BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Házi feladat dokumentáció

Üzleti Intelligencia

Félév: 2025 ősz

Kriptovaluta és részvény adatok elemzése

Sőrés Barna - (CASI6C)

sores.barni@gmail.com

Elkészült feladat rövid ismertetése

A projekt célja egy teljes üzleti intelligencia rendszer kialakítása volt kriptovaluta és részvény adatok feldolgozására és vizuális elemzésére. A rendszer automatikusan gyűjti a nyers adatokat a CoinPaprika API-ból, amely a kriptovaluta adatokért felelős, és a yfinance Python csomag segítségével a Yahoo Finance-ről, amely a részvény adatokért. Majd többlépcsős ETL folyamatokon keresztül tisztítja, egységesíti és adattárházba rendezi őket.

A feldolgozott adatok alapján dinamikus Grafana dashboardok készültek, amelyek lehetővé teszik az árfolyamok, trendek, volatilitás és anomáliák valós idejű vizsgálatát. A rendszer támogatja több eszköz összehasonlítását, rövidtávú momentum jelzések megjelenítését, valamint az egyes kriptók és részvények részletes idősoros elemzését. A teljes megoldás Docker Compose segítségével konténerizált környezetben futtatható, így könnyen hordozható és egyszerűen telepíthető.

Ütemezés

A rendszer öt percenként kéri le az adatokat mindkét külső forrásból, majd automatikusan lefuttatja a teljes feldolgozási folyamatot. Ez magába foglalja a nyers adatok betöltését, a tisztítást, a stage réteg frissítését, a DWH-ba történő betöltést, valamint a legfrissebb árfolyamok cache-elését is. Ennek köszönhetően a dashboardok mindig naprakész információt használnak, ami különösen fontos pénzügyi adatok esetén.

Az ütemezés kialakításánál a CoinPaprika API korlátaihoz kellett igazodni: az ingyenes csomag átlagosan öt percenként frissít adatot. Így végül egy olyan rendszer jött létre, amely a specifikációnál gyorsabban frissíti az adatokat, miközben a külső források korlátait is figyelembe veszi.

Adatbázis táblák

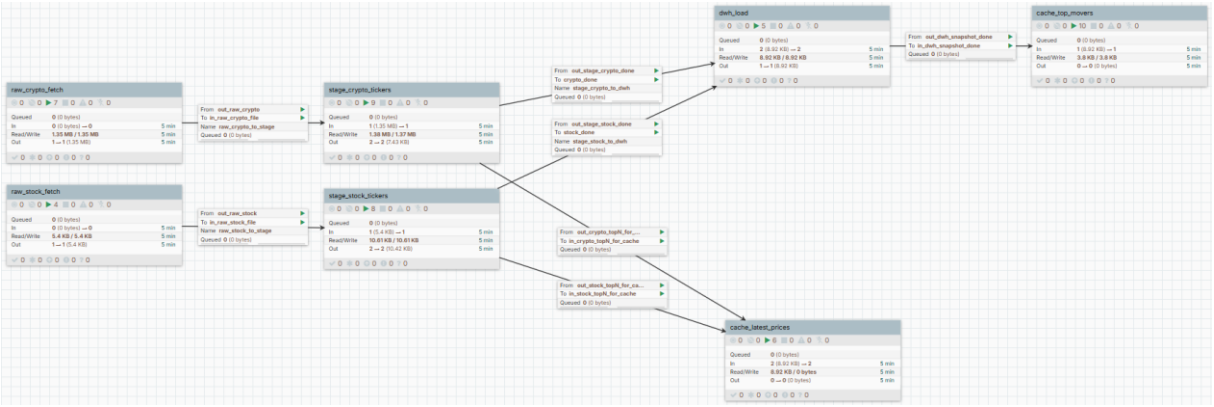
Az adatbázis minden szükséges táblája az első indításkor automatikusan létrejön a postgres-init mappában található init scriptek alapján. Ezek hozzák létre a stage réteg tábláit, ahová a nyers részvény és kriptovaluta adatok kerülnek először. A stage rétegből épül fel az adattárház struktúrája: az asset_dim dimenziótábla, az intraday_price_fact fact tábla, valamint egy nézet, amely napi záróár értékeket számít az intraday adatok alapján.

A stage táblák friss adatai a load_intraday_snapshot tárolt eljáráson keresztül töltődnek át a DWH táblákba. Ez az eljárás felelős azért, hogy az eszközök adatai (symbol, név, típus) naprakészek legyenek, valamint hogy minden új árfolyam egyedi időbélyeg alapján kerüljön a fact táblába.

A rendszer az SP500 összetételét is előkészíti: az sp500_companies csv tartalma automatikusan bekerül egy config táblába. Ezt később a raw_stock_fetch process group használja, hogy csak az előre definiált részvények adatai töltődjenek le és kerüljenek fel a pipeline-ba.

NiFi Flow

Az elkészült adatfeldolgozó folyamat hét process groupból áll: raw_crypto_fetch, raw_stock_fetch, stage_crypto_tickers, stage_stock_tickers, dwh_load, cache_top_movers és cache_latest_prices. Ezek együtt valósítják meg a teljes ETL folyamatot a nyers API hívástól kezdve az adatok transzformálásán át egészen az adattárházba töltésig és a Redis cache frissítéséig. A flow tervezésénél szempont volt a hatékonyság, ezért ahol lehetett batch alapú feldolgozást alkalmaztam, hogy minimalizáljam a soronkénti overheadet és gyorsabb legyen a teljes pipeline.



1. Flow Struktúra

raw_crypto_fetch

Ez a process group felel a kriptovaluta adatok begyűjtéséért. A CoinPaprika API /v1/tickers végpontjáról kérdezi le a legfrissebb árfolyamokat. A folyamat tartalmaz retry logikát is: ha egy lekérés hibával tér vissza a rendszer tíz alkalommal megpróbálja újra.

A sikeres lekérés után beállításra kerül a MinIO bucket és a fájl kulcsa, amely a lekérdezés időpontján alapul, hogy később egyértelműen visszakereshető legyen. Végül a teljes JSON válasz mentésre kerül a MinIO-ba.

raw_stock_fetch

Ez a process group hasonló logikát követ, de API-hívás helyett egy ExecuteProcess komponens indítja a python-etl mappában található, saját Python scriptet. A script parancssori argumentumként megkapja, hány részvény adatát kell lekérnie. Ez az érték határozza meg, hogy a config táblában tárolt SP500 lista első N eleme kerüljön lekérésre (alapból market cap szerint csökkenő sorrendben).

A részvényadatok a yfinance könyvtár segítségével töltődnek le, majd ugyanúgy MinIO-ba kerülnek, mint a kriptovaluta adatok.

stage_crypto_tickers

Ebben a rétegben történik meg a MinIO-ba mentett kriptós JSON visszaolvasása. A korábban elmentett MinIO kulcs alapján betöltjük a fájlt, majd a batch kap egy batch_id-t, ami később fontos lesz a dwh_load működéséhez.

Ezután történnek a transzformációk: először kiválasztjuk az első N elemet (ahol N a Parameter Contextből érkezik) majd átnevezzük és síkba rendezünk minden mezőt. A kimeneten két irányba továbbítjuk az adatot: egyrészt Redis cache frissítésre, másrészt a DWH-be történő betöltés előkészítéséhez. Előtte persze még elmentjük a transzformált adatokat a megfelelő stage táblába.

stage_stock_tickers

A részvényadatok feldolgozása ugyanazon logika szerint történik, mint a kriptóké, csak itt az első N szűrésre nincs szükség, mivel ez már a Python scriptben megtörtént. A transzformáció is egyszerűbb, mivel az adatszerkezet laposabb, így elegendő csak a mezőneveket módosítani.

cache_latest_prices

A stage rétegekből ide érkeznek a transzformált adatok. Ezek batchben kerülnek eltárolásra Redisben, így a rendszer mindig gyorsan el tudja érni a legfrissebb kriptó és részvényárakat. A TTL 6 percre van állítva, ami igazodik az 5 perces frissítési ütemhez.

dwh_load

Itt történik meg a DWH betöltés. A stage rétegek második output portjai csak a fragment_id-t továbbítják. Ennek az az oka, hogy csak akkor szeretném lefuttatni a betöltést, ha mindkét eszköztípus (kriptó és részvény) friss adatai biztosan megérkeztek.

A MergeContent komponens Defragment stratégiája gyakorlatilag egy AND kapuként működik: csak akkor engedi tovább a flow-t, ha az adott batchhez tartozó összes szükséges részfolyamat befejeződött. Ha ez teljesült, akkor meghívásra kerül a load_intraday_snapshot tárolt eljárás, amely a DWH táblákba tölti az adatokat.

cache_top_movers

Amikor a DWH betöltés sikeres volt, ez a process group frissíti a Redisben tárolt top movers adatokat (külön kriptóra, részvényre és egyesített nézetre). Ezeket a Grafana dashboardok használják, ezért itt különösen fontos a gyors elérés és az, hogy a legújabb értékek legyenek benne.

Grafana Dashboardok

A projekt két külön Grafana dashboardot tartalmaz, amelyek a frissen lekért és feldolgozott részvény és kriptó adatokat jelenítik meg. A megvalósításhoz az Infinity plugint használom, amely lehetővé teszi, hogy közvetlenül egy HTTP API-ból kapott JSON-t jelenítsek meg grafikonokon. Mivel az adatok egy részét Redisben tárolom olyan szerkezetben, amit a Grafana beépített Redis Data Source pluginnel nem lehet értelmezni, valamint a PostgreSQL-ben lévő adatokhoz is API-n keresztül szerettem volna egységesen hozzáférni, ezért készült egy külön Python API. Ez az API egységes, jól strukturált válaszokat ad a Grafana számára, így minden dashboard problémamentesen tudja lekérni a szükséges adatokat.

A Market Movers & Momentum dashboard a napi részvénypiaci és kriptovaluta mozgások gyors áttekintésére készült. Tartalmaz top movers bar chartokat, amelyek a legnagyobb 24 órás százalékos változást mutatják meg. Emellett likviditás–momentum scatter plotok jelennek meg, amelyek a return és a volumen kapcsolatát ábrázolják. A tőzsdei zárás figyelembe van véve, ezért a részvényes panelek csak kereskedési időben mutatnak adatot.

A Market Overview dashboard egy kiválasztott eszköz vagy eszközök részletesebb elemzését adja. Látható rajta az utolsó egy órás árfolyamgrafikon, egy indexelt teljesítménygrafikon, valamint trend és volatilitási sávok (MA10 + Bollinger-sávok). A rövid távú momentumot egy gauge mutató jeleníti meg, amely azt mutatja, hogy az ár a 10 periódusú mozgóátlag felett vagy alatt helyezkedik el. Táblázatok is vannak, amelyek a kripto és részvény piac szereplőit listázzák.

Projekt felállítása

NiFi:

A legegyszerűbb módja a flow beállításának, ha az üres canvason létrehozunk egy új process groupot, majd a neve mellett található ikonra kattintva beimportáljuk a Nifi_Flow mappában található NiFi_Flow.json fájlt. Ennek hatására a teljes struktúra megjelenik, minden process, kapcsolat és Controller Service a helyére kerül.

2. Nifi Flow Import

Ezután a fő flowban engedélyezni kell a következő Controller Service-eket: JsonReader, JsonWriter, PostgresPool, RedisConnectionPool, RedisDistributedMapCacheClientService. A PostgresPool és a RedisConnectionPool esetében előtte be kell állítani a jelszót, amit az Infrastructure mappa .env fájljában lehet megtalálni.

Ezen kívül a raw_crypto_fetch, raw_stock_fetch, stage_crypto_tickers és stage_stock_tickers processekben szintén engedélyezni kell az AWSCredentialsProviderControllerService nevű Controller Service-t.

Grafana:

A Grafana_Dashboards mappában található a két dashboard json fájl. Mielőtt ezeket beimportáljuk, létre kell hozni egy Infinity Data Source-t. A konfiguráción belül a URL, Headers & Params résznél meg kell adni a Base URL értékét, ami legyen http://host.docker.internal:8000.

Ha ez megvan, a New Dashboard gombra kattintva be lehet másolni a json fájlok tartalmát, és létrejönnek a dashboardok. Ezenkívül minden panelnél be kell menni az edit módba és elmenteni. Erre azért van szükség, hogy a panel automatikusan átállítsa a Data Source-t a most létrehozott Infinity kapcsolatra.

Forráskód elérhetőség

Github: https://github.com/barni10102/uzleti-intelligencia

Google Drive: https://drive.google.com/drive/folders/1qyMOwA3poa-msLgmhUjv_BjLxVGFyawu?dmr=1&ec=wgc-drive-globalnav-goto