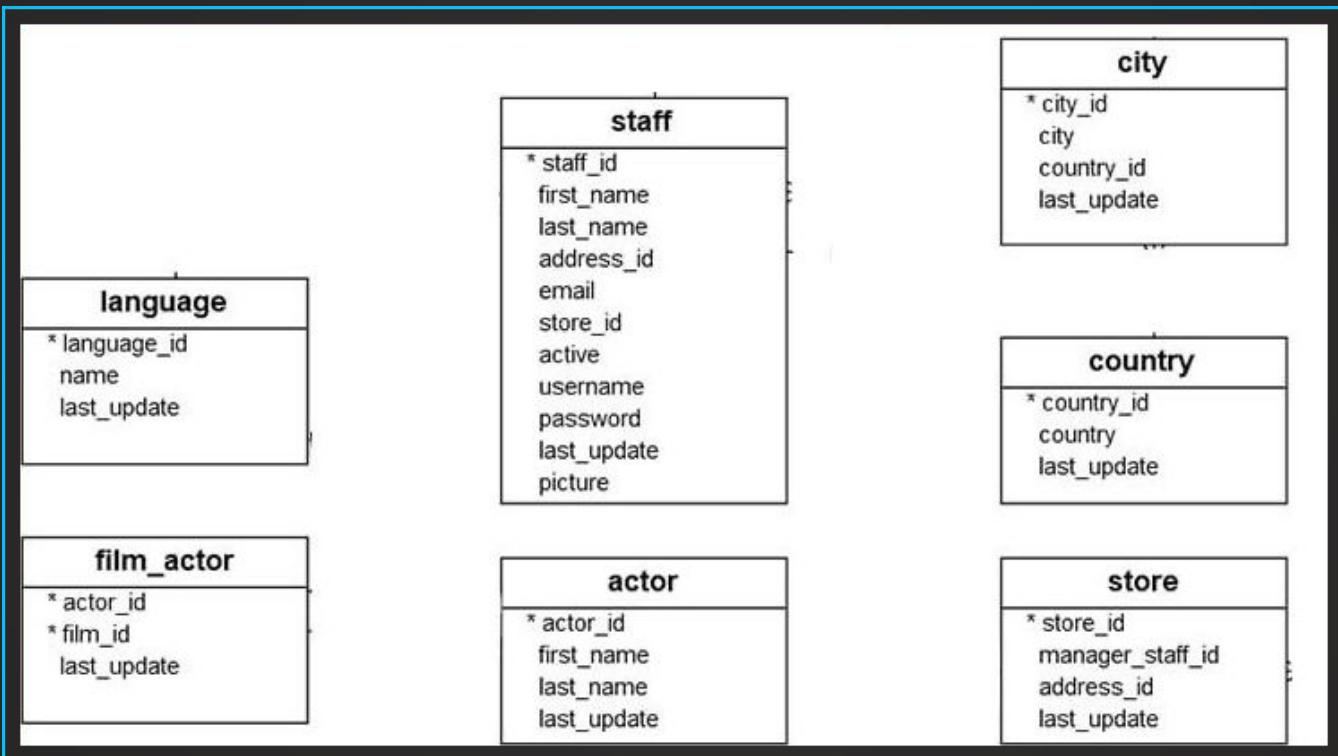
6.5 sec

3.3 UML DIAGRAM (1)



UML Diagram 1 of DVD Rental Tables in Google firebase

3.3 UML DIAGRAM (2)



UML Diagram 2 of DVD Rental Tables in Google firebase

04

Automatization

How migration can be automated continuously

4.1 Automatization in Firebase (1)

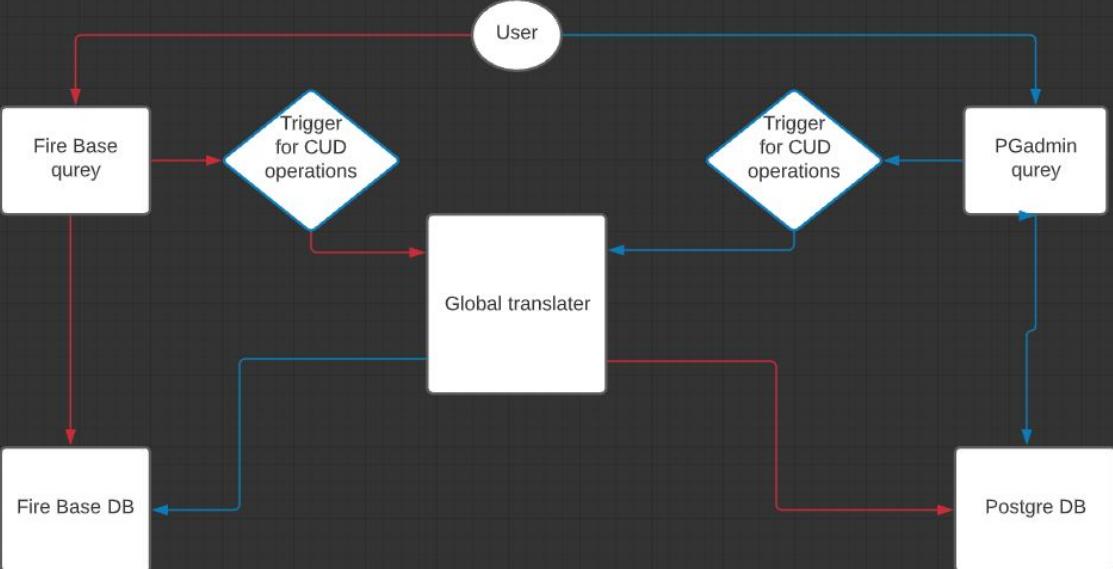
To have a real automatized migration, all the data should be completely synced with the data in PGadmin in real-time.

A user can either query to the FireBase or PGadmin. For the sake of presentation, we will check the case where a user is performing a **deleting operation** in PGadmin. As soon as the query runs, a trigger will rise. This trigger will send the query to a Global translator which will translate the Postgres query to a firebase query that will be implemented to the Firebase too.

This applies to any **Create**, **Update**, and **Delete** query.

And vice versa from any Database to another.

4.1 Automatization in Firebase (2)



Automatization process in Firebase

05

Queries

Comparing queries and performance

5.1 Query 2 (From Assignment)

Let's start simple...

Insertion, Update, Deletion

5.1.1 Insertion query (1)

Insertion Query format in Firebase:

```
db.reference(PATH).set(VALUE)
```

5.1.1 Insertion query (2)

SQL  0.044s

```
INSERT INTO actor(actor_id, first_name, last_name, last_update)  
VALUES (1000, 'John', 'Smith', '2013-05-26 14:47:57.620000');
```

5.1.1 Insertion format (3)

Firebase Realtime DB



0.849s

```
def insert_actor_with_id(actor_id, first_name, last_name, last_update):
    # Preparing the new entry
    entry = {
        'actor_id': actor_id, # Assuming the IDs are 1-indexed
        'first_name' : first_name,
        'last_name' : last_name,
        'last_update' : last_update
    }

    # The insert operation
    db.reference('actor').child(str(actor_id)).set(entry)
```

5.1.1 Insertion format (4)

Firebase Realtime DB (with auto ID)

```
def insert_actor(first_name, last_name, last_update):
    # This is for having a manual id
    all_actors = dict(db.reference("actor").get())
    size = len(all_actors)

    # Preparing the new entry
    entry = {
        'actor_id': size + 1,  # Assuming the IDs are 1-indexed
        'first_name' : first_name,
        'last_name' : last_name,
        'last_update' : last_update
    }

    # The insert operation
    db.reference('actor').child(str(size+1)).set(entry)
```

5.1.2 Selection query

Selection Query format in Firebase:

```
db.reference(PATH)  
.order_function()  
.criteria_function()
```

5.1.3 Update query(1)

SQL  0.055s

```
UPDATE staff
SET store_id = 2
WHERE store_id = 1;
```

5.1.3 Update query(2)

Firebase Realtime DB



0.472s

```
def update_store(old_store_id,new_store_id):
    print(old_store_id)
    # Getting all staff members that work in the store with id 'old_store_id'
    staff = db.reference('staff').order_by_child('store_id').equal_to(old_store_id).get()

    # Converting to the appropriate format
    staff = dict(staff)
    staff = list(staff.values())

    # Updating them one by one
    for x in staff:
        print(x)
        db.reference('staff').child(str(x['staff_id'])).child('store_id').set(new_store_id)
```

5.1.3 Delete query(1)

SQL  0.052s

```
DELETE FROM store
WHERE store_id = 1;
```

5.1.3 Delete query(2)

Firebase Realtime DB



0.230s

```
def delete_store(store_id):  
    # Simple and easy  
    db.reference('store').child(str(store_id)).delete()
```

Live demo

5.2 Query 1 (From Assignment)

SQL  33.701s

```
SELECT *,  
       (SELECT COUNT(*) FROM rental r2, payment p2 WHERE r2.rental_id= p2.rental_id AND p2.amount<p.amount)  
          AS count_smaller_pay  
FROM rental r, payment p  
WHERE r.rental_id = p.rental_id;
```

5.2 Query 1 (From Assignment)

Aggregate function



```
SELECT *,  
       (SELECT COUNT(*) FROM rental r2, payment p2 WHERE r2.rental_id= p2.rental_id AND p2.amount<p.amount)  
          AS count_smaller_pay  
FROM rental r, payment p  
WHERE r.rental_id = p.rental_id;
```

Join



5.2 Query 1 (From Assignment)

The Firebase RealTime Database doesn't support either of them.

We need to do the **operations manually**.

5.2 Query 1 (From Assignment)

The slowest join ever



```
def query_1_join_operation_slow():
    payments = list(db.reference('payment').get())

    res = []

    #The slow part : getting the rental information directly from the database
    for i in payments:
        rental = list(dict(db.reference('rental').order_by_child('rental_id').equal_to(i['rental_id']).get()).values())
        tmp = i
        # In case there is no corresponding rental
        if len(rental) == 0:
            continue

        # We are assuming that there is only one rental for each payment
        for j in rental[0]:
            tmp[j] = rental[0][j]
        res.append(tmp)

    return res
```

5.2 Query 1 (From Assignment)

The fastest join

```
def query_1_local():
    # Acquiring all tables
    payments = list(db.reference('payment').order_by_child('rental_id').get())
    rentals = list(db.reference('rental').order_by_child('rental_id').get())

    # The tables are sorted, so we can use 2 pointers to match the rental_id
    p_i = 0
    joint_table = []
    for i in rentals:
        while payments[p_i]['rental_id'] < i['rental_id']:
            p_i += 1
        if payments[p_i]['rental_id'] == i['rental_id']:
            tmp = payments[p_i]
            for j in i:
                tmp[j] = i[j]
            joint_table.append(tmp)
```

5.2 Query 1 (From Assignment)

Doing Aggregation  3.725s

```
# Making the aggregation
payments_by_amount = list(db.reference('payment').order_by_child('amount').get())
for i in joint_table:

    # Binary search the answer
    low = 0
    high = len(payments_by_amount)
    while low < high:
        if low < 0:
            low = -1
            break
        mid = (low + high) // 2
        if payments_by_amount[mid]['amount'] < i['amount']:
            low = mid + 1
        else:
            high = mid
    i['count_smaller_pay'] = low

return joint_table
```

5.2 Query 1 (From Assignment)

Using denormalization

```
def query_1_with_extra_table():
    table = db.reference('payment_rental_smaller_pay').get()
    if table is None:
        print("The table doesn't exist. Going to compute the table content...")
        new_table = query_1_local()
        db.reference('payment_rental_smaller_pay').set(new_table)
    return new_table
return table
```

5.2 Query 1 (From Assignment)

Using denormalization (with listeners)

```
db.reference('payment').listen(cb)
```

First time



5.694s

Subsequent times



1.930s

06

Time for Questions

All questions about Google Firebase are welcome

Thank you
for your attention!