PostgreSQL Veritabanı Sunucusu Üzerinde

Slony-I ile Replikasyon[1]

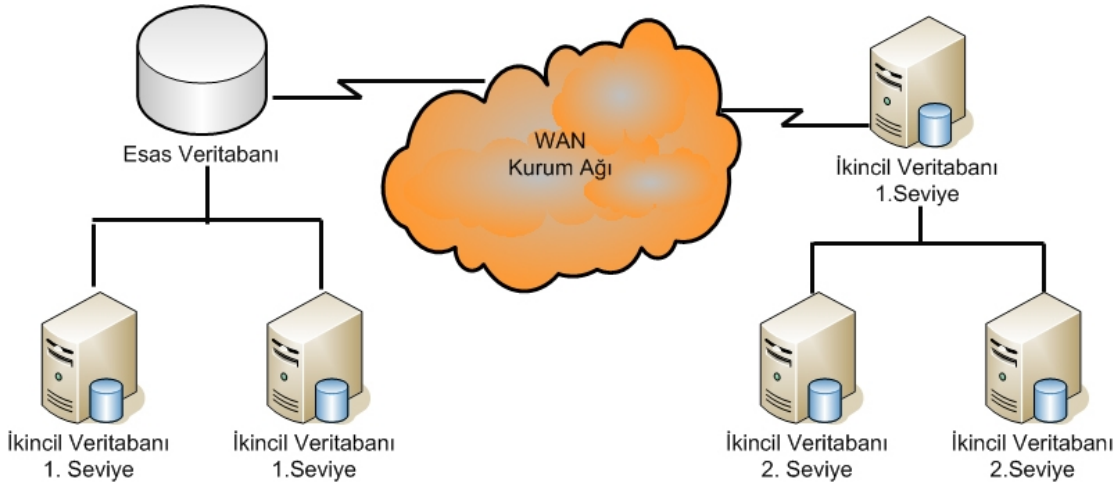
İsmail YENİGÜL
EnderUNIX Çekirdek Takım Üyesi
ismail at enderunix dot org
ismail.yenigul at endersys dot com dot tr
http://www.enderunix.org
http://www.endersys.com.tr

1 Bu belgenin güncel bir haline http://www.enderunix.org/docs/postgresql/slony_replication.pdf adresinden ulaşabilirsiniz.

İçerik

1. Giriş

Slony-I, tek ana (*master*) PostgreSQL sunucusundan birden fazla ikincil (*slave*) PostgreSQL sunucusuna (*master to multiple slaves*) eşzamansız (*asenkron*) olarak çalışan bir veri eşleme sistemidir. Slony-I, veri yedeklenmesi, başarım ve güvenlik gibi unsurların önplana çıktığı orta ve büyük ölçekli PostgreSQL sunucuları için ideal bir çözüm ortamı sunmaktadır. Genel bir Slony-I yapısı aşağıdaki gibi olabilir. Tüm slave sunucular değişiklikleri doğrudan master sunucudan alabileceği gibi diğer bir slave sunucudan da alabilir.



Genel bir Slony-I yapısına örnek.

Slony-I, veri eşlemesini tablo bazında gerçekleştirmektedir. Kabaca izah etmeye çalışacak olursak, veri eşlemesi yapılacak tabloya `UPDATE`, `INSERT` ve/veya `DELETE` komutları için `AFTER` niteliğinde *trigger*'lar eklenerek, bu kısıtlar altında yapılan her değişiklik Slony-I tarafından kendi kayıt tablosuna yazılmaktadır. Bu sayede ikincil sunucular bu tablodaki değişikliklere bakarak kendi tablolarını güncellerler.

Verinin eşlenmesi esnasında göz önünde tutulması gereken kısıtlar ise şu şekilde listeleyebiliriz:

- Otomatik olarak şema değişikliklerini henüz tespit edememektedir.
- `BYTEA` tipindeki sütunların eşlenmesi için herhangi kısıt bulunmazken, *LO* (L*arge Object*) tipi nesnelerin veri eşlemesi gerçekleştirilmemektedir.[2]
- Otomatik olarak *fail-over* yapamamaktadır.[3]
- Slony-I, PostgreSQL 7.3.3 ve daha üstü versiyonları trafından desteklemektedir.

2 Bunun sebebi, LO tipindeki verilerin veritabanında saklanması esnasında (sayısı çoğu zaman önceden kestirilemeyen) birden fazla satır kullanılmasıdır.

3 Slony-I geliştiricilerine göre, bu görev ağ izleme yazılımları tarafından gerçekleştirilimelidir. Slony-I esas görevi olan veri eşlemeyi yapmalıdır.

2. Kurulum

Slony-I kaynak kodlarına http://slony.info/ adresinden ulaşabilirsiniz. Programın derlenmesi için gerekli adımlar şu şekilde kısaca özetlenebilir: (Bu noktadan sonra, sisteminizde çalışır durumda bir PostgreSQL sunucusu olduğu kabul edilecektir.)

```
# tar jxvf slony1-1.2.9.tar.bz2[4]
# ./configure
# make
# make install
```

2.1. Slony-I Kavramları

Slony-I'deki en önemli iki kavram `slonik`[5] ve `slon` komutlarıdır.

Slonik

`slonik`, komut satırında çalışan bir yapılandırma yöneticisidir. Slonik,

- Veri eşlemesindeki sunucuların (*nodes*) tanımlanması,
- Bu sunucuların birbirleri ile nasıl haberleşeceği,
- Hangi tabloların eşleneceği,
- Eşlemenin yapılmakta olduğu bir sistemde eşlenmesi istenen yeni bir tablonun sisteme nasıl ekleneceği,
- Hali hazırda eşlenmekte olan bir tablonun bu sistemden nasıl çıkarılacağı

gibi işleri yapmakta yardımcı olan bir araçtır.

`slonik` komutu genellikle ana veritabanı sunucusunun bulunduğu sunucu üzerinde çalıştırılır. Ayrıca, `slonik` kullanılarak yapılacak bir değişiklik esnasında, eşleme sisteminin üyesi diğer tüm veritabanı sunucularının, değişikliğin gerçekleştirildiği sunucuya erişebilir olması gerekmektedir.

Slon

`slon`, Slony-I sistemindeki tüm sunuculardaayrı ayrı çalışan bir servis olup, veri eşlemesini gerçekleştirir. Bu sebeple, eşleme sisteminin aktif olduğu tüm süreç boyunca, `slon` servisi ayakta olmalıdır. Çalışan bir ikincil veritabanı sunucusunda, `slon` servisinin ayakta olmadığı bir durumda, ana sunucuda gerçekleştirilecek değişiklikler ikincil sunucuya yansıtılamaz. Ek olarak, yeni başlatılan bir *slon* servisi, çalışmaya bir önceki kaldığı konumundan devam eder.

Cluster

Eşleme sistemindeki sunucuların (*nodes*) tümünün oluşturduğu yapıya verilen isim.

4 Bu belgenin yazıldığı tarih itibari ile, 1.1.2.9 ve 1.1.1.9 en güncel Slony-I sürümleridir. Ve bu yazı boyunca verilecek örneklerde 1.1.2.9 sürümü baz alınacaktır.

5 Ek olarak, *Slon* kelimesi Rusçada fil, *slony* ise küçük fil anlamına gelmektedir.

Node

Cluster'daki her bir veritabanı sunucusuna verilen ad.

Replication Set (Replikasyon Kümesi)

Eşlenen tablo ve diğer içerik elemanlarının (`SEQUENCE` gibi) oluşturduğu grup.

Origin (Ana kaynak)

İlgili replikasyon kümesinin ana kopyasının tutulduğu sunucu.

Subscriber (Üye)

Sadece okuma işlemleri gerçekleştirmek üzere eşlenen ikincil replikasyon kümelerine verilen ad.

Provider (Sağlayıcı)

Üye (*subscriber*) sunucuların replikasyon verilerini sağlayan ana veritabanı sunucusu.

Event (Olay)

Ana kaynakta herhangi bir veri değişikliği olduğu zaman, bu değişiklik ilgili ikincil sunuculara bildirilir. Bu tür bildirimlere olay (*event*) adı verilmektedir.

Path (Yol)

Bir `slon` servisinin diğer bir sunucuya nasıl bağlanacağını belirleyen tanımlamaları içerir.

3. Slony Yapılandırması

3.1. Gereksinimler

Slony-I ile replikasyon yapmadan önce PostgreSQL tarafında bazı hazırlıkların tamamlanması gerekir.

Tablolardaki PRIMARY KEY Alanları

Replikasyonu yapılacak tablolarda, satırların kimliklendirilmesini sağlamak üzere `PRIMARY KEY` alanlarının olması gerekir.

Veritabanı Karakter Kodlaması

Slony-I, eşleşme yapılacak sunucularda veritabanı karakter kodlamasının aynı olduğunu kabul eder. Veritabanı oluşturulurken buna dikkat edilmelidir.

Zaman Ayarları

Replikasyon yapılacak sunucularda, `slon` servisinin hangi değişikliklerin uygulanıp uygulanmayacağına karar vermesi için sunucuların zaman ayarları önceden yapılandırılmış olmalıdır. Bu işin otomatiğe bağlanması için NTP (*Network Time Protocol*) kullanılması tavsiye edilir.

Ağ Bağlantısı Ayarları

Tüm sucunulardaki `postgresql.conf` dosyaları, ağ arayüzünü dinleyecek şekilde yapılandırılmalıdır. Bunun için `postgresql.conf` dosyasında yer alan `listen_address` değişkeni üzerinde ilgili değişiklik yapılmalıdır.[6]

Tüm sunucuların birbirlerine parola sormadan erişebiliyor olması gerekir. Bunun için her sunucunun `pg_hba.conf` dosyasında (ya da kullanılan ilgili güvenlik mekanizmasında) hem kendi IP adresine, hem de diğer sunuculara gerekli erişim hakkı verilmelidir.

Örneğin *cluster*'daki sunucu IP adresleri 192.168.0.10 ve 192.168.0.20 ise her iki sunucudaki `pg_hba.conf` dosyasında aşağıdaki satırlara benzer satırlar olmalıdır.

```
host    all       all       192.168.0.10   255.255.255.255    trust
host    all       all       192.168.0.20   255.255.255.255    trust
```

Her iki makinenin birbirine erişilebilir olduğundan mutlaka emin olunmalıdır. Sunucuların birbirine bağlantı kuramadığı bir durumda replikasyon çalışmayacaktır.

3.2. Örnek Yapılandırma

Aşağıda, şu özelliklere göre örnek bir replikasyon anlatılacaktır.

Ana Sunucu	192.168.0.10
Ana Sunucu Veritabanı	ecm
İkincil Sunucu	192.168.0.20
İkincil Sunucu Veritabanı	ecm

Her iki sunucuda da replikasyon için gerekecek ağ veritabanı bağlantılarında, **slony** kullanıcısı kullanılacaktır.

6 listen_address ve pg_hba.conf üzerindeki değişikliklerin geçerli olması için veritabanı sunucusu baştan başlatılmalıdır.

Slony yapılandırmasına başlamadan önce her iki sunucuda da **slony** kullanıcısı ve PL/pgSQL prosedürel dili oluşturulmalıdır. Ve ek olarak, **slony** kullanıcısı `superuser` haklarına sahip olmalıdır.

Ana Sunucuda Yapılacak Değişiklikler

```
# su  - postgres[7]
$ createuser slony            # `slony' adında kullanıcı oluşturuluyor
Yeni rol superuser olsun mu? (y/n) y
CREATE ROLE
$ createlang plpgsql ecm      # `ecm' veritabanı üzerinde PL/pgSQL
                              # oluşturuluyor.
```

slony şema değişikliklerini otomatik olarak almadığı için ana sunucu üzerindeki şemaların ilk başka ikincil sunucuya aktarılması gerekir.

```
$ pg_dump -s ecm  >/tmp/ecm.schema
```

İkincil Sunucuda Yapılacak Değişiklikler

```
# su  - postgres
$ createuser slony
Yeni rol superuser olsun mu? (y/n) y
CREATE ROLE
$ createdb ecm                # İkincil sunucuda veritabanı sıfırdan
CREATE DATABASE               # oluşturuluyor.
$ createlang plpgsql ecm

$ scp root@192.168.0.10:/tmp/ecm.schema /tmp
$ psql -f /tmp/ecm.schema     # Ana sunucudan alınan veritabanı şeması,
                              # ikincil sunucuya uygulanır.
```

Aşağıdaki `slonik.sh` dosyası ile replikasyonu yapılandırmak için `slonik` komutuna verilecek parametreler tanımlanmıştır.

```
#!/bin/sh
#
# slonik.sh - slonik yapılandırma betiği
#

CLUSTER=endersyscluster
DB1=ecm
DB2=ecm
HOST1=192.168.0.10
HOST2=192.168.0.20
USER=slony
slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$HOST1 user=$USER';
node 2 admin conninfo = 'dbname=$DB2 host=$HOST2 user=$USER';
init cluster (id = 1, comment = 'Master Node');
create set (id = 1, origin = 1, comment = 'endersys data');

set add table (set id = 1, origin = 1, id = 1,
```

7 Veritabanını hangi kullanıcı ile initdb ettiğinize bağlı olarak buradaki postgres kullanıcı adı değişebilir.

```
        full qualified name = 'public.account',
        comment = 'account table');

set add table (set id = 1, origin = 1, id = 2,
        full qualified name = 'public.accountdetails',
        comment = 'accountdetails table');

set add table (set id = 1, origin = 1, id = 3,
        full qualified name = 'public.acctype',
        comment = 'acctype table');

set add table (set id = 1, origin = 1, id = 4,
        full qualified name = 'public.comments',
        comment = 'comments table');

set add sequence (set id = 1, origin = 1, id = 1,
        full qualified name = 'public.account_oid_seq',
        comment = 'Sequence account_oid_seq');

set add sequence (set id = 1, origin = 1, id = 2,
        full qualified name = 'public.comments_id_seq',
        comment = 'Sequence comments_id_seq');

store node (id = 2, comment = 'Node 2');
store path (server = 1, client = 2,
     conninfo = 'dbname=$DB1 host=$HOST1 user=$USER');
store path (server = 2, client = 1,
     conninfo = 'dbname=$DB2 host=$HOST2 user=$USER');
_EOF_
```

cluster name

`cluster name` ifadesi ile bu replikasyon ile ilgili kümelemeye (*cluster*) isim verilir. `slonik` buradaki örneğe göre `_$CLUSTER` (yani `_endersyscluster`) şeklinde bir şema oluşturur. Ve replikasyon ile ilgili verileri bu şema altında oluşturduğu tablolarda saklar. Aşağıda slony yapılandırmasından sonra `_endersyscluster` şeması altında oluşturulan tabloların listesi verilmiştir.

```
ecm=# SELECT table_name
        FROM information_schema.tables
       WHERE table_schema = '_endersyscluster';
   table_name
-----------------
 sl_nodelock
 sl_setsync
 sl_trigger
 sl_table
 sl_sequence
 sl_node
 sl_listen
 sl_path
 sl_subscribe
 sl_set
 sl_event
 sl_confirm
```

```
 sl_seqlog
 sl_log_1
 sl_log_2
 sl_registry
 sl_seqlastvalue
 sl_config_lock
 sl_status
(19 rows)
```

Buradaki en önemli tablo `sl_log_1`[8] tablosu diyebiliriz. Tablolardaki değişiklikler bu tabloya yazılmaktadır. Değişiklikler ikincil sunuculara aktarıldıktan sonra ilgili kayıt bu tablodan silinmektedir.

node 1 admin conninfo

Kümeleme bünyesindeki sunuculara `slonik` komutunun, hangi sunucuya ve veritabanına hangi kullanıcı ile erişeceği tanımlanır. `Node 1` genelde ana sunucuyu ifade eder. Burada tekrar belirtmekte fayda var: *`slonik` komutunun çalıştırıldığı anda replikasyon dahilindeki sunucuların ve PostgreSQL veritabanının çalışıyor olması gerekmektedir.* (*Fail-over* durumu hariç).

init cluster

Kümelemenin ilk yapılandırması bu komut ile yapılır. Bu komut yukarıda anlatılan `_endersyscluster` şemasını ve ilgili tabloları (ve gerekli bileşenleri) oluşturacaktır. Buradaki `id = 1`, bir üstteki `node 1` düğümüne işaret etmektedir.

create set

Replikasyonu yapılacak tablolar kümeler halinde tanımlanır. Bir küme içerisinde uygulamanın ihtiyaç duyduğu tablolar ve bunlarla ilişkisi olan tablolar tanımlanır. Bir replikasyon sistemde birden fazla küme tanımlanabilmektedir. `origin = 1` ile ana sunucunun 1 numaralı düğüm olduğu ifade edilir.

set add table

`set add table` ifadesi ile bir üstte tanımlanan kümede hangi tabloların olacağı belirtilir. `set id = 1`, bu tablonun 1 numaralı kümeye ait olduğunu, `id = 1`, tablonun kümedeki numarasını, `full qualified name`, tablo adını - bulunduğu şema ile birlikte - ifade eder. `comment` ile tablo hakkındaki yorumu içerir. Buradaki en önemli husus, tablo numaralarının (`id`) replikasyon sistemi genelinde tekil olması gerekir. Başka bir küme içinde bile olsa aynı `id` farklı bir tablo için kullanılamaz. Ayrıca bir tablo aynı anda iki farklı küme içerisinde tanımlanamaz.

8 Bu tablo MySQL veritabanı sunucusu tarafından veritabanındaki tüm değişikliklerin tutulduğu binlog dosyasına aşağı yukarı eşdeğerdir diyebiliriz.

set add sequence

`set add sequence` ifadesi ile bir üstte tanımlanan kümede hangi `SEQUENCE` olacağı belirtilir. Buradaki değişkenler `set add table` ile aynıdır. Tek fark `SEQUENCE` için `id` numarası ile tablolar için `id` numaraları arasında bir ilişki yoktur. Bu örnekte de görüldüğü gibi `public.account_oid_seq` için `id` olarak 1 kullanılmıştır. Tablolar ile `SEQUENCE` arasında `id` farkının olmamasının sebebi `SEQUENCE` ve tablo bilgilerinin kümeleme şeması üzerinde farklı tablolarda (`sl_table` ve `sl_sequence`) tutulmasıdır.

store node

Bu komut mevcut *cluster* yapılandırmasına yeni bir sunucu eklemek için kullanılır. Burada 2 nolu sunucu mevcut *cluster* sistemine eklenmiştir.

store path

`store path` parametresi replikasyon servislerinin haberleşmek için birbirlerine nasıl bağlanacağını belirler. Örnekte olduğu gibi çift yönlü bir bağlantı için iki tane `store path` parametresi tanımlanmıştır.

Tüm bu açıklamalardan sonra slony yapılandırmasını yapmak için aşağıdaki komutlar verilir.

```
# su - postgres
$ chmod 755 slonik.sh
$ ./slonik.sh
```

Ekrana herhangi bir hata mesajı çıkmazsa yapılandırma başarıyla gerçekleştirilmiş demektir.

Slony yapılandırmasını teyit etmek için slony tablolarında aşağıdaki sorgular yapılabilir.

```
# su - postgres
$ pgsql -U slony ecm

ecm=# SELECT  * FROM _endersyscluster.sl_node;
 no_id | no_active | no_comment | no_spool
-------+-----------+------------+----------
     1 | t         | Master Node| f
     2 | t         | Node 2     | f
(2 satır)

ecm=# SELECT  * FROM _endersyscluster.sl_table ;
 tab_id | tab_reloid |  tab_relname   | tab_nspname | tab_set |    tab_idxname    | tab_altered |     tab_comment
--------+------------+----------------+-------------+---------+-------------------+-------------+---------------------
      1 |     506340 | account        | public      |       1 | pk_account        | t           | account table
      2 |     506354 | accountdetails | public      |       1 | pk_accountdetails | t           | accountdetails table
      3 |     506360 | acctype        | public      |       1 | pk_acctype2       | t           | acctype table
      4 |     506496 | comments       | public      |       1 | pk_comments       | t           | comments table
(4 satır)
```

```
ecm=# SELECT  * FROM _endersyscluster.sl_sequence ;
 seq_id | seq_reloid |   seq_relname   | seq_nspname | seq_set |       seq_comment
--------+------------+-----------------+-------------+---------+-------------------------
      1 |     506352 | account_oid_seq | public      |       1 | Sequence account_oid_seq
      2 |     506499 | comments_id_seq | public      |       1 | Sequence comments_id_seq
(2 satır)


ecm=# SELECT * FROM _endersyscluster.sl_path ;
 pa_server | pa_client |                 pa_conninfo                  | pa_connretry
-----------+-----------+----------------------------------------------+--------------
         2 |         1 | dbname=ecm host=192.168.0.20 user=slony      |           10
         1 |         2 | dbname=ecm host=192.168.0.10 user=slony      |           10
(2 satır)
```

Veri eşlemesi yapılabilmesi için her iki sunucuda `slon` işleminin çalışıyor olması gerekir.

Ana sunucuda

```
$ slon endersyscluster "host=192.168.0.10 dbname=ecm user=slony" &
```

İkincil sunucuda

```
$ slon endersyscluster "host=192.168.0.20 dbname=ecm user=slony" &
```

komutları çalıştırılarak, slon servisleri ayağa kaldırılır.

`slon` servisinin bu şekilde ayrı sunucular üzerinden verilebildiği gibi, her iki komut da tek bir sunucu üzerinden verilebilir. Burada mühim olan `slon` servisinin hangi sunucudaki hangi veritabanı için işlem yapacağını doğru şekilde belirtmektir.

Slony replikasyonunun başlaması için yukarıdaki yapılandırma yeterli değildir. Bir önceki betikte kümelerden bahsetmiştik ve `create set` komutu ile 1 nolu tablo ve SEQUENCE kümesini tanımlamıştık. Kümeyi tanımladıktan sonra ana ve ikincil sunucunun bu kümeye üye yapılması gerekir. Sunucuları kümeye üye yapmak demek, veri eşlemesinin başlatılması demektir. Sunucular kümeye üye olmadıkları takdirde hiçbir veri eşlemesi yapılmayacaktır. Kümeye üye yapmak için ise `subscribe set` komutu aşağıdaki gibi kullanılır.

```
#!/bin/sh
#
# subscribe.sh - Sunucuları kümeye üye yapmak için kullanılacak betik.
#
CLUSTER=endersyscluster
DB1=ecm
DB2=ecm
HOST1=192.168.0.10
HOST2=192.168.0.20
USER=slony
slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$HOST1 user=$USER';
node 2 admin conninfo = 'dbname=$DB2 host=$HOST2 user=$USER';
subscribe set (id = 1, provider = 1, receiver = 2, forward = yes);
_EOF_
```

`subscribe.sh` dosyası yukarıdaki gibi tanımlandıktan sonra aşağıdaki komutlar verilerek çalıştırılır.

```
$ chmod 755 subscribe.sh
$ ./subscribe.sh
```

Slony, olmayan şemaları oluşturma veya şemadaki değişiklikleri aktarma gibi bir özelliğe sahip olmadığından dolayı eğer `slonik.sh` dosyasında belirtilen tablolar ve `SEQUENCE` nesneleri ikincil sunucuda oluşturulmadı ise replikasyon başlamayacaktır. Bunu engellemek için konunun en başında ana sunucu üzerindeki şema (`ecm.schema`) `pg_dump` komutu ile alınıp ikincil sunucuya aktarılmıştır. Bu işlem yukarıda yapılmadı ise mutlaka yapılmalıdır.

Sadece şemalar ikincil sunucuya atılmalıdır, replikasyon başlamadan önce tablolar ve `SEQUENCE` nesnelerinde hiçbir veri olmaması tavsiye edilir. Eğer replikasyon başladığında burada önceden veriler var ise bu veriler silinecektir. *Subscribe* işlemi ile tabloların mevcut verileri olduğu gibi ikincil sunuculara aktarılacaktır. Bundan sonraki değişiklikler `sl_log_1` tablosu üzerinden aktarılacaktır.

Biraz bekledikten sonra ikincil sunucudaki veritabanına bağlanarak verilerin eşlenip eşlenmediğini kontrol edebilirsiniz.

3.3. Replikasyona Yeni Eleman Ekleme

Mevcut replikasyon kümelerine doğrudan yeni tablo ve `SEQUENCE` eklenemediği için önce ayrı bir replikasyon kümesi oluşturulur. Daha sonra bu küme, daha önceki kümeyle birleştirilir. Bir kümenin diğer bir küme ile birleştirilmesi işlemi `merge set` komutu ile gerçekleştirilir.

Örneğin `contact` tablosunu ve `contact_seq SEQUENCE` elemanını 10 numaralı kümede oluşturmak için aşağıdaki gibi bir betik oluşturulur.

```
#!/bin/sh
#
# yenitablo.sh - Replikasyona yeni eleman ekler.
#

CLUSTER=endersyscluster
DB1=ecm
DB2=ecm
HOST1=192.168.0.10
HOST2=192.168.0.20
USER=slony
slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$HOST1 user=$USER';
node 2 admin conninfo = 'dbname=$DB2 host=$HOST2 user=$USER';

create set (id = 10, origin = 1, comment = 'endersys ilave data');

set add table (set id = 10, origin = 1, id = 5,
        full qualified name = 'public.contact',
        comment = 'contact table');
```

```
set add sequence (set id = 10, origin = 1, id = 3,
        full qualified name = 'public.contact_seq',
        comment = 'Sequence public.contact_seq');
_EOF_
```

`contact` tablosu için verilen 5 nolu `id` değerinin daha önceki kümelerde hiçbir tabloya verilmemiş olması gerekir. `contact_seq` için de aynı durum geçerlidir.

Kümenin oluşturulması için aşağıdaki komutlar verilir.

```
$ chmod 755 yenitablo.sh
$ ./yenitablo.sh
```

Bundan sonra `node 1` ve `node 2`'yi 10 numaralı kümeye üye yapmak gerekiyor. Üyelik sonucunda veriler eşlendikten sonra `merge set` komutu verilmelidir. *Subscribe* işlemi bitmeden *merge* işlemi yapılırsa sorunlar çıkacağı için *subscribe* işleminin bitmesi beklenmelidir. Bu bekleme işlemini yapmak için aşağıda görüldüğü gibi `subscribe set` komutundan sonra `wait for event` komutu verilmiştir. Bu komut kendisinden bir önceki olayın bitmesini beklemektedir. Yani `subscribe` işlemini. Önceki komut tamamlandıktan sonra bir sonraki komutlara geçiş yapılmaktadır.

```
#!/bin/sh
#
# subandmerge.sh
#

CLUSTER=endersyscluster
DB1=ecm
DB2=ecm
HOST1=192.168.0.10
HOST2=192.168.0.20
USER=slony
slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$HOST1 user=$USER';
node 2 admin conninfo = 'dbname=$DB2 host=$HOST2 user=$USER';

subscribe set (id = 10, provider = 1, receiver = 2);
wait for event[9] (origin = 2, confirmed = 1);
sync (id=1);
merge set ( id = 1, add id = 10, origin = 1 );
_EOF_
```

Ardından, betik şu şekilde çalıştırılır:

```
$ chmod 755 subandmerge.sh
$ ./subandmerge.sh
```

En sondaki `merge set` komutu ile 10 numaralı kümedeki tüm tablo ve SEQUENCE elemanları 1 numaralı kümeye eklendikten sonra 10 numaralı kümeye ait slony tablolarında hiçbir bilgi kalmayacaktır.

9 Buradaki origin değerinin node 2 olduğuna dikkat ediniz.

3.4. Tablolar Üzerinde Şema Değişikliği

Eşlenmiş tablolar üzerinde doğrudan SQL cümleleri çalıştırarak şema değişikliği (tabloya yeni sütün ekleme, veri tipini değiştirme vs.) yapılamaz[10]. Bunun yerine replikasyon üyesi sunucular üzerinde bu tür değişiklikler yapmak için `execute script` komutu kullanılmalıdır.

`execute script` komutu parametre olarak işletilecek SQL cümleciklerini içeren bir dosya alır. Bu dosya içerisinde kesinlikle `BEGIN`, `COMMIT` gibi *transaction* ifadeleri bulunmamalıdır.

```
--
-- yenialan.sql - Replikasyon kümesi içinde ilgili yeni alan tanımlar.
--
ALTER TABLE contact ADD COLUMN city character varying(15);
ALTER TABLE contact ALTER COLUMN city SET STORAGE EXTENDED;
```

Yukarıdaki komutları işletmek için aşağıdaki gibi bir betik hazırlanmalıdır.

```
#!/bin/sh
#
# execute.sh - Replikasyon kümeleri üzerinde değişiklik icra edecek
#              SQL betiklerini tüm cluster genelinde işletir.
#

CLUSTER=endersyscluster
DB1=ecm
DB2=ecm
HOST1=192.168.0.10
HOST2=192.168.0.20
USER=slony
slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$HOST1 user=$USER';
node 2 admin conninfo = 'dbname=$DB2 host=$HOST2 user=$USER';

EXECUTE SCRIPT (
   SET ID = 1,
   FILENAME = '/tmp/yenialan.sql',
   EVENT NODE = 1
);
_EOF_
```

3.5. Yedeğe Geçme (Fail-Over) Senaryosu

Ana sunucunun çalışamaz hale geldiğinde ikincil sunucuyu devreye almak için `fail-over` komutu kullanılır. Aşağıda 1 nolu ana sunucunun devre dışı kalması durumunda 2 nolu ikincil sunucuyu devreye almak için kullanılabilecek betik bulunmaktadır.

```
#!/bin/sh
#
```

10 Aksi halde bu değişiklikler diğer ikincil sunuculara yansıtılmayacaktır.

```
# failover.sh
#

CLUSTER=endersyscluster
DB1=ecm
DB2=ecm
HOST1=192.168.0.10
HOST2=192.168.0.20
USER=slony
slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$HOST1 user=$USER';
node 2 admin conninfo = 'dbname=$DB2 host=$HOST2 user=$USER';

FAILOVER (
   ID = 1,
   BACKUP NODE = 2
);
_EOF_
```

3.6. Slony Servisini Kaldırma

Herhangi bir sebepten dolayı veritabanından slony servisine ait her şeyi silmek için `uninstall node` komutu aşağıdaki gibi kullanılabilir.

```
#!/bin/sh
#
# uninstall.sh
#

CLUSTER=testcluster
DB1=ecm
DB2=ecm
HOST1=192.168.0.10
HOST2=192.168.0.20
USER=slony
slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$HOST1 user=$USER';
node 2 admin conninfo = 'dbname=$DB2 host=$HOST2 user=$USER';

UNINSTALL NODE ( ID = 2 );
UNINSTALL NODE ( ID = 1 );
_EOF_
```

3.7. Ekler

3.8. Teşekkürler

Volkan Yazıcı'ya PostgreSQL konusundaki çalışmalarından ve desteklerinden dolayı teşekkür ederim.

3.9. Kaynaklar

- http://slony.info/documentation/
- http://www.linuxjournal.com/article/7834
- Douglas, Korry & Douglas, Susan. *PostgreSQL: The comprehensive guide to building, programming, and administering.* July 2005, Sams Publishing