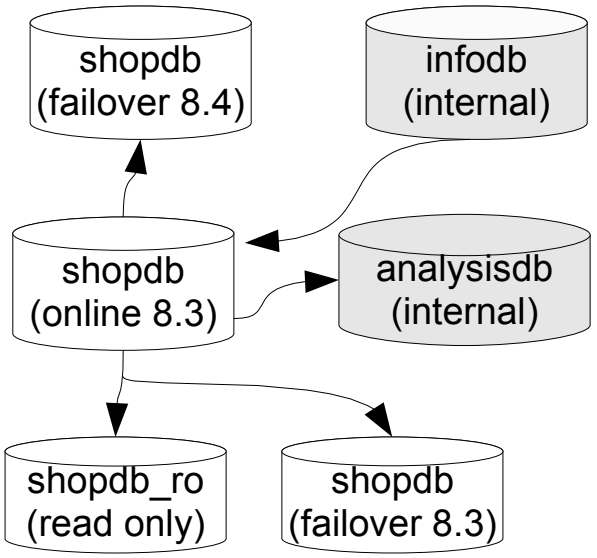
Londiste

Replication system for PostgreSQL

About Londiste

- Londiste is easy to use asynchronous master/slave replication tool written in python and is part of Skytools package.
- Skytools is a package containing PgQ module for Postgres, Python framework and several tools built on top of it
- Londiste uses PgQ as transport layer therefore it needs PgQ maintenance daemon called ticker.

- In Skype we use Londiste:
  - to transfer online data into internal databases
  - to create failover databases for online databases
  - to upgrade PostgreSQL versions
  - to distribute internally produced data into online servers
  - to create read only replicas for load balancing



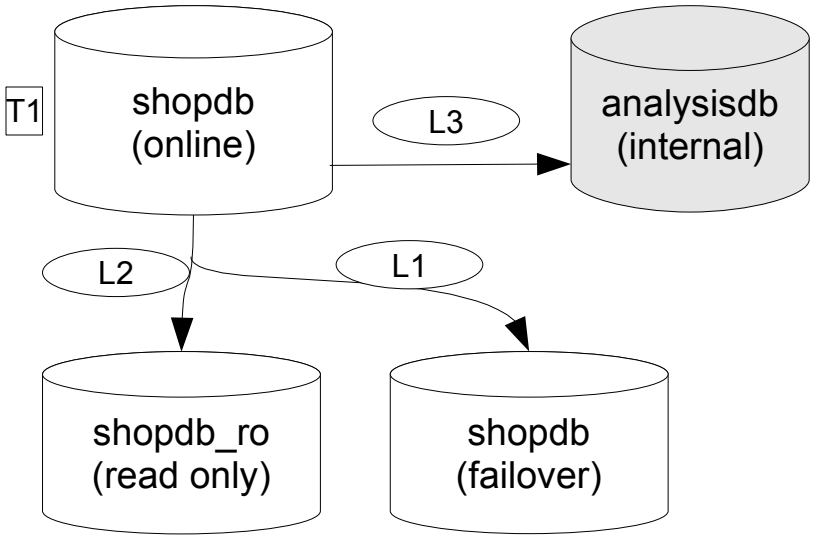
Londiste Skytools 2

Currently in use

Skytools 2: Londiste

- One Londiste process manages both provider and subscriber side
  - For example Londiste process L3 manages tables replication from shopdb to analysisdb
- Tables can be added/removed without affecting replication of other tables
- One process is not communicating with other processes
- pgqadm.py is used for ticking (T1)

- There is no support for SQL script execution
- There is no support for parallel copy
- There is no support for cascading



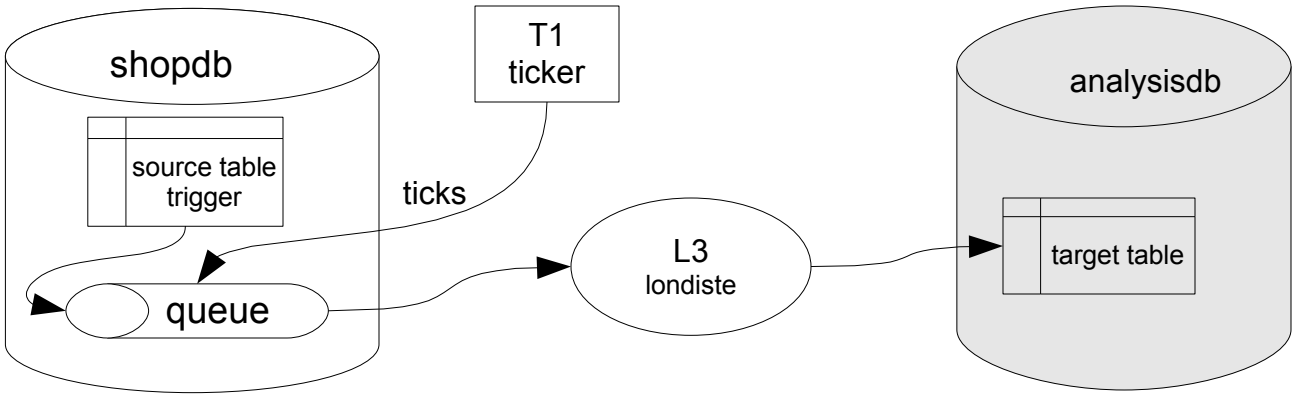
Skytools 2: Building from source tarball

- get latest tarball from:
  http://pgfoundry.org/projects/skytools
- Dependencies:
  - C compiler and GNU make
  - PostgreSQL development headers and libraries
  - Python development package
  - Psycopg2
- install skytools
  ```
  $ ./configure --prefix=/usr/local
  $ make
  $ make install
  ```

Skytools 2: Pgq setup

Since Londiste uses PgQ as transport layer you need to set it up first.
Basic PgQ setup would be illustrated by the following steps:

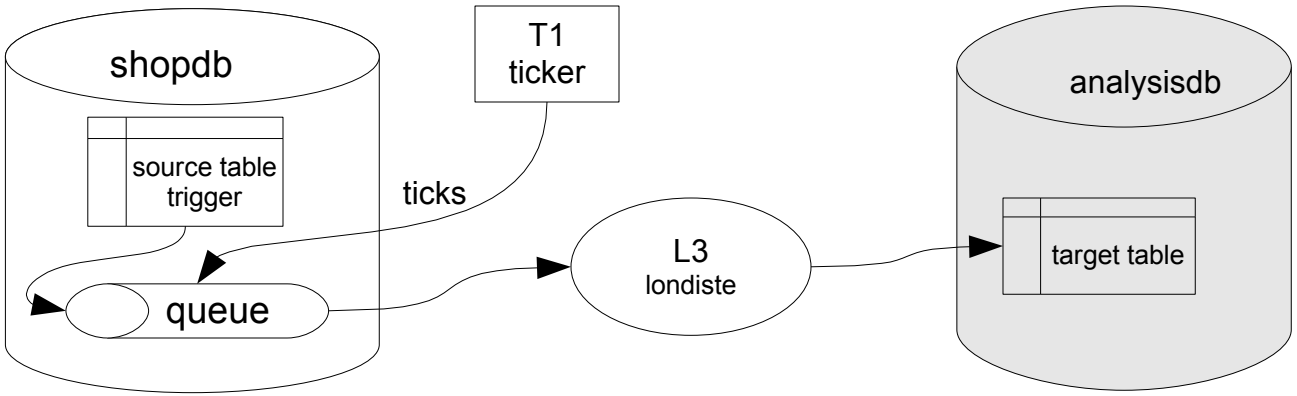
1. create the database
2. edit a PgQ daemon configuration file, say ticker.ini
3. install PgQ internal tables
   $ pgqadm.py ticker.ini install
4. launch the PgQ ticker on database machine as daemon
   $ pgqadm.py -d ticker.ini ticker



Skytools 2: Ticker configuration

```
[pgqadm]
job_name = T1
db = dbname= shopdb

logfile = ~/log/%(job_name)s.log
pidfile = ~/pid/%(job_name)s.pid
```



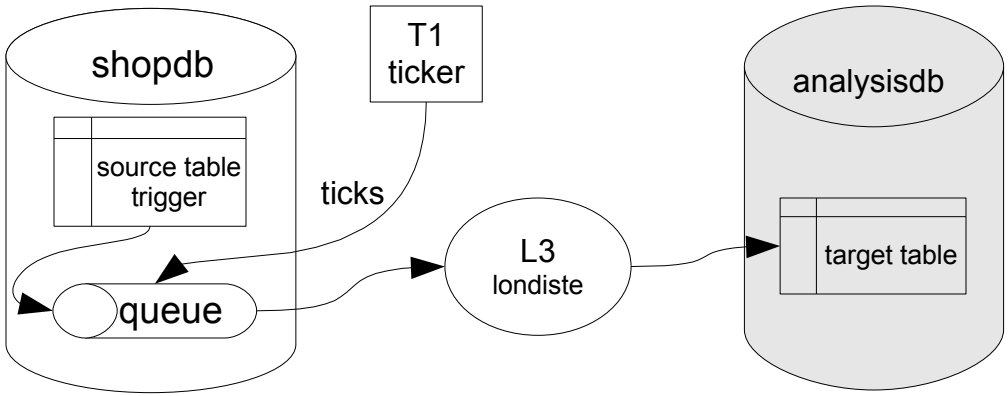
Skytools 2: Londiste setup

1. edit a londiste configuration file, lets say conf.ini
2. install londiste on the provider and subscriber databases. This step requires admin privileges on both provider and subscriber sides, and both install commands can be run remotely:
   $ londiste.py conf.ini provider install
   $ londiste.py conf.ini subscriber install
3. check that you have ticker running:
   $ ps -ef | grep ticker
4. launch the londiste replay process:
   $ londiste.py -d conf.ini replay
5. add table to replicate from the provider database:
   $ londiste.py conf.ini provider add table1
6. add table to replicate to the subscriber database:
   $ londiste.py conf.ini subscriber add table1

Skytools 2: Londiste Configuration

```
[londiste]
job_name = L3
pgq_queue_name = shopdb_replica
provider_db = dbname=shopdb
subscriber_db = dbname=analysisdb

logfile = ~/log/%(job_name)s.log
pidfile = ~/pid/%(job_name)s.pid
```



Londiste Skytools 3

Are We there yet?

Skytools 3: Keep good features from SkyTools 2

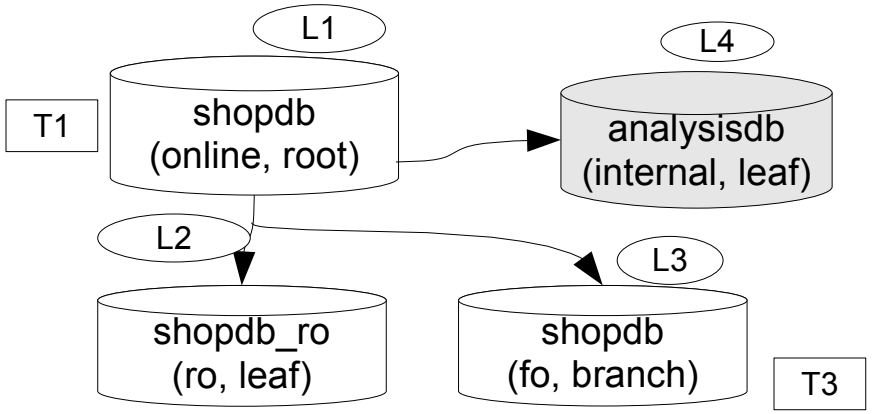
- Londiste process connects to only 2 databases
- Londiste only pulls data from queue
  - Administrative work happens in separate process (ticker)
  - Downtime of one Londiste process doesn't affect other replication or queue processes
- Relaxed attitude about tables
  - Adding/removing a table doesn't affect replication of other tables
  - no attempt is made to keep consistent picture between tables during initial copy

Skytools 3: New in Londiste

- Parallel copy - during initial sync several tables can be copied at the same time.
- EXECUTE command, to run SQL script on all nodes.
- Automatic table or sequence creation by importing the structure from provider node.
- Cascading support
  - Its goal is to keep identical copy of queue contents in several nodes.
  - Advanced admin operations: switchover, change-provider, pause/resume
  - Londiste process manages target node only

Skytools 3: Cascading

- **set** – group of nodes that distribute a single queue
- **node** – database that participates in cascade set
- node types:
  - **root** – master node of a cascade set
  - **branch** – node that carries full contents of the queue (can be provider)
  - **leaf** – data-only node (events are replicated, but can't be provider to other nodes)



Skytools 3: Building from source tarball

It is not a polished release, but a snapshot of current development tree. Although it may happen to have couple of working use-cases.

- get latest tarball: http://skytools.projects.postgresql.org/testing/skytools-3.0a1.tgz
- Dependencies:
  - C compiler and GNU make
  - PostgreSQL development headers and libraries
  - Python development package
  - Psycopg2
- $ ./configure --prefix=/usr/local
- $ make
- $ make install

Skytools 3: setup replica

1. edit a londiste configuration file, lets say L1.ini
2. Install Londiste and initialize nodes:

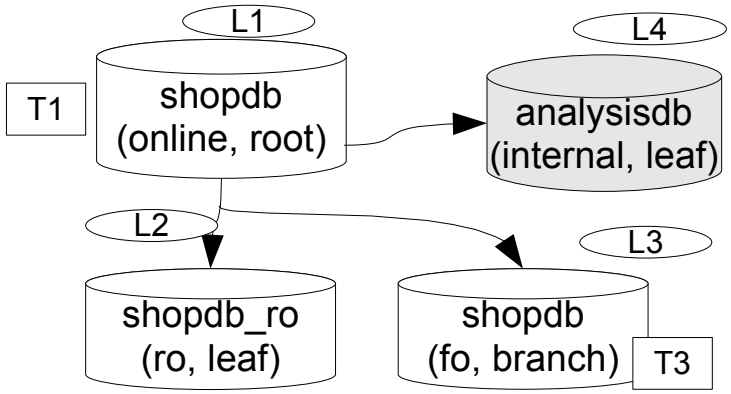
```
$ londiste L1.ini create-root shopdb dbname=shopdb

$ londiste L2.ini create-leaf shopdb_ro
  dbname=shopdb_ro --provider=dbname=shopdb

$ londiste L3.ini create-branch shopdb_T3
  dbname=shopdb –provider=dbname=shopdb

$ londiste L4.ini create-leaf analysisdb
  dbname=analysisdb --provider=dbname=shopdb
```

```
[londiste]
job_name = L1
db = dbname=shopdb
queue_name = shopdb_replica
logfile = log/%(job_name)s.log
pidfile = pid/%(job_name)s.pid
```



Skytools 3: setup replica

3. edit a ticker configuration file, lets say T1.ini
4. run ticker:

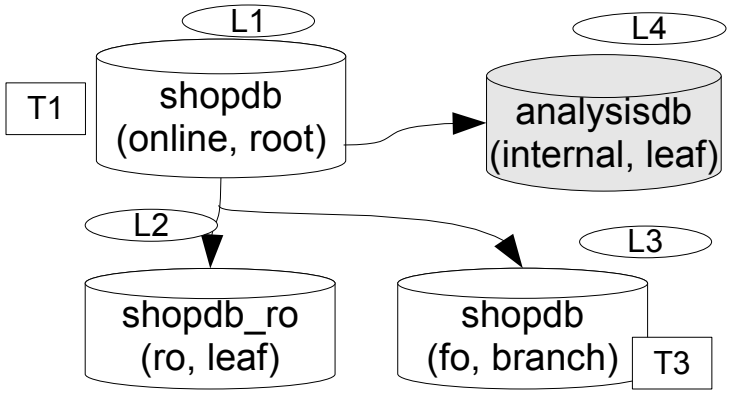
```
$ pgqadm T1.ini ticker -d

$ pgqadm T3.ini ticker -d
```

5. run Londiste:

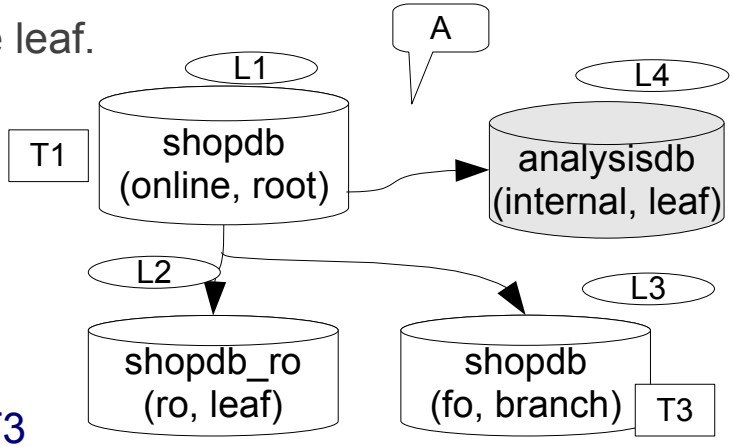
```
$ londiste L1.ini replay -d
$ londiste L2.ini replay -d
$ londiste L3.ini replay -d
$ londiste L4.ini replay -d
```

```
[pgqadm]
job_name = T1
db = dbname=shopdb
logfile = log/%(job_name)s.log
pidfile = pid/%(job_name)s.pid
```



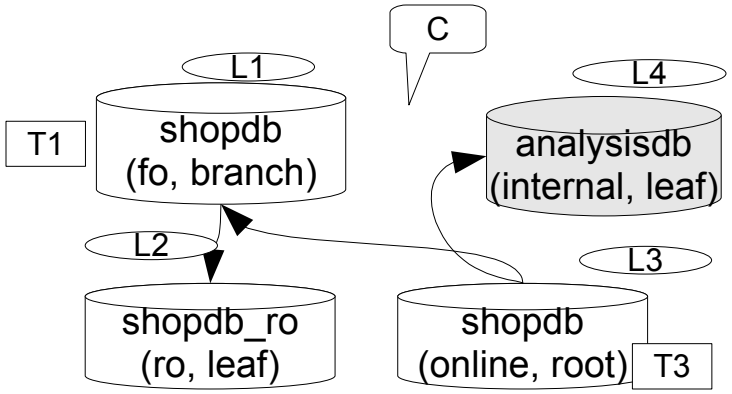
Skytools 3: Advanced admin operation examples

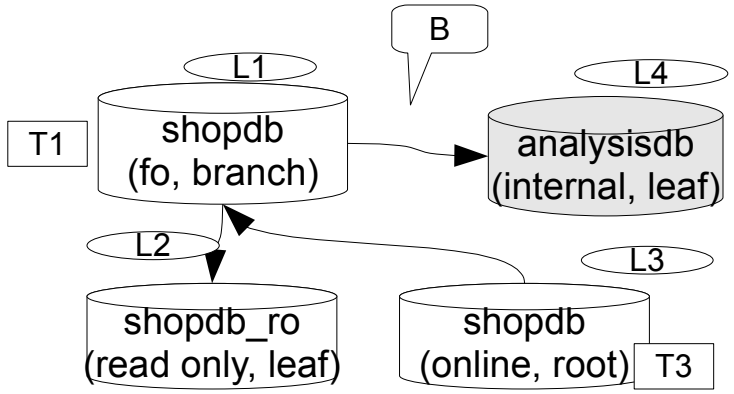
In example Ticker process T1 for shopdb (online) and T3 for shopdb (fo).
Londiste process L1 is root, L3 is branch, L2 and L4 are leaf.



1. Shopdb failover can be done with one command
   $ londiste L1.ini switchover --target=shopdb_T3

2. Analysisdb can change provider with command
   $ londiste L4.ini change-provider --target=shopdb_T3





Extras

Ticker

- Ticker reads event id sequence for each queue.
- If new events have appeared, then inserts tick if:
  - Configurable amount of events have appeared
    ticker_max_count (500)
  - Configurable amount of time has passed from last tick
    ticker_max_lag (3 sec)
- If no events in the queue, creates tick if some time has passed.
  - **ticker_idle_period (60 sec)**
- extra parameters in configuration file:
  - how often to run maintenance [seconds]
    maint_delay = 600
  - how often to check for activity [seconds]
    loop_delay = 0.5

Skytools 3: Building from GIT

It is for people who don't fear work-in-progress code and are prepared to give feedback on issues. Especially welcome would be people who could submit code/doc patches, to help us bring the final 3.0 release faster.

- fetch git tree:
  $ git clone git://github.com/markokr/skytools-dev.git
- fetch libusual submodule:
  $ git submodule init
  $ git submodule update
- generate ./configure script
  $ make boot
- now build as usual (--with-asciidoc is required when building from GIT)
  $ ./configure --prefix=... --with-asciidoc
  $ make
  $ make install

References

- http://skytools.projects.postgresql.org/doc/londiste.ref.html
- http://wiki.postgresql.org/wiki/Londiste_Tutorial
- http://skytools.projects.postgresql.org/skytools-3.0/doc/skytools3.html
- http://pgfoundry.org/projects/skytools