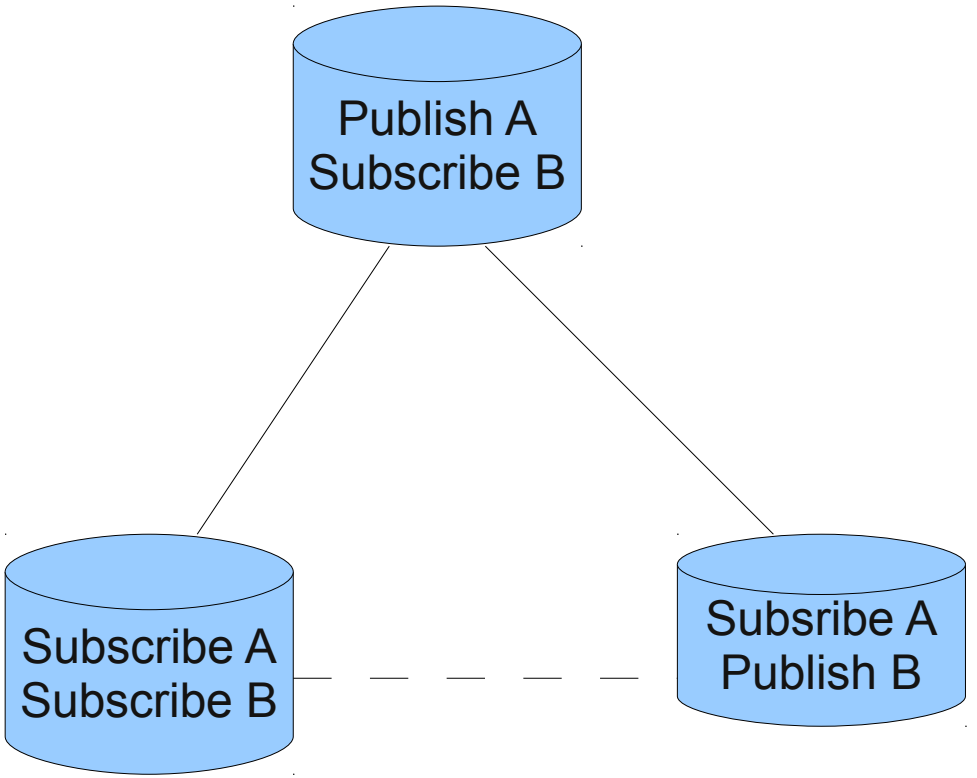
Data replication

Publish – subscribe model



High availability – redundancy - primary server, secondary server(s)

Failover, switchover, replication, database cluster, split-brain

Load Balancing

- load balancing database servers is easify for r/o access.
- modification much more complicated – write requests must be propagated to all servers in the cluster – no single solution.

Master servers – R/W requests
Slave servers – R/O requests
Stand-by servers – not accessible until they become master

Some solutions – only one server can handleupdate requests

Synchronous solutions – the data modifying transaction is not committed until accepted by all servers (distributed transaction) => no data lost on failover

Asynchronous solutions – commit → delay → propagation to other servers in the cluster => some data updates may be lost on failover

If synchroneous too slow => asynchroneous

Granularity point of view:

per-server high availability solution
per-database
per-table

Types of solution:

Shared-disk failover

- single disk array shared by multiple servers
- if the main DB server fails, another DB server mounts the DB
- synchronization overhead avoided by having just one instance of the database
- rapid failover with no data loss
- SAN (storage area networks) lower level communication protocol than NFS
- NFS is OK if mounted synchroneously (without caching)

- the stand-by server cannot never access the DB while the primary server is ON
- disk array – single point of fialure, when failed/corrupted, the DB becomes unavailable

File-system Replication

- HW redundnacy
- changes to a filesystem mirrored to another filesystem residing on another host
- mirroring must assure that the secondary filesystem is a consistent copy of the primary one
- the order of writes must be preserved

- LINUX world has a popular filesystem mirroring solution called DRBD

Warm stand-by thoough PITR (point-in-time recovery)

- a stand-by server is being kept current by reading the stream of write-ahead log (WAL) records (warm standby aka log shipping)
- whent the master server fails, the stand-by server has almost all data available and can be quickly made the master

- asynchronous solution
- only the entire DB server granularity

Master-slave replication

- all data modification queries go to the master server.
- the master server asynchronously sends data changes to the slave server
  => possible data loss during fail over
- the slave can answer read-only queries while the master server is running.
- the slave server ideal for OLAP queries

Example: Slony-I (per-table granularity, support for multiple slaves)

Statement-based replication moddleware

- a program intercepts every SQL query and sends it to one or all servers.
- each server operates independently.
- R/W queries are sent to all servers, while R/O queries can be sent to just one server, allowing the read workload to be distributed.

If queries are simply broadcast unmodified, functions like random(), CURRENT_TIMESTAMP, and sequences would have different values on different servers. If this is unacceptable, either the middleware or the application must query such values from a single server and then modify the write queries using those values before broadcasting them to the servers.
Also, care must be taken that all transactions either commit or abort on all servers, perhaps using two-phase commit.

Examples: Pgpool-II, Sequoia

Asynchronous Multimaster Replication

- data synchronization for occassionaly connected applications
- asynchronous multimaster replication
- each server works independently
- periodically communicates with the other servers to identify conflicting transactions
- the conflicts resolved by users / conflict resolution rules

Synchronous Multimaster Replication

- each server can accept write requests
- modified data is transmitted from the original server to every other server before each transaction commits.
- read requests can be sent to any server.
- sometimes shared disk used to reduce the communication overhead
- no problem with non-deterministic functions like random()
- advantage: any server can accept write requests. Hence, no need to distribute work-loads between masters and slaves

- Heavy write activity can cause excessive locking, leading to poor performance.

Comparison of high-availabilty approaches

Source: http://www.postgresql.org/docs/8.3/static/high-availability.html

Feature	Shared disk failover	Filesystem replication	Warm standby PITR	Master-slave replication	Statement-based repl. middleware	Asynchron. multimaster replication	Synchron. multimaster replication
No special HW required		X	X	X	X	X	X
Multiple masters allowed					X	X	X
No master server overhead	X		X		X		
No waiting for multiple servers	X		X	X		X	
No data lost on master failure	X	X			X		X
Slaves accept R/O queries				X	X	X	X
Per-table granularity				X		X	X
No conflict resolution necessary	X	X	X	X			X
Communic. method	Shared disk	Disk blocks	WAL	Table rows	SQL	Table rows	Table rows & row locks

Other solutions – incomparable with the previous ones

Data Partitioning

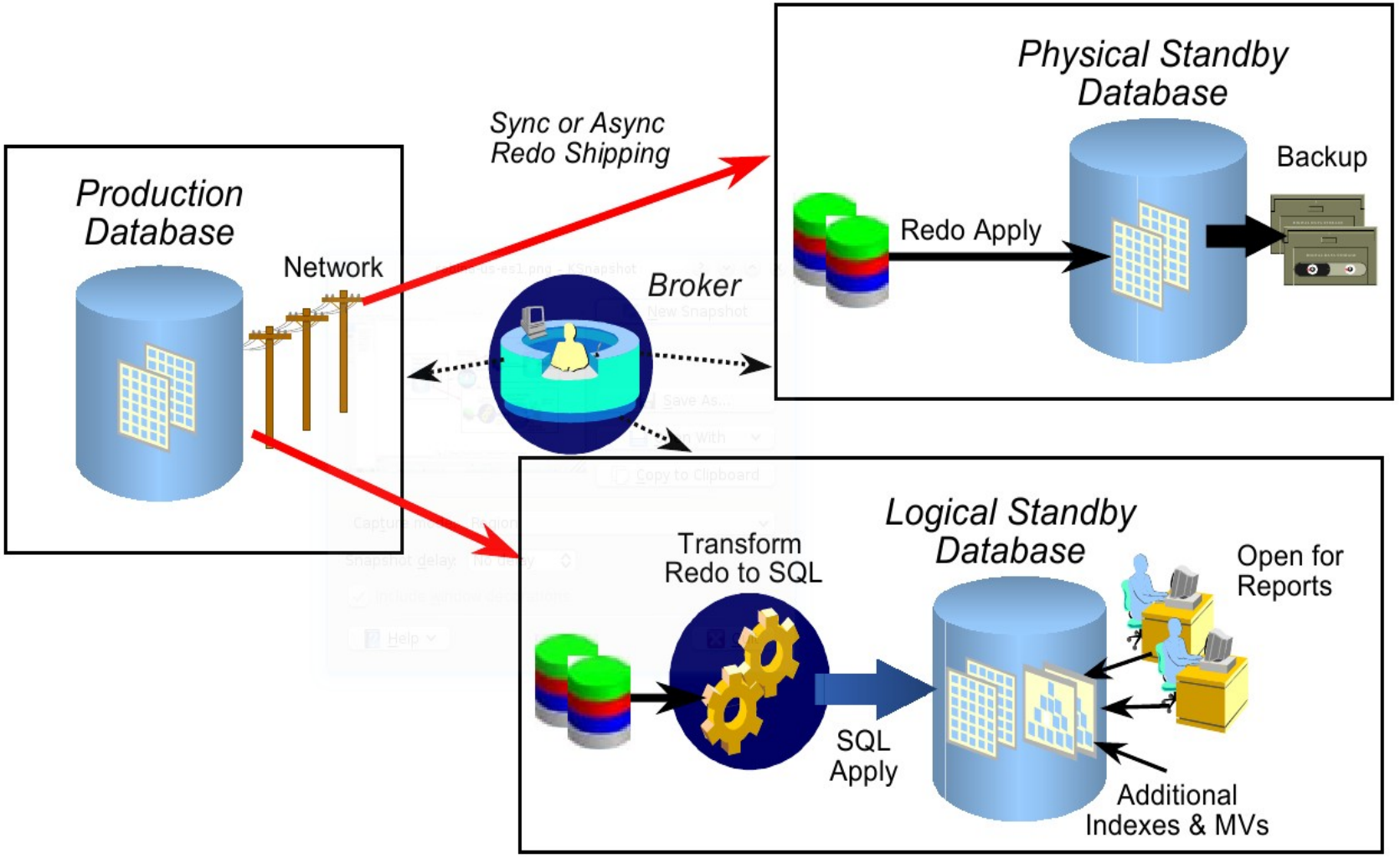
- tables split into data sets, each set modifiable by only one server.
  (e.g. data split (partitioned) by offices, with a server in each office.)
  master/slave replication can be used to keep a read-only copy of the other office's data on each server.

Multiple-Server Parallel Query Execution

- So far – we could see multiple servers processing different queries at the same time.
- In this method, multiple servers handle a single query.
- multiple servers work concurrently on a single query.
- accomplished by splitting the data among servers
- each server executes its part of the query
- central server combines the results and returns the result to the user.

Example: Pgpool-II

ORACLE Data Guard



From: Oracle Data Guard in Roacle Database 10g (Disaster Recovery for the Enterprise), An Oracle White Pater, December 2003