Dokumentation der Synchonisation:

* Atomarität : Ein Problem das auch bei verteilten Datenbanken eine große Rolle spielt ist, dass die Daten auf beiden Seiten der heterogenen DBs gleich ist und nicht inkonsistent.
* Unterschiede in Datenmodelle: Das heißt, dass es zu unterschieden in Datentypen kommen kann. Diese Unterschiede muss man dann vorher abfangen und Parsen damit beide Datenbanken die Values dann auch in die jeweilige DB schreiben
* Unterschiede in Befehlen: Man muss sich zusätzlich auch um das erkennen und ändern der Befehle für beide DBs kümmern.
* Erkennung ob Änderungen in einer der DBs stattgefunden sind. Triggern der synchronisation.
* Erreichbarkeit der Server: sind die Benutzer eingeloggt, kennt die middleware die usernames und die passwörter als auch die location und den Namen der DB.

Connection: Unsere Idee war das die Middlewar passiv ist und mit Trigger oder Listener auf Änderungen in der Datenbank reagiert und die Änderungen dann in die Replikations speichert. Während der synchronisation kann der Benutzer noch weiter in der Datenbank arbeiten, da wir mit einem Resultset arbeiten wollen. Nach der synchronisation muss überprüft werden ob der Benutzer nicht noch weitere Änderungen vorgenommen hat.

Es gebe auch die Möglichkeit nach jeder Änderung in mysql mysqldump aufzurufen, die DB zu dumpen mit compatible = postgres und das erzeugte sql skript dann in postgres zu laden mit hilfe von psql test < mysqloutputfile.sql

Das kann man mit den reinen Daten der db machen oder man kann nur die inserts in postgres einfügen. Bei der zweiten Variante würde das Problem auftreten, dass wir nicht update oder delete beachten.