Sistem Pengambilan Data Dari SQL Server ke Firebase

1st Ikhsan Deantony
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
ikhsandeantony@student.telkomuniversity.ac.id

2nd Ahmad Sugiana
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
sugianaa@telkomuniversity.ac.id

3rd Junarto Halomoan
Fakultas Teknik Elektro
Universitas Telkom
Bandung,Indonesia
junartho@telkomuniversity.ac.id

***Abstrak* — Penelitian ini bertujuan untuk mempermudah pengawasan Centralized Traffic Control (CTC) dengan mengintegrasikan SQL Server Management Studio (SSMS) ke Jupyter Notebook dengan mengunakan skrip Python. Proses integrasi ini memungkinkan perubahan data di SQL Server dapat terkirim ke Python, karna Firebase merupakan *platform* NoSQL sehingga data SQL harus diubah menjadi JSON agar dapat dikirimkan ke Firebas. Penelitian ini melibatkan pembuatan penghubungan SQL *server* dengan Python dan pengiriman datanya dan penghubungan Python dengan Firebase dan pengiriman datanya, serta pengujian durasi waktu yang dibutuhkan untuk sinkronisasi data antara SQL dan Pyhton. Hasil pengujian menunjukkan bahwa pengiriman data SQL ke Python memiliki waktu yang bervariasi dari 1 sampai 10 detik dan memiliki rata-rata pengiriman selama 5 detik dikarenakan faktor-faktor yang mempengaruhi kecepatan pengambilan data dari SQL ke Python, seperti beban server SQL, efisiensi skrip Python, koneksi, spesifikasi *device*, serta hanya 1 *device* untuk menjalankan ketiga *program* tersebut.**

Kata kunci—Centralized Traffic Control, SQL Server, Python, Firebase

I. PENDAHULUAN

Persinyalan kereta api merupakan suatu bentuk, warna atau cahaya untuk memberikan isyarat kepada petugas perkeretaapian agar perjalanan kereta api aman. [1] Jenis persinyalan kereta api terbagi menjadi persinyalan elektrik dan mekanik. Persinyalan Elektrik adalah suatu peralatan untuk memberikan isyarat berupa cahaya atau warna yang memiliki arti tertentu dan biasanya berupa lampu LED (*Light Emitting Diode*).[2] Sinyal mekanik adalah perangkat sinyal yang digerakkan secara mekanik, contohnya papan atau lengan instruksi yang dinaikkan dan diturunkan untuk memberi perintah kepada masinis kereta api.[3] Kondisi persinyalan kereta api dapat dipantau melalui CTC yaitu sebuah sistem *monitoring* dan kontrol persinyalan kereta api yang berasal dari Amerika Utara. CTC juga dapat digunakan untuk memantau kondisi dari persinyalan kereta api apakah berjalan dengan keadaan normal atau sedang terjadi gangguan yang dipantau oleh operator dan teknisi *maintenance* di ruang kontrol. [4] Dalam pemantauan persinyalan kereta api di Indonesia operator harus berada di ruang kontrol dalam melakukan pemantauan kondisi. Untuk membagikan tampilan CTC saat ini hanya menggunakan sistem *Local Area Network* (LAN) yang memiliki batas jarak sehingga pemantauan menjadi tidak efisien dan memerlukan waktu yang cukup lama untuk melakukan perbaikan jika terjadi kegagalan pada kondisi persinyalan[5]. Untuk mengatasi ketidakefisienan dalam pemantauan dan masalah waktu yang cukup lama dalam melakukan perbaikan jika terjadi kegagalan pada kondisi persinyalan, kami memberikan solusi dengan memanfaatkan teknologi *Internet of Things (*IoT) dimana persinyalan yang ditampilkan di SCADA datanya dikirimkan ke *cloud* lalu dikirimkan ke smartphone sehingga data kondisi persinyalan dapat dipantau operator dan teknisi *maintenance* tanpa harus berada di *control room* dan dapat meningkatkan kecepatan respon teknisi *maintenance* pada perbaikan dan pemeliharaan instrumen pada persinyalan kereta api.[6]

II. KAJIAN TEORI

A. SQL Server Management Studio

SQL Server Management Studio adalah *Relational Database Management System* (RDMS) yang digunakan oleh Microsoft yang memiliki fungsi utama sebagai *database* 7 server yang dapat menyimpan dan mengirimkan data. SSMS menggunakan bahasa pemrograman SQL.[7] SQL dapat digunakan untuk mengambil, memasukkan, memperbarui, menghapus data, dan mengatur struktur *database* [10]

B. Jupyter Notebook

Jupyter Notebook adalah aplikasi *web open-source* untuk membuat dokumen yang berisikan kode, visualisasi yang mendukung berbagai bahasa pemrograman, dalam *capstone design* ini kami menggunakan Jupyter Notebook untuk menjalankan skrip berbahasa Python agar dapat mengirim data SQL ke Firebase.[8]

C. Firebase

Firebase Realtime Database merupakan basis data *online* yang dapat digunakan sebagai media penyimpanan data dari aplikasi. Firebase memiliki beberapa fitur, salah satunya yaitu realtime database yang disimpan secara *cloud*, layanan ini menggunakan API, data disimpan sebagai JSON dan disinkronkan secara *realtime* ke setiap klien yang terhubung, apabila ada perubahan pada data yang tersimpan, maka setiap *user* yang terhubung akan menerima pembaruan data secara otomatis. Firebase merupakan platform NoSQL yang berarti

tidak dapat menerima langsung data SQL sehingga perlu 10 menggunakan skrip Python untuk mengubah data SQL menjadi data JSON sebelum dikirim ke Firebase.[9]

III. METODE

A. Pengujian Data Transfer SQL ke Python

1. Cara Kerja

Pengiriman data SQL ke Python dilakukan dengan mengonfigurasi koneksi SQL *Server* dan *database* yang sesuai sehingga data SQL dapat ditarik dengan Python untuk disimpan.

2. Implementasi

a. Koneksi antara Python dan SQL

Agar data SQL dapat terkirim ke Python perlu dikoneksikan dengan menghubungkan *server* SQL dan *database*.

```
import pyodbc
import pandas as pd
import re
import time
from datetime import datetime
import firebase_admin
from firebase_admin import credentials, db as firebase_db

# Konfigurasi koneksi ke SQL Server
conn = pyodbc.connect('DRIVER={SQL SERVER};'
                      'SERVER=IKHSAN-PC\SQLEXPRESS;'
                      'DATABASE=CLI_TELKOM_ST_SIM 020124_DATA_CTC;'
                      'Trusted_Connection=True')

# Konfigurasi Firebase
cred = credentials.Certificate('C:/Users/ikhsan/Documents/SUG1/tap-project-1a9ec-firebase-adminsdk-d66x5-2051e133ee.json')
firebase_admin.initialize_app(cred, {
    'databaseURL': 'https://tap-project-1a9ec-default-rtdb.firebaseio.com'
})
```

GAMBAR 3.31
Konfigurasi koneksi SQL Server

Gambar1 merupakan skrip untuk menghubungkan Python ke server SQL dengan menggunakan pyodbc yang dikoneksikan ke s*erver* dan *database* yang diinginkan.

b. Pengambilan Data SQL

```
ref = firebase_db.reference('DATA_CTC')  # Mendapatkan referensi ke node 'DATA_CTC' di Firebase

# Fungsi untuk membaca last_timestamp dari file
def read_last_timestamp(file_path):
    try:
        with open(file_path, 'r') as file:
            return file.read().strip()  # Membaca timestamp terakhir dari file
    except FileNotFoundError:
        return '0000-00-00 00:00:00.000'  # Nilai default jika file tidak ditemukan

# Fungsi untuk menyimpan last_timestamp ke file
def save_last_timestamp(file_path, timestamp):
    with open(file_path, 'w') as file:
        file.write(timestamp)  # Menyimpan timestamp terakhir ke file

timestamp_file = 'last_timestamp.txt'  # Nama file untuk menyimpan timestamp terakhir
last_timestamp = read_last_timestamp(timestamp_file)  # Membaca timestamp terakhir dari file

# Fungsi untuk mengambil data baru dari SQL Server
def fetch_new_data(last_timestamp):
    query = "SELECT * FROM TRENDTABLE1 WHERE TS > ? ORDER BY TS DESC"
    try:
        return pd.read_sql(query, conn, params=[last_timestamp])  # Menjalankan query untuk mengambil data baru
    except Exception as e:
        print(f"Error executing query: {e}")  # Menangkap dan mencetak kesalahan jika query gagal
        return pd.DataFrame()  # Mengembalikan DataFrame kosong jika terjadi kesalahan
```

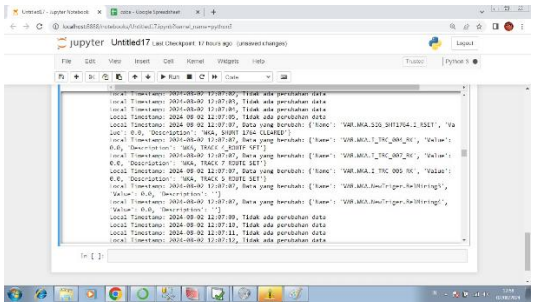
GAMBAR3.2
Pengambilan data

Gambar 3.2 pengambilan data SQL dari tabel 'TRENDTABLE1' dengan kondisi hanya baris yang memiliki stempel waktu 'TS' yang lebih besar dari 'last_timestamp' yang lalu diurutkan hasilnya berdasarkan kolom 'TS' dalam urutan menurun atau data yang terbaru.

3. Langkah Pengujian

a. Mengaktifkan SQL Server Management Studio dan *connect* ke server.
b. Mengaktifkan Jupyter Notebook beserta skrip Python
c. Run skrip Python sampai terlihat *output* berupa tabel dari SQL yang berarti terkoneksi dengan SQL Server Management Studio

Catat waktu saat Python berhasil menerima data SQL



Gambar 3.3
Output

B. Pengujian Data Transfer Python ke Firebase

1. Cara Kerja

Pengiriman data Python ke Firebase perlu mengonfigurasi skrip Python dengan Firebase menggunakan kredensial dari *file* JSON dan mengatur referensi ke lokasi *database* Firebase.

2. Implementasi

a. Koneksi antara Python dan Firebase

```
import pyodbc
import pandas as pd
import re
import time
from datetime import datetime
import firebase_admin
from firebase_admin import credentials, db as firebase_db

# Konfigurasi koneksi ke SQL Server
conn = pyodbc.connect('DRIVER={SQL SERVER};'
                      'SERVER=IKHSAN-PC\SQLEXPRESS;'
                      'DATABASE=CLI_TELKOM_ST_SIM 020124_DATA_CTC;'
                      'Trusted_Connection=True')

# Konfigurasi Firebase
cred = credentials.Certificate('C:/Users/ikhsan/Documents/SUG1/tap-project-1a9ec-firebase-adminsdk-d66x5-2051e133ee.json')
firebase_admin.initialize_app(cred, {
    'databaseURL': 'https://tap-project-1a9ec-default-rtdb.firebaseio.com'
})
```

GAMBAR 3.4
Konfigurasi Koneksi Firebase

Gambar 3.4 merupakan skrip untuk menghubungkan Python dan Firebase dengan menggunakan kredensial dari *file* JSON yang didownload dari Firebase dan mengatur referensi ke lokasi *database* Firebase menggunakan URL.

b. Source Code

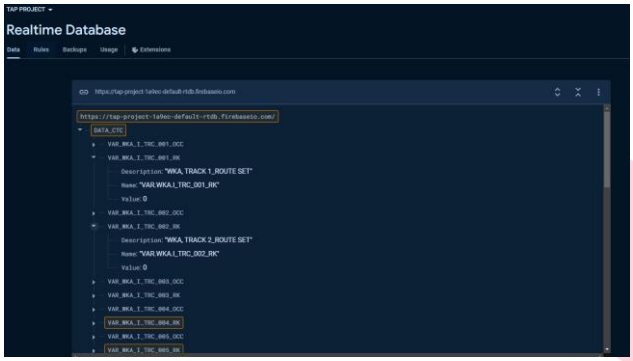
```
try:
    while True:
        # Mengonversi last_timestamp ke format yang sesuai
        last_timestamp_str = datetime.strptime(last_timestamp, '%Y-%m-%d %H:%M:%S.%f').strftime('%Y-%m-%d %H:%M:%S')

        latest_data = fetch_new_data(last_timestamp_str)  # Mengambil data baru dari SQL Server
        changed_data = {}

        for index, row in latest_data.iterrows():  # Iterasi melalui setiap baris data baru
            name = row['Name']
            value = row['Value']
            description = row['Description']

            user_data = {
                'Name': name,
                'Value': value,
                'Description': description,
            }
            clean_name = re.sub(r'[.#$[\]]', '_', name)  # Membersihkan nama agar sesuai dengan format Firebase
            changed_data[clean_name] = user_data

        # Kirim data yang berubah ke Firebase
        if changed_data:
            local_timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')  # Mendapatkan timestamp lokal saat ini
            for clean_name, user_data in changed_data.items():
                print(f"Local Timestamp: {local_timestamp}, Data yang berubah: {user_data}")
                ref.child(clean_name).set(user_data)  # Mengirim data yang berubah ke Firebase

            # Update waktu terakhir data yang diambil
            last_timestamp_dt = latest_data['TS'].max()  # Mengambil timestamp terbaru dari data yang diambil
            last_timestamp = last_timestamp_dt.strftime('%Y-%m-%d %H:%M:%S.%f')  # Mengupdate last_timestamp
            save_last_timestamp(timestamp_file, last_timestamp)  # Menyimpan last_timestamp yang baru

        else:
            local_timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
            print(f"Local Timestamp: {local_timestamp}, Tidak ada perubahan data")  # Jika tidak ada data yang berubah

        time.sleep(1)  # Menunggu 1 detik sebelum memeriksa lagi
except KeyboardInterrupt:
    local_timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    print(f"Local Timestamp: {local_timestamp}, Loop dihentikan oleh pengguna.")  # Menangani penghentian oleh pengguna
finally:
    # Simpan last_timestamp terakhir ke file saat skrip selesai
    save_last_timestamp(timestamp_file, last_timestamp)  # Menyimpan last_timestamp terakhir
    conn.close()  # Tutup koneksi dengan benar
```

GAMBAR 3.5
Source Code untuk mengirim data ke Firebase

Gambar 3.5 merupakan skrip untuk mengirimkan data perubahan ke Firebase jika ada data baru yang muncul di SQL.

3. Langkah Pengujian

a. Mengaktifkan Jupyter Notebook beserta skrip Python
b. Run skrip Python sampai terlihat *output* berupa tabel dari SQL
c. Buka Firebase dengan *project* yang sesuai
d. Pastikan data pada Firebase sesuai dengan data pada Python

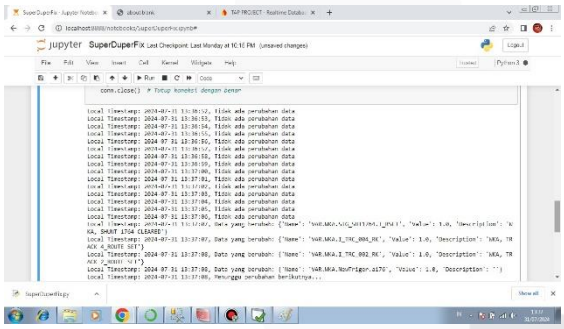


GAMBAR 3.6
Data Firebase

IV. HASIL DAN PEMBAHASAN

A. Hasil Pengujian Data Transfer SQL ke Python

Percobaan konektivitas SQL dan Python dengan menggunakan skrip Python pyodbc



GAMBAR 4.1
Output Python

Pada Gambar 4.1 merupakan tampilan saat Python berhasil terhubung dengan memunculkan data dari SQL Server Management. Data yang berhasil diterima yaitu data nama, *value*, dan deskripsi.

Percobaan	SQL	Python	Waktu SQL ke Python
1	16:05:43	16:05:45	0:00:02
2	16:10:27	16:10:35	0:00:08
3	16:11:21	16:11:25	0:00:04
4	16:12:17	16:12:25	0:00:08
5	16:13:21	16:13:25	0:00:04
6	16:13:58	16:14:05	0:00:07
7	16:15:05	16:15:15	0:00:10
8	16:15:58	16:16:05	0:00:07
9	16:16:44	16:16:45	0:00:01
10	16:17:40	16:17:45	0:00:05
11	16:18:26	16:18:35	0:00:09
12	16:18:57	16:19:05	0:00:08
13	16:19:58	16:20:05	0:00:07
14	16:20:33	16:20:35	0:00:02
15	16:21:11	16:21:15	0:00:04
16	16:21:44	16:21:45	0:00:01
17	16:22:27	16:22:34	0:00:07
18	16:23:00	16:23:05	0:00:05
19	16:23:45	16:23:54	0:00:09
20	16:24:20	16:24:25	0:00:05
21	16:24:57	16:25:04	0:00:07
22	16:25:33	16:25:34	0:00:01
23	16:26:37	16:26:45	0:00:08
24	16:27:10	16:27:15	0:00:05
25	16:27:52	16:27:55	0:00:03
26	16:28:25	16:28:35	0:00:10
27	16:29:10	16:29:15	0:00:05
28	16:29:47	16:29:55	0:00:08
29	16:30:24	16:30:24	0:00:00
30	16:32:21	16:32:25	0:00:04
Rata Rata			0:00:05

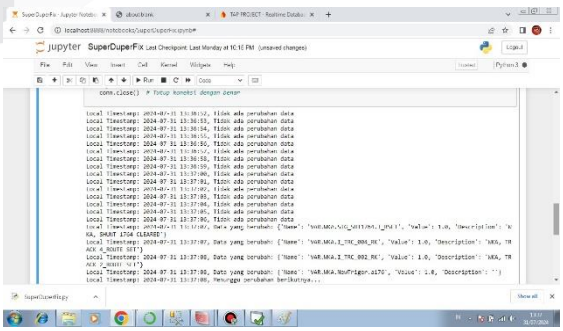
GAMBAR 4.2
pengukuran waktu pengiriman

Pada Gambar 4.2 merupakan hasil pengujian waktu pengiriman dari SQL ke Python dengan rata-rata waktu pengiriman mencapai 5 detik dengan waktu terlama mencapai 10 detik.

4. Analisis Pengujian

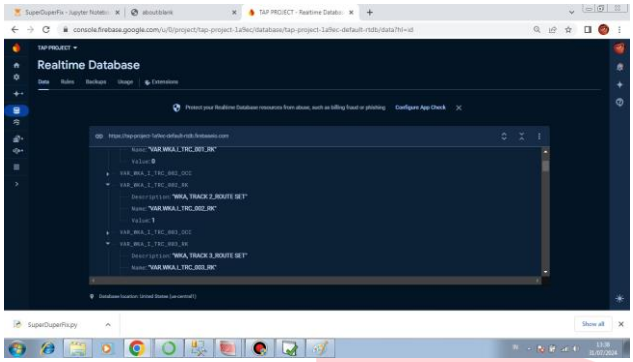
Hasil dari pengujian dapat dilihat pada Gambar 4.2 menunjukkan bahwa data dari SQL Server Management Studio dapat terkirim ke Python dengan menghubungkan *server* SQL dan *database* yang sesuai. Pengiriman data ini memiliki variasi waktu dari 1 sampai 10 detik dan memiliki rata-rata 5 detik. Variasi ini menunjukkan adanya faktor-faktor yang mempengaruhi kecepatan pengambilan data dari SQL ke Python, seperti beban server SQL, efisiensi skrip Python, koneksi, spesifikasi *device*, serta hanya 1 *device* untuk menjalankan ketiga *program* tersebut.

B. Hasil Pengujian Data Transfer Python ke Firebase



GAMBAR 4.3
Output Python

Pada Gambar 4.3 merupakan data-data yang berhasil Python terima dan langsung dikirimkan ke Firebase berupa data nama, *value,* dan deskripsi.



GAMBAR 4.4
Firebase

Pada Gambar 4.4 merupakan tampilan saat data Python diterima oleh Firebase, data-data ini tersimpan pada DATA_CTC yang terbagi menjadi data sinyal dan data rel yang berisikan nama, *value*, dan deskripsi.

5. Analisis Pengujian

Hasil analisis pengujian pengiriman data Python ke Firebase berjalan sangat baik, data yang terkirim sesuai dan lengkap, waktu pengiriman data tersebut juga sangat cepat, data tersebut langsung dikirimkan ke Firebase saat Python berhasil menerima data dari SQL sehingga tidak ada data yang tertinggal, tertumpuk bahkan terhapus

V. KESIMPULAN

Pada pengujian data transfer SQL ke Python dan Python ke Firebase menunjukkan bahwa pengiriman data SQL ke Python memiliki waktu yang bervariasi dari 1 sampai 10 detik dan memiliki rata-rata pengiriman selama 5 detik. Variasi ini menunjukkan adanya faktor-faktor yang mempengaruhi kecepatan pengambilan data dari SQL ke Python, seperti beban server SQL, efisiensi skrip Python, koneksi, spesifikasi *device*, serta hanya 1 *device* untuk menjalankan ketiga *program* tersebut. Sementara itu pengiriman dari Python ke Firebase memiliki tingkat kesesuaian data sebesar 100%, data-data yang diterima tersimpan pada DATA_CTC yang terbagi menjadi data sinyal dan data rel yang berisikan nama, *value*, dan deskripsi. Waktu pengiriman data sangat cepat, data tersebut langsung terkirim ke Firebase saar Python berhasil menerima data dari SQL.

REFERENSI

[1] D. S. O. A. Darmawan, “Peningkatan Keamanan Perjalanan Kereta Api Dengan Penggunaan Sistem Axle Counter Dan Media Transmisi Fiber Optic Untuk Hubungan Blok Di Persinyalan Vpi (Studi Kasus Hubungan Blok Stasiun Surodadi – Pemalang),” 2017.

[2] L. Nawangwulan, “Persinyalan,” 2021.

[3] E. Sayuri1, “Perancangan Sistem Persinyalan Elektrik Di Stasiun Berbasis PLC Omron,” 2017.

[4] J. Alvert, Controlize Traffic Control, 1999.

[5] S. Aisyah, “Prototipe Pemindah Wesel Kereta Menggunakan RFID,” 2017.

[6] G. S. Chandra, “Pemanfaatan Flutter dan Electron Framework pada Aplikasi Inventori dan Pengaturan Pengiriman Barang,” 2020.

[7] T. Anastasius, “Pengembangan Sistem informasi,” 2010.

[8] linda, “project jupyter: a computer code that transformed science,” 2021.

[9] D. N. Ramadan, “Perancangan dan Realisasi Mobil Remote Control Menggunakan Firebase,” 2017.

[10] S. YAHYA, “Rancang Bangun Sistem Lelang Online,” 2010.