Örnek Bir Stok Listeleme Scripti

Stok kart listeleme işleminizde stok sayınız 5000'in üzerinde ise ve offset kullanarak en sondaki stoğa erişme ihtiyacınız oluyorsa bu offset arttıkça stoğa erişme süreniz de artacaktır. Bu sürenin sabit kalıp en azından belirlenebilir bir süre içerisinde gerçekleşebilmesi için bir çözüm yolu önereceğiz. Aşağıda örneğini bulunan python scriptinin çalışma mantığı şu şekilde; _key parametresinin value değerini başlangıçta sıfır vererek stoklarınızın unique bir değeri olan ilk _key değerini alıyoruz, artan şekilde sıralı olarak ilk 100 sonucu elde ettikten sonra value parametresine son _key değerini veriyoruz, bu şekilde limit 100 değeri sabit olmak üzere tüm stokkartlarınızı listeleyebilirsiniz. Yani aslında kendi **offset** değerimizi kendimiz belirlemiş olup offsetten farklı olarak da her seferinde tüm listeyi çekip o offset değerine gitmektense filtreleme ile direkt sonraki stokkartlara gidip hızlı veri getirmiş oluyoruz. Bu yöntem size 2 saniye gibi bir süre içerisinde 100 stokkart listeleyerek stoklarınızın tamamını 2'şer saniyelik dilimler halinde en azından belirlenebilir bir sürede alabilmenizi sağlayacak. Ayrıca **selectedcolumns** parametresi servisten tüm alanları çekmektense sadece kullanacağınız alanların gelmesini sağlayarak performans artırır. Bu örneği sadece scf_stokkart_detay_listele için değil scf_stokkart_listele için de kullanabileceğinizi belirtmek isteriz.

```
 # -*- coding: utf-8 -*-

import json
import urllib2
import time
# Servis adresleri
uyekodu = 'diademo'
kullanici = 'demo'
sifre = 'demo'
firma_kodu = 21
donem_kodu = 5

wsAdresSis = "https://%s.ws.dia.com.tr/api/v3/sis/json" % uyekodu
wsAdresScf = "https://%s.ws.dia.com.tr/api/v3/scf/json" % uyekodu

print '-- BAŞLADI --'

try:
    # Login işlemi
    print 'Login yapılıyor...'
    postBody = {"login":
                    {"username": kullanici,
                     "password": sifre,
                     "disconnect_same_user": True,
                     "lang": "tr"
                     }
                }
    req = urllib2.Request(wsAdresSis,
                          data=json.dumps(postBody),
                          headers={"Content-Type": "application/json"})
    resultLogin = urllib2.urlopen(req).read()
    resultLogin = json.loads(resultLogin)
    print 'Login başarılı'
```

```
    startToplam = time.time()  # Toplam işlem süresini hesaplamak için
kullanılacak

    # scf_stokkart_detay_listele servisinde offset kullanmadan, filtre ve
limit vererek
    # belirli sayıdaki stok bloklarını (limit = 100) tahmin edilebilir
sürede elde edeceğiz.

    baslangic = 0  # Başlangıç _key değeri
    i = 0
    limit = 100
    tarih_ilk_cagri = "2000-01-01 00:00"  # Tüm stokları ilk sefer almak
için bu değeri kullanabilirsiniz.
    tarih = "2021-03-05 16:00"  # Bu tarihten itibaren değişen stokları
almak için bu şekilde kullanabilirsiniz.

    while i < 1000:  # 100*1000 stok için dolaşır. 100.000 stok üstü için
arttırılmalıdır.
        i += 1

        # Süreyi başlatalım. Her 100 stok için işlem süresini
hesaplayacağız.
        start = time.time()

        # Servisimizi uygun filtre ile çağıralım.
        body = {"scf_stokkart_detay_listele":
                    {"session_id": resultLogin['msg'],
                     "firma_kodu": firma_kodu,
                     "donem_kodu": donem_kodu,
                     "filters": [{"field": "durum", "operator": "=",
"value": "A"},
                                 {"field": "_key", "operator": ">", "value":
baslangic},
                                 # İlk çağrıda _date filtresi vermeyip veya
eski bir tarih verip sonraki çağrılarda
                                 # son çağrı tarih-saatinizi bu formatta
vererek son güncellenenleri elde edebilirsiniz.
                                 {"field": "_date", "operator": ">",
"value": tarih_ilk_cagri}],
                     "sorts": [{"field": "_key", "sorttype": "ASC"}],
                     "params": {"_key_sis_depo": 0, # Miktarın hesaplanacağı
depo key değeri
                                "_key_sis_depo_filtre": 0, # Miktarın
hesaplanacağı depo filtre key değeri
                                # İleri tarihli fişlerdeki miktarı da
dikkate alır. Alınmayacaksa günün tarihi verilmelidir.
                                "tarih": "2099-12-31",

                                # selectedcolumns parametresi ile
seçilebilecek alanlar:
                                # __dinamik__1 - 10, fiyat1 - 10, ekalan1 -
```

```
6, _key_scf_marka, _key_scf_odeme_plani,
                                # _key_sis_ozelkod1 - 11, doviz1 - 10,
_key_tedarikciler,
                                # kdvdurumu1 - 10, fiyatkartsayisi,
ozelliksayisi, gercek_stok_fat, gercek_stok_irs,
                                # sip_verilen, sip_alinan, urunagaclari,
urunozellikleri,
                                # b2c_durum, b2c_goster, b2b_gonder,
b2c_depomiktari... gibi
                                # selectedcolumns parametresi verilmediğinde
gelen herhangi bir ya da daha fazla alanı
                                # seçerek veri fazlalığından
kaçınabilirsiniz.
                                # m_birimler, m_fiyatlar, m_resimler,
m_varyantlar, m_urunOzellikleri, m_kategoriler alanları
                                # stokta varsa gelmektedir, bunlar
defaulttur.
                                # Sadece ihtiyacımız olan alanları çekerek
sorguyu hızlandırabilirsiniz.
                                "selectedcolumns": ["_key", "_date",
"stokkartkodu", "aciklama", "anabirimkey", "ozelkod2",
                                                    "kdvdurumu", "fiyat1",
"fiyat2", "gercek_stok", "fiili_stok",
                                                    "anabarkod"]
                                },
                    "limit": limit
                    }
                }
        reqStok = urllib2.Request(wsAdresScf,
                                  data=json.dumps(body),
                                  headers={"Content-Type":
"application/json"})

        resultStok = urllib2.urlopen(reqStok).read()
        resultStok = json.loads(resultStok)

        end = time.time()
        sure = end - start
        print '\n%s. Çağrı:' % i
        print 'İşlem süresi: %f' % sure  # Yanıt süresi

        if resultStok['code'] == '200' and len(resultStok['result']) > 0:
            print 'Gelen stok sayısı:', len(resultStok['result'])
            print 'Gelen stok kodları:', map(lambda x: x['stokkartkodu'],
resultStok['result'])

            # Stok sayısı limitten az ise tekrar dolaşmayalım.
            if len(resultStok['result']) < limit:
                break
            else:
                # Sonraki başlangıç değerimiz son gelen _key değeri olmalı
```

```
                baslangic = resultStok['result'][len(resultStok['result']) -
1]['_key']
        else:
            break

    endToplam = time.time()
    print '\nToplam işlem suresi: %f' % (endToplam - startToplam)
    print '-- BİTTİ --'
except Exception, e:
    import traceback
    traceback.print_exc()
```