بسم الله الرحمن الرحیم

شرکت مهندسی نرم افزاری هلو

گزارش اتصال به DB

کاری از امیرعلی نسیمی

جریان کاری مربوط به Kafka

۱. وارد کردن کتابخانه‌ها و ماژول‌های لازم
   - `json`برای کار با داده‌های JSON.
   - `time`برای مدیریت زمان و تاخیر.
   - `mysql.connector`برای اتصال به پایگاه داده MySQL.
   - `django.db`برای مدیریت اتصالات پایگاه داده در Django.
   - `sys`, `os`و `pathlib.Path` برای مدیریت مسیرها و تنظیمات سیستم.
   - `Data_Connection.kafka_setup`برای تنظیمات Kafka شامل موضوع Kafka و تولید کننده پیام(producer).

۲. تعریف کلاس:`Command`
   - این کلاس از `BaseCommand` در Django به ارث می‌برد و یک دستور سفارشی را تعریف می‌کند.

۳. تعریف تابع:`check_db`
   - این تابع تلاش می‌کند تا به پایگاه داده MySQL متصل شود و اتصال موفق را تایید می‌کند.
   - در صورت بروز خطا در اتصال، پیام خطا چاپ می‌شود و برنامه با کد خروجی ۱ متوقف می‌شود.

٤. تعریف متد:`handle`
   - این متد اصلی دستور را اجرا می‌کند.
   - ابتدا تابع `check_db` برای بررسی اتصال به پایگاه داده فراخوانی می‌شود.
   - اتصال به پایگاه داده خارجی از طریق `connections['external_table']` برقرار می‌شود.

- یک حلقه بی‌نهایت برای بررسی و پردازش ورودی‌های جدید از جدول `audit_log` استفاده می‌شود.

۵. تعریف تابع داخلی:`Process`

- این تابع ورودی‌های جدید از جدول `audit_log` را پردازش می‌کند.
- برای هر ورودی جدید، یک پیام Kafka ساخته می‌شود و به سیستم Kafka ارسال می‌شود.
- پس از ارسال پیام، وضعیت ورودی در جدول `audit_log` به "پردازش شده" تغییر می‌کند و تغییرات در پایگاه داده ذخیره می‌شود.

٦. اجرای حلقه اصلی

- در هر تکرار حلقه، یک کوئری برای بازیابی ورودی‌های جدید (پردازش نشده) اجرا می‌شود.
- اگر خطای برنامه‌نویسی (`ProgrammingError`) رخ دهد، یک پیام خطا چاپ شده و برنامه ۱۰ ثانیه منتظر می‌ماند و سپس تلاش دوباره انجام می‌شود.
- ورودی‌های جدید بازیابی شده و به تابع `Process` ارسال می‌شوند.
- پس از هر بار بررسی، برنامه برای یک دوره کوتاه (۱ ثانیه) منتظر می‌ماند و سپس دوباره بررسی می‌کند.

جریان کاری مربوط به MQTT

۱. وارد کردن کتابخانه‌ها:

- کتابخانه‌های ``json`، `time`، `mysql.connector``، و ماژول‌های `django.db` و `django.core.management.base` وارد می‌شوند.

۲. تعریف کلاس Command:

- کلاس `Command` از `BaseCommand` ارث بری می‌کند و شامل دو متد `check_db` و `handle` است.

۳. متد check_db:

- این متد تلاش می‌کند تا به یک دیتابیس MySQL خارجی متصل شود.
- اگر اتصال موفق باشد، پیامی مبنی بر موفقیت آمیز بودن اتصال چاپ می‌شود و اتصال بسته می‌شود.
- در صورت بروز خطا در اتصال، پیام خطا چاپ شده و برنامه با `exit(۱)` خاتمه می‌یابد.

٤. متد handle:

- ابتدا متد `check_db` فراخوانی می‌شود تا اطمینان حاصل شود که اتصال به دیتابیس برقرار است.
- اتصال به دیتابیس خارجی از طریق `connections['external_table']` ایجاد می‌شود.
- کوئری SQL برای دریافت رکوردهایی که هنوز پردازش نشده‌اند (`processed = FALSE`) تنظیم می‌شود.
- یک حلقه بی‌نهایت برای بررسی مستمر دیتابیس وجود دارد.

۵. حلقه اصلی:

- درون حلقه، کوئری اجرا می‌شود تا رکوردهای جدید از جدول `audit_log` واکشی شوند.
- در صورت بروز خطا در اجرای کوئری، پیام خطا چاپ شده و برنامه به مدت ۱۰ ثانیه متوقف می‌شود، سپس دوباره تلاش می‌کند.
- اگر رکوردهای جدید یافت شوند، متد `Process` فراخوانی می‌شود تا این رکوردها پردازش شوند.

٦. متد Process:

- هر رکورد جدید پردازش می‌شود و یک پیام MQTT ساخته می‌شود که شامل نام جدول، داده‌ها (به صورت JSON)، و زمان ثبت رکورد است.
- پیام ساخته شده چاپ می‌شود.
- رکورد در دیتابیس به عنوان پردازش شده علامت‌گذاری می‌شود و تغییرات ذخیره می‌شوند.
- حلقه به مدت ۱ ثانیه متوقف می‌شود و سپس دوباره اجرا می‌شود تا رکوردهای جدید را بررسی کند.

نکات کلیدی:

- کد به طور مداوم دیتابیس را برای یافتن رکوردهای جدید بررسی می‌کند.
- اگر رکورد جدیدی پیدا شود، آن را پردازش کرده و تغییرات را ذخیره می‌کند.
- اگر خطایی در اتصال یا اجرای کوئری رخ دهد، برنامه به مدت کوتاهی متوقف می‌شود و سپس دوباره تلاش می‌کند.