UNITED STATES DISTRICT COURT
DISTRICT OF DELAWARE

PURE DATA SYSTEMS, LLC	
Plaintiff,	**Civil Action No.**
v.	**JURY TRIAL DEMANDED**
ENTERPRISEDB CORPORATION	
Defendant,	

COMPLAINT FOR PATENT INFRINGEMENT

This is an action for patent infringement arising under the Patent Laws of the United States of America, 35 U.S.C. § 1 et seq. in which Plaintiff Pure Data Systems, LLC ("PDS" or "Plaintiff") files this patent infringement action against Defendant EnterpriseDB Corporation ("EnterpriseDB" or "Defendant").

BACKGROUND

1. Plaintiff PDS is the assignee of all right, title, and interest in and to U.S. Patent No. 5,999,947, entitled "Patent for inventions covering a method of distributing database differences" ("the '947 Patent," attached as Exhibit A), and U.S. Patent No. 6,321,236 ("the '236 Patent," attached as Exhibit B), entitled "Patent for inventions covering a system for distributing differences") (collectively, the "Patents-in-Suit"). PDS has the exclusive right to assert all causes of action arising under the Patents-in-Suit and the right to remedies for infringement thereof, including past infringement.

2. The inventive concepts of the Patents-in-Suit are directed to a technical solution to solve a problem unique to data storage systems, by greatly enhancing and facilitating the operation and efficiency of data storage systems.

3. For example, the inventions are directed to distributing differences from a

server computer, which is a hardware system, configured to store a current version of data, which is distributed and updated over a communications network, which is also a hardware system.

4. The claimed invention further recites receiving a request from a client computer, which is also a hardware system. It further recites translating differences from a generic format (or in other claims a first format) into specific format that is compatible with the type of data on the client computer (or in other claims a second format), and transmitting the differences to the client. This improves the functioning of the data storage system, for example, by efficiently using system resources and permitting client systems that are intermittently (as opposed to continuously) connected to a server system to synchronize with information from the server. (‘947 Patent, col. 1, lines 9-19; ‘236 Patent, col. 1, lines 13-23).

5. Without the claimed invention, data storage systems would, for example, be required to download the entire set of data, which requires large amounts of bandwidth, is expensive, and time consuming. (‘947 Patent, col. 2, lines 1-8; ‘236 Patent, col. 2, lines 5-12.) Without the claimed invention, another drawback is the need to make a dynamic comparison of the client and original database, which requires large amounts of handshaking and data transfer. (‘947 Patent, col. 2, lines 9-17; ‘236 Patent, col. 2, lines 13-21.)

6. The technology claimed in the Patents-in-Suit presented new and unique advantages over the state of the art at the time. Although the inventions taught in the claims of the Patents-in-Suit have by today been widely adopted by leading businesses, at the time of the invention, the technologies were innovative.

7. For example, during prosecution of the application that issued as the ‘947 Patent, the Examiner at the United States Patent and Trademark Office attempted to apply as prior art U.S. Patent No. 5,758,355 to Buchanan to the pending claims. The applicants explained that Applicant further argued that Buchanan does not teach “translating database differences from a generic format into instructions specific to the type of database engine

associated with the client copy," but rather "merely discloses the concept of bi-directional synchronization of a client database and a server database, and does not make any reference to translating database differences at a particular data format." Similarly, during prosecution of the application that issued as the '236 Patent, the applicants distinguished Buchanan on the basis that it does not disclose a database with a translated format.

8. As another example, during prosecution of the application that issued as the '947 Patent, the Examiner at the United States Patent and Trademark Office also attempted to apply U.S. Patent No. 5,634,052 to Morris the pending claims. The applicants explained that in their invention, database differences are transmitted from the server to the client, which enables the client computer to maintain an updated copy of a database table stored at the server. In contrast, Morris discloses a system whereby a delta file, which represents the differences between a base file and a new version of the base tile, is transmitted from the client to the server. While transmitting the delta files from the client to the server enables a file stored at the client to be backed up and archived at the server, this function is significantly different from that of the claimed invention and fails to disclose all the elements of the claim.

9. As another example, during prosecution of the application that issued as the '236 Patent, the Examiner at the United States Patent and Trademark Office attempted to apply U.S. Patent No. 5,870,765 to Bauer to the pending claims. The applicant distinguished the pending claims on the basis that they are directly opposed to the disclosure of the Bauer patent.

10. The claims of the '701 Patent are not directed to a "method of organizing human activity," "fundamental economic practice long prevalent in our system of commerce," or "a building block of the modern economy." Instead, they are limited to technological solutions for data storage systems.

11. Additionally, the technology claimed in the Patents-in-Suit does not preempt

all ways for distributing differences from a server computer. For example, the claims apply only to using different data formats on the server (e.g. a generic format) and client (e.g. a specific format). It follows that Defendant could choose other ways of distributing differences, such as using the same data formats on both the client or server, or by using a specific format on the server and a generic format on the client.

12. Additionally, the prior art cited on the face of the Patents-in-Suit remains available for practice by the Defendant, and the Patents-in-Suit do not preempt practice any of those prior art systems or methods. The claims of the Patents-in-Suit cannot be practiced by a human alone and there exists no human analogue to the methods and systems claimed in the Patents-in-Suit. The claims are specifically directed to distributing data from server computers to client computers. Components such as server and client computer exist only in the context of computer-based systems, and cannot be practiced by a human alone.

13. By practicing a system for distributing differences corresponding to one or more change events, EnterpriseDB has infringed the claims of the Patents-in-Suit.

PARTIES

14. PDS is a Texas Limited Liability Company with a principal place of business at 1400 Preston Road, Suite 400, Plano, Texas 75093.

15. On information and belief, EnterpriseDB is a Delaware Corporation headquartered at 34 Crosby Dr., Suite 201, Bedford, MA 01730. EnterpriseDB may be served with process by delivering a summons and a true and correct copy of this Complaint to its registered agent for receipt of service of process, C T Corporation System, 84 State Street, Boston, MA 02109.

JURISDICTION AND VENUE

16. This action arises under the patent laws of the United States, Title 35 of the United States Code. Accordingly, this Court has subject matter jurisdiction under 28 U.S.C.

§§ 1331 and 1338(a).

17. This Court has personal jurisdiction over EnterpriseDB because, among other reasons, EnterpriseDB has established minimum contacts with the forum state of Delaware. For example, EnterpriseDB is incorporated under the laws of Delaware, and has purposefully availed itself of the benefits of doing business in the State of Delaware such that the exercise of jurisdiction over EnterpriseDB would not offend traditional notions of fair play and substantial justice.

18. Venue is proper in this District under 1400(b) at least because EnterpriseDB is a resident of Delaware.

COUNT I
INFRINGEMENT OF U.S. PATENT NO. 5,999,947

19. Plaintiff incorporates by reference each of the allegations in the foregoing paragraphs, and further alleges as follows:

20. On December 7, 1999, the United States Patent and Trademark Office issued the ‘947 Patent for inventions covering a method of distributing database differences.

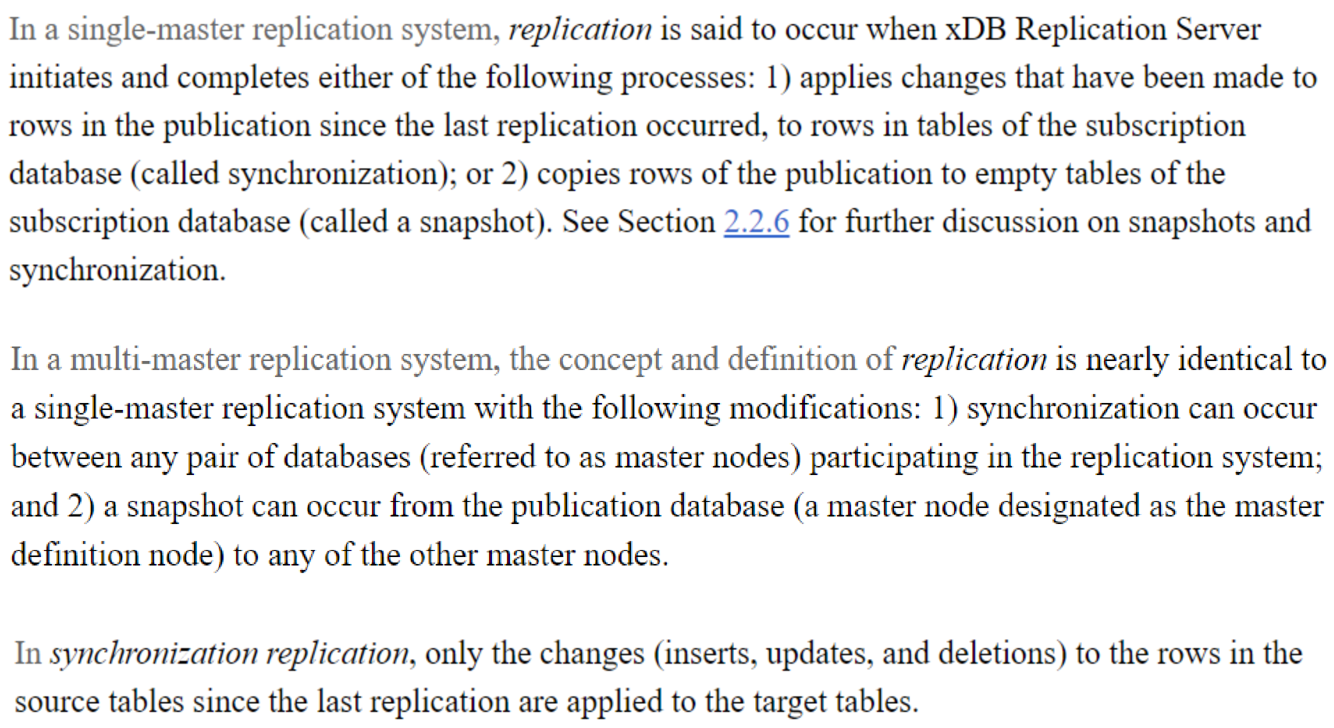
21. In one claimed embodiment, a method of distributing database differences corresponding to database change events made to a database table located on a server computer to client copies of the database table located on one or more client computers, each client computer capable of having different database engines comprising the steps of: storing database differences at the server computer in a generic format; receiving from a client computer a request for all database differences needed to make a client copy of the database table current; translating the differences from the generic format into instructions having a specific format compatible with the type of database engine associated with the client copy of the database table; and transmitting the instructions to the client computer for execution on the client database engine to make the client copy of the database table current.. A true and

correct copy of the '304 Patent is attached as Exhibit A.

22. EnterpriseDB has directly and indirectly infringed one or more claims of the '947 Patent, in this judicial District and elsewhere in the United States.

23. For example, EnterpriseDB has directly infringed the '947 Patent, including but not limited to claim 6, by practicing a method of distributing database differences corresponding to database change events, according to the claims of the '947 Patent. The following analysis is exemplary only, and EnterpriseDB reserves the right to make additional or different allegations in its infringement contentions.

24. For example, EnterpriseDB distributes database differences corresponding to database change events made to a database table located on a server computer to client copies of the database table located on one or more client computers, each client computer capable of having different database engines. For example, EnterpriseDB replication systems distribute database change events from a server to a client, e.g.:



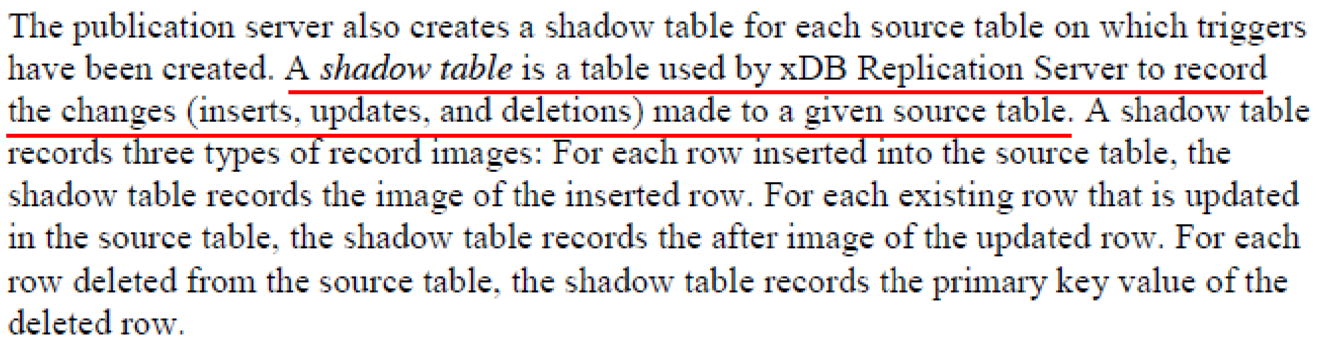
In a single-master replication system, *replication* is said to occur when xDB Replication Server initiates and completes either of the following processes: 1) applies changes that have been made to rows in the publication since the last replication occurred, to rows in tables of the subscription database (called synchronization); or 2) copies rows of the publication to empty tables of the subscription database (called a snapshot). See Section 2.2.6 for further discussion on snapshots and synchronization.

In a multi-master replication system, the concept and definition of *replication* is nearly identical to a single-master replication system with the following modifications: 1) synchronization can occur between any pair of databases (referred to as master nodes) participating in the replication system; and 2) a snapshot can occur from the publication database (a master node designated as the master definition node) to any of the other master nodes.

In *synchronization replication*, only the changes (inserts, updates, and deletions) to the rows in the source tables since the last replication are applied to the target tables.

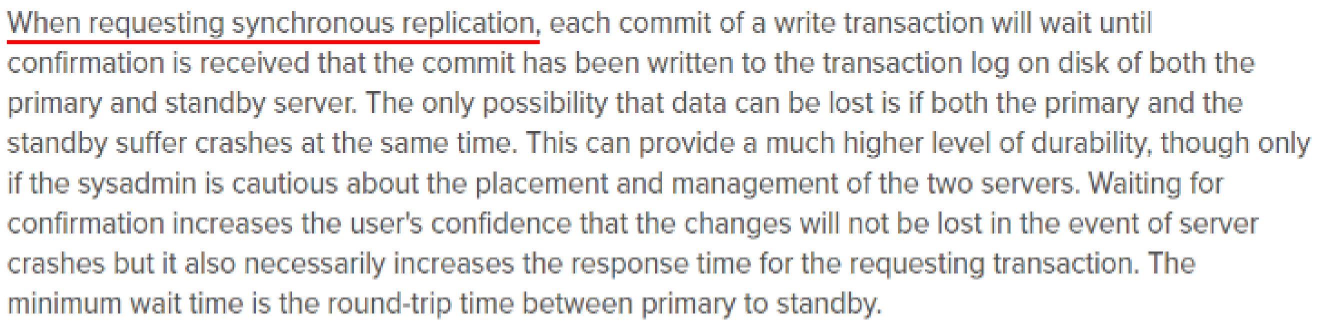
https://www.enterprisedb.com/docs/en/5.1/repguide/EDB_Postgres_Replication_Server_Users_Guide.1.09.html

25. EnterpriseDB stores database differences at the server computer in a generic format. For example, database differences are stored in a generic format at a server computer, e.g.:

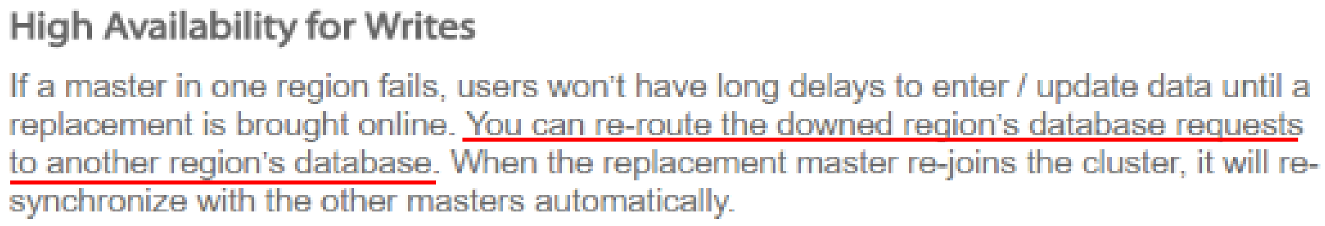

The publication server also creates a shadow table for each source table on which triggers have been created. A *shadow table* is a table used by xDB Replication Server to record the changes (inserts, updates, and deletions) made to a given source table. A shadow table records three types of record images: For each row inserted into the source table, the shadow table records the image of the inserted row. For each existing row that is updated in the source table, the shadow table records the after image of the updated row. For each row deleted from the source table, the shadow table records the primary key value of the deleted row.

http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf

26. EnterpriseDB receives from a client computer a request for all database differences needed to make a client copy of the database table current. For example, a request from a client computer is received to update the client copy of the database, e.g.:

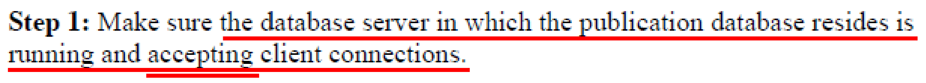

When requesting synchronous replication, each commit of a write transaction will wait until confirmation is received that the commit has been written to the transaction log on disk of both the primary and standby server. The only possibility that data can be lost is if both the primary and the standby suffer crashes at the same time. This can provide a much higher level of durability, though only if the sysadmin is cautious about the placement and management of the two servers. Waiting for confirmation increases the user's confidence that the changes will not be lost in the event of server crashes but it also necessarily increases the response time for the requesting transaction. The minimum wait time is the round-trip time between primary to standby.

https://www.enterprisedb.com/docs/en/9.6/pg/warm-standby.html

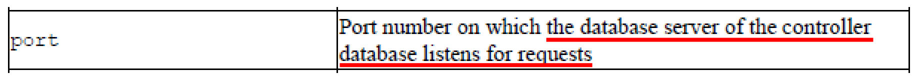

High Availability for Writes

If a master in one region fails, users won't have long delays to enter / update data until a replacement is brought online. You can re-route the downed region's database requests to another region's database. When the replacement master re-joins the cluster, it will re-synchronize with the other masters automatically.

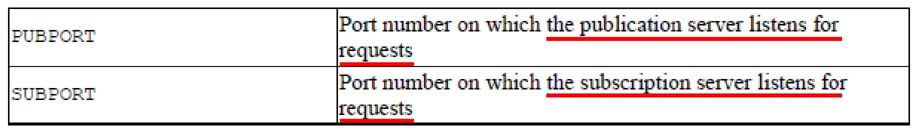
http://translate.enterprisedb.com/products-services-training/products-overview/xdb-replication-server-multi-master



Step 1: Make sure the database server in which the publication database resides is running and accepting client connections.



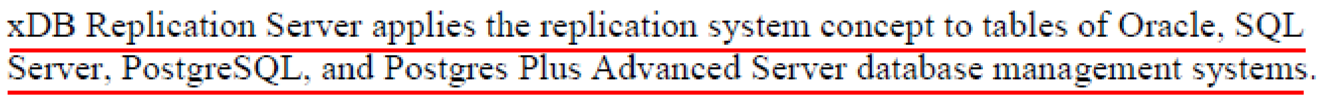
`port`	Port number on which the database server of the controller database listens for requests



`PUBPORT`	Port number on which the publication server listens for requests
`SUBPORT`	Port number on which the subscription server listens for requests

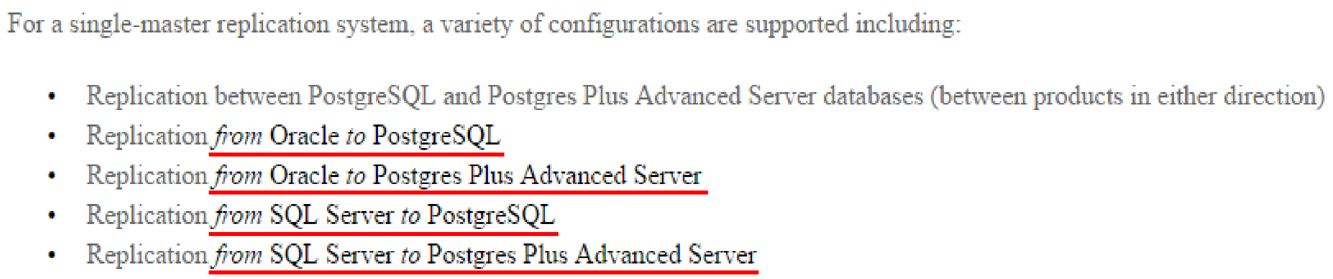
http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf

27. EnterpriseDB translates the differences from the generic format into instructions having a specific format compatible with the type of database engine associated with the client copy of the database table. For example, differences are translated from the generic format to a specific format compatible with the database of the client, e.g.:



xDB Replication Server applies the replication system concept to tables of Oracle, SQL Server, PostgreSQL, and Postgres Plus Advanced Server database management systems.

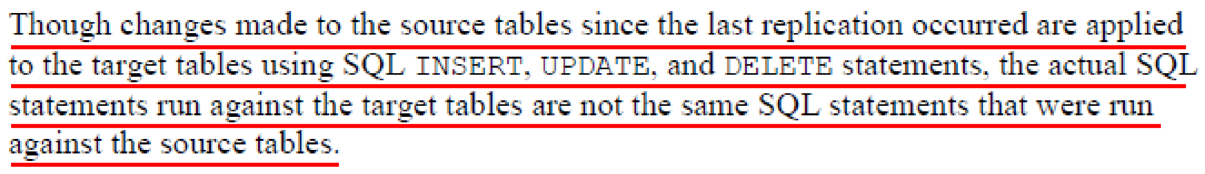
http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf



For a single-master replication system, a variety of configurations are supported including:

- Replication between PostgreSQL and Postgres Plus Advanced Server databases (between products in either direction)
- Replication *from* Oracle *to* PostgreSQL
- Replication *from* Oracle *to* Postgres Plus Advanced Server
- Replication *from* SQL Server *to* PostgreSQL
- Replication *from* SQL Server *to* Postgres Plus Advanced Server

https://www.enterprisedb.com/docs/en/6.0/repguide/EDB_Postgres_Replication_Server_Users



Though changes made to the source tables since the last replication occurred are applied to the target tables using SQL `INSERT`, `UPDATE`, and `DELETE` statements, the actual SQL statements run against the target tables are not the same SQL statements that were run against the source tables.

http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf

In the *trigger-based method* changes to rows in the source tables result in the firing of row-based triggers. These triggers record the changes in shadow tables. The changes recorded in the shadow tables are then periodically extracted from the shadow tables, converted to an in-memory data structure, and applied to the target tables by means of SQL statements executed using JDBC. See Section 2.2.9 for information on the trigger-based method.

https://www.enterprisedb.com/docs/en/6.0/repguide/EDB_Postgres_Replication_Server_Users_Guide.1.09.html

When synchronization replication occurs, the publication server executes JDBC batches of SQL statements (also referred to as *transaction sets*) against the target tables. The batches contain an `INSERT` statement for each shadow table row recording an insert operation, an `UPDATE` statement for each shadow table row recording an update operation, and a `DELETE` statement for each shadow table row recording a delete operation. Each batch is executed in one transaction.

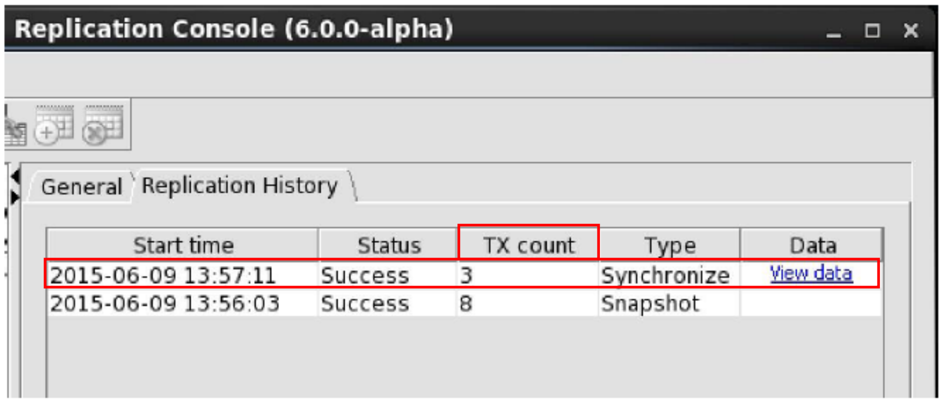
http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf

Note: A single SQL statement executed against a source table may result in many rows recorded in a shadow table, and therefore, many SQL statements executed against the target table. For example, if a single `UPDATE` statement affects 10 rows in the source table, 10 rows will be inserted into the shadow table – one for each row in the source table that was updated. When the publication server applies the changes to the target table, 10 `UPDATE` statements will be executed.

http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf

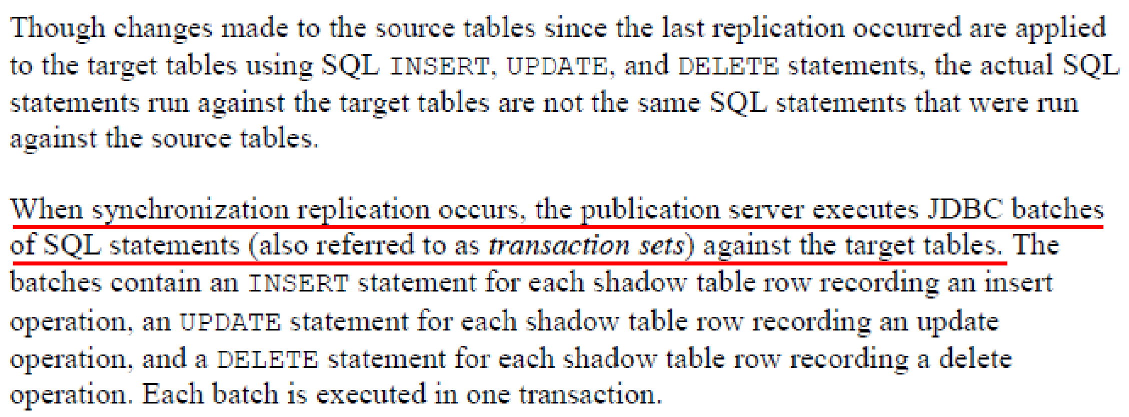
28. EnterpriseDB transmits the instructions to the client computer for execution on the client database engine to make the client copy of the database table current. For example, instructions are transmitted to the client computer to update the client database table, e.g.:

Step 2: Click the Replication History tab to show a history of replications for this table.



Start time	Status	TX count	Type	Data
2015-06-09 13:57:11	Success	3	Synchronize	View data
2015-06-09 13:56:03	Success	8	Snapshot	

Figure 7-42 - Table replication history tab



Though changes made to the source tables since the last replication occurred are applied to the target tables using SQL INSERT, UPDATE, and DELETE statements, the actual SQL statements run against the target tables are not the same SQL statements that were run against the source tables.

When synchronization replication occurs, the publication server executes JDBC batches of SQL statements (also referred to as *transaction sets*) against the target tables. The batches contain an INSERT statement for each shadow table row recording an insert operation, an UPDATE statement for each shadow table row recording an update operation, and a DELETE statement for each shadow table row recording a delete operation. Each batch is executed in one transaction.

http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf

29. By practicing a system for distributing differences corresponding to one or more change events, EnterpriseDB is infringing the claims of the ’947 Patent, including but not limited to claim 6. EnterpriseDB has committed these acts of infringement without license or authorization.

30. EnterpriseDB has injured PDS and is liable to PDS for direct and indirect infringement of the claims of the ‘947 Patent pursuant to 35 U.S.C. § 271(a), (b), and (c).

31. As a result of Defendant’s infringement of the ’947 Patent, PDS has suffered harm and seeks monetary damages in an amount adequate to compensate for infringement, but in no event less than a reasonable royalty for the use made of the invention by EnterpriseDB, together with interest and costs as fixed by the Court.

COUNT II

INFRINGEMENT OF U.S. PATENT NO. 6,321,236

32. Plaintiff incorporates by reference each of the allegations in the foregoing paragraphs, and further alleges as follows:

33. On November 20, 2001, the United States Patent and Trademark Office issued

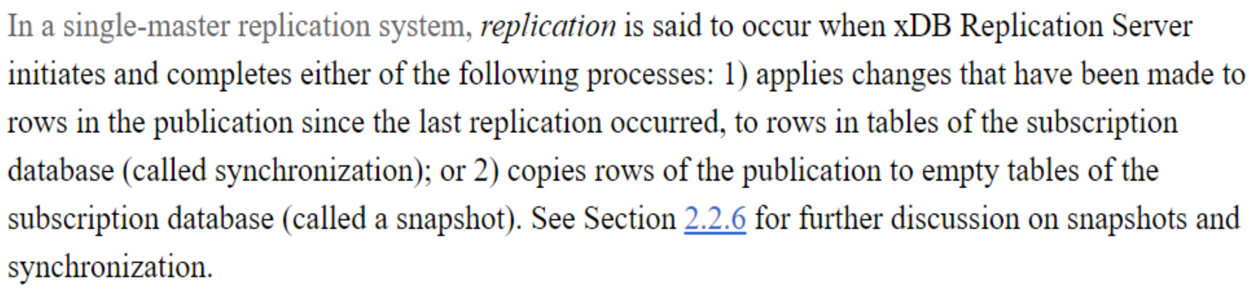
the ‘236 Patent for inventions covering a system for distributing differences. One claimed embodiment recites a system for distributing differences corresponding to one or more change events made to a data store located on a server computer, the differences being distributed to one or more client copies of at least a portion of the data store, wherein the one or more client copies of the at least a portion of the data store are located on one or more client computers, the system comprising: a current server version of the data store configured to permit modifications to data contained therein; a reference server version of the data store; a differencing engine that identifies, at a given instance in time, any differences between the current server version of the data store and the reference server version of the data store; one or more updates storing one or more differences generated by the differencing engine wherein the one or more differences are in a first format; a translator that converts any differences destined for the client copy of the at least a portion of the data store from the first format into a second format; a communication network; and a synchronizer that obtains from the differencing engine any differences that are needed to make the one or more client copies of the at least a portion of the data store current, and transmits the differences to the one or more client copies of the at least a portion of the data store by way of the communication network.” A true and correct copy of the ‘236 Patent is attached as Exhibit B.

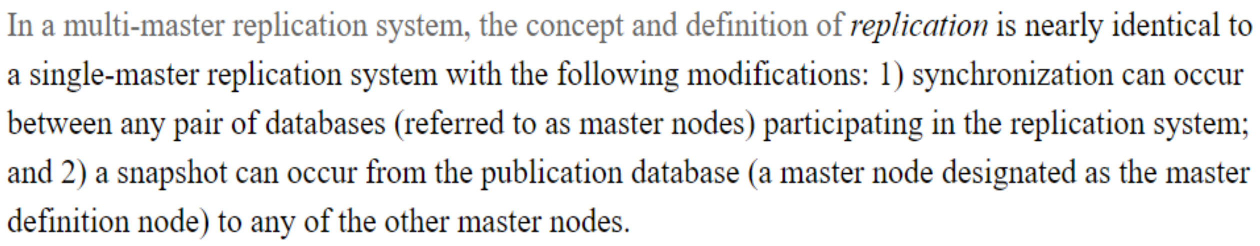
34. EnterpriseDB has been and is now directly and indirectly infringing one or more claims of the ‘236 Patent, in this judicial District and elsewhere in the United States.

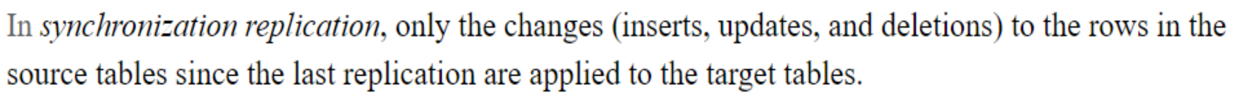
35. For example, EnterpriseDB directly infringes the ‘236 Patent, including but not limited to claim 1, by practicing a system for distributing differences corresponding to one or more change events, according to the claims of the ‘236 Patent. The following analysis is exemplary only, and EnterpriseDB reserves the right to make additional or different allegations in its infringement contentions.

36. EnterpriseDB makes, uses and offers a system for distributing differences corresponding to one or more change events made to a data store located on a server

computer, the differences being distributed to one or more client copies of at least a portion of the data store, wherein the one or more client copies of the at least a portion of the data store are located on one or more client computers. For example, EnterpriseDB makes, uses, and offers a system for distributing database change events from a server to a client, e.g.:


In a single-master replication system, *replication* is said to occur when xDB Replication Server initiates and completes either of the following processes: 1) applies changes that have been made to rows in the publication since the last replication occurred, to rows in tables of the subscription database (called synchronization); or 2) copies rows of the publication to empty tables of the subscription database (called a snapshot). See Section 2.2.6 for further discussion on snapshots and synchronization.


In a multi-master replication system, the concept and definition of *replication* is nearly identical to a single-master replication system with the following modifications: 1) synchronization can occur between any pair of databases (referred to as master nodes) participating in the replication system; and 2) a snapshot can occur from the publication database (a master node designated as the master definition node) to any of the other master nodes.


In *synchronization replication*, only the changes (inserts, updates, and deletions) to the rows in the source tables since the last replication are applied to the target tables.

https://www.enterprisedb.com/docs/en/5.1/repguide/EDB_Postgres_Replication_Server_Users_Guide.1.09.html

37. The EnterpriseDB system comprises a current server version of the data store configured to permit modifications to data contained therein; a reference server version of the data store; a differencing engine that identifies, at a given instance in time, any differences between the current server version of the data store and the reference server version of the data store. For example, EnterpriseDB's system detects differences between current and reference versions of a data store, e.g.:

If during synchronization replication, conflicting changes are pending against the same row from different master nodes, the conflict resolution strategy determines which of the conflicting changes is accepted and replicated to all master nodes. The conflicting changes that are not accepted are discarded.

- When a conflict is detected, the conflict information such as the transaction ID, conflict type, and conflict detection timestamp are logged in the conflict table on the target master node.
- For a conflicting transaction, the replication server checks if any conflict resolution strategy has been selected for the specific table. If a strategy is found, it is applied accordingly and the conflict status is marked as *resolved*. If a strategy cannot be applied, the conflict status is marked as *unresolved* (also called *pending*).

The following is a brief summary of each conflict resolution strategy:

- **Earliest Timestamp.** The conflicting change with the earliest timestamp is accepted and replicated to all other master nodes. All other conflicting changes are discarded.
- **Latest Timestamp.** The conflicting change with the latest timestamp is accepted and replicated to all other master nodes. All other conflicting changes are discarded.

http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf

6.6.7 Automatic Conflict Resolution Example

This example illustrates a scenario where a transaction change originating from the first master node is successfully applied to the second master node, but conflicts with the third master node. The conflict is resolved automatically.

The conflict resolution option is set to latest timestamp.

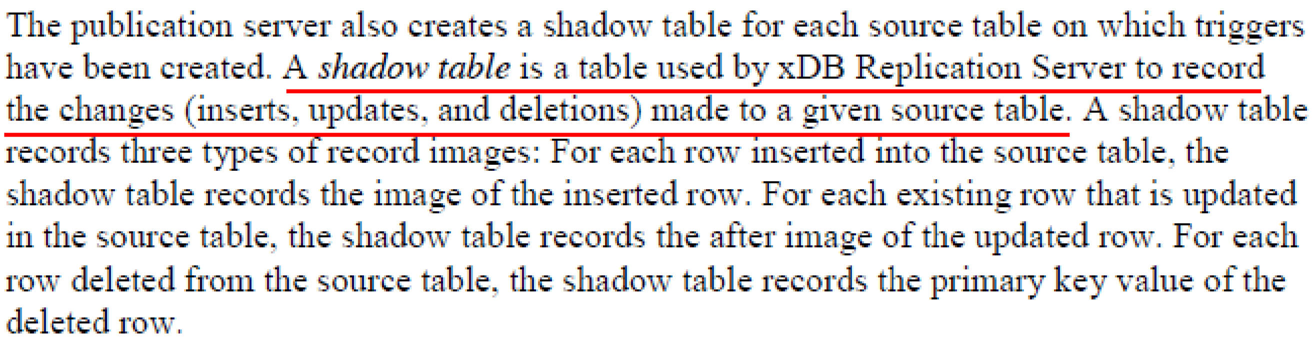
Table 6-5 – Automatic Conflict Resolution Example

Timestamp	Action	Master Node A	Master Node B	Master Node C
t0		id = 2, address = 'ADDR'	id = 2, address = 'ADDR'	id = 2, address = 'ADDR'
t1	**Node A:** UPDATE addrbook SET address = 'ADDR A' WHERE id = 2;	id = 2, address = **'ADDR A'**	id = 2, address = 'ADDR'	id = 2, address = 'ADDR'
t2	**Node C:** UPDATE addrbook SET address = 'ADDR C' WHERE id = 2;	id = 2, address = 'ADDR A'	id = 2, address = 'ADDR'	id = 2, address = **'ADDR C'**
t3	Synchronization pushes Node A changes to Node B. Changes successfully applied.	id = 2, address = 'ADDR A'	id = 2, address = **'ADDR A'**	id = 2, address = 'ADDR C'
t4	Synchronization pushes Node A changes to Node C. Current address on Node C <> old value on Node A ('ADDR C' <> 'ADDR') hence conflict detected. Latest change on Node C accepted and Node A change discarded.	id = 2, address = 'ADDR A'	id = 2, address = 'ADDR A'	id = 2, address = **'ADDR C'**
t5	No changes on Node B. Node C changes pushed to Node A that is successfully applied	id = 2, address = **'ADDR C'**	id = 2, address = 'ADDR A'	id = 2, address = 'ADDR C'
	(Node A change already marked as discarded and hence is overwritten.)			
t6	Node C changes pushed to Node B that is successfully applied. All nodes are in sync and have consistent state.	id = 2, address = 'ADDR C'	id = 2, address = **'ADDR C'**	id = 2, address = 'ADDR C'

In this scenario, Node C contains the current version of the data store and Node A has a reference version of the data store.

http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf

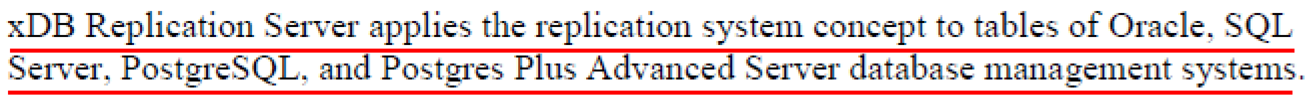
38. The EnterpriseDB system comprises one or more updates storing one or more differences generated by the differencing engine wherein the one or more differences are in a first format. For example, the updates contain differences stored in a first format, e.g.:



The publication server also creates a shadow table for each source table on which triggers have been created. A *shadow table* is a table used by xDB Replication Server to record the changes (inserts, updates, and deletions) made to a given source table. A shadow table records three types of record images: For each row inserted into the source table, the shadow table records the image of the inserted row. For each existing row that is updated in the source table, the shadow table records the after image of the updated row. For each row deleted from the source table, the shadow table records the primary key value of the deleted row.

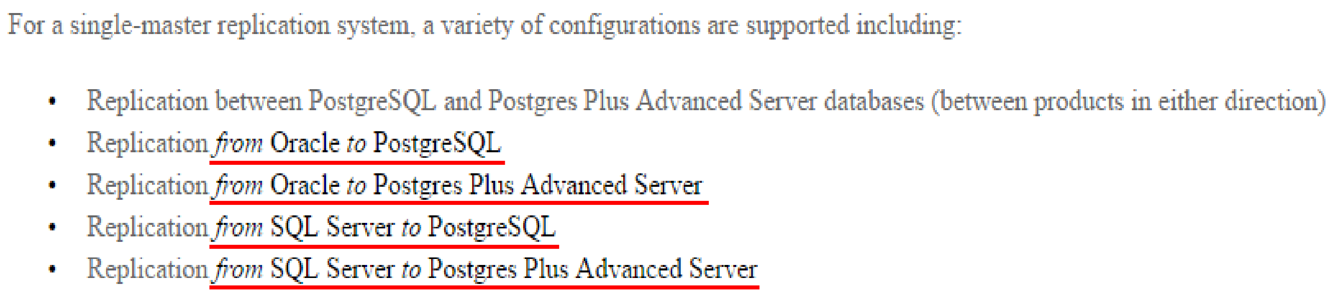
http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf

39. The EnterpriseDB system comprises a translator that converts any differences destined for the client copy of the at least a portion of the data store from the first format into a second format. For example, the differences are translated into a format destined for a client copy, e.g.:



xDB Replication Server applies the replication system concept to tables of Oracle, SQL Server, PostgreSQL, and Postgres Plus Advanced Server database management systems.

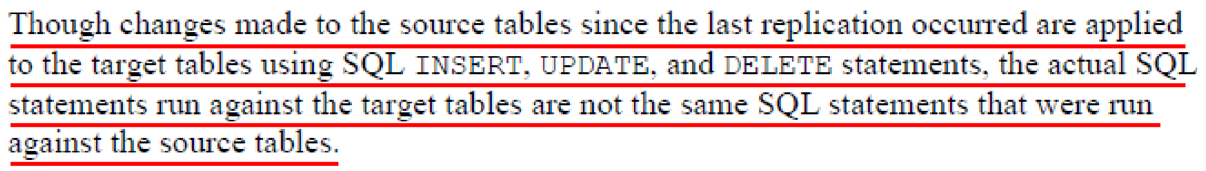
http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf



For a single-master replication system, a variety of configurations are supported including:

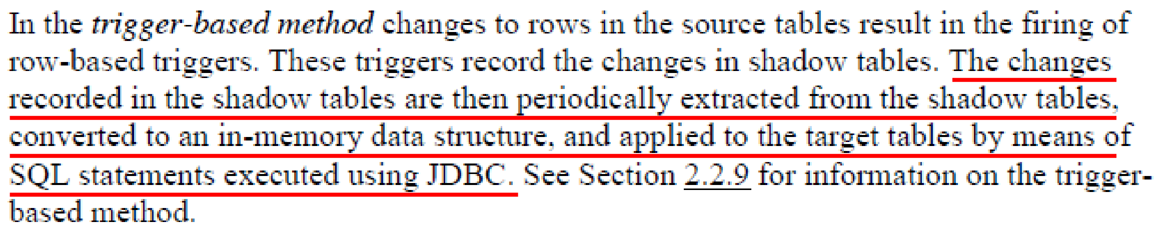
- Replication between PostgreSQL and Postgres Plus Advanced Server databases (between products in either direction)
- Replication *from* Oracle *to* PostgreSQL
- Replication *from* Oracle *to* Postgres Plus Advanced Server
- Replication *from* SQL Server *to* PostgreSQL
- Replication *from* SQL Server *to* Postgres Plus Advanced Server

https://www.enterprisedb.com/docs/en/6.0/repguide/EDB_Postgres_Replication_Server_Users_Guide.1.09.html



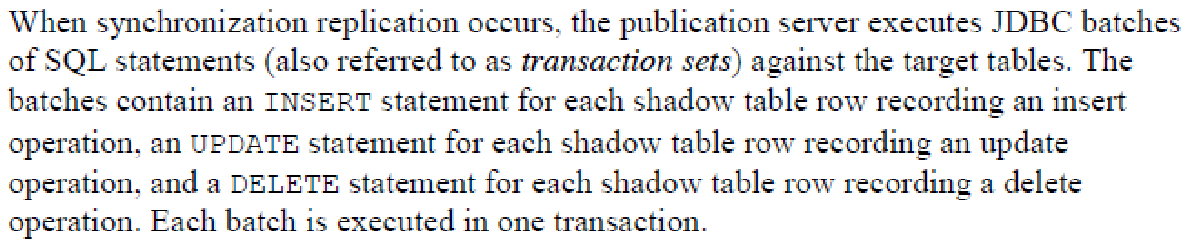
Though changes made to the source tables since the last replication occurred are applied to the target tables using SQL INSERT, UPDATE, and DELETE statements, the actual SQL statements run against the target tables are not the same SQL statements that were run against the source tables.

http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf



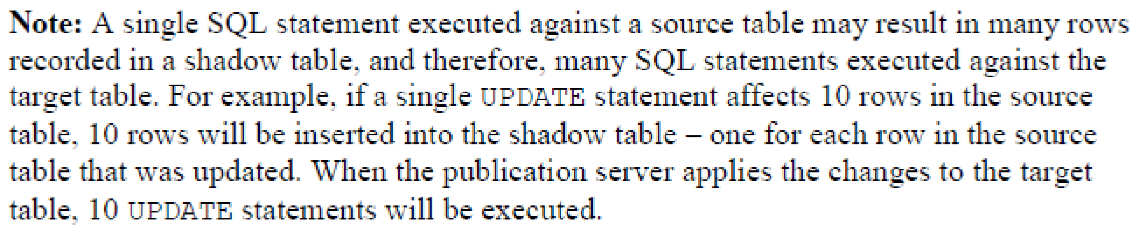
In the *trigger-based method* changes to rows in the source tables result in the firing of row-based triggers. These triggers record the changes in shadow tables. The changes recorded in the shadow tables are then periodically extracted from the shadow tables, converted to an in-memory data structure, and applied to the target tables by means of SQL statements executed using JDBC. See Section 2.2.9 for information on the trigger-based method.

https://www.enterprisedb.com/docs/en/6.0/repguide/EDB_Postgres_Replication_Server_Users_Guide.1.09.html



When synchronization replication occurs, the publication server executes JDBC batches of SQL statements (also referred to as *transaction sets*) against the target tables. The batches contain an INSERT statement for each shadow table row recording an insert operation, an UPDATE statement for each shadow table row recording an update operation, and a DELETE statement for each shadow table row recording a delete operation. Each batch is executed in one transaction.

http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf



Note: A single SQL statement executed against a source table may result in many rows recorded in a shadow table, and therefore, many SQL statements executed against the target table. For example, if a single UPDATE statement affects 10 rows in the source table, 10 rows will be inserted into the shadow table – one for each row in the source table that was updated. When the publication server applies the changes to the target table, 10 UPDATE statements will be executed.

http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf

40. The EnterpriseDB system comprises a communication network. For example, EnterpriseDB's system requires the use of a communication network, e.g.:

- The publication and subscription databases may be in different geographic locations, thereby requiring multiple networked hosts.

2.4.5 Distributed Replication

xDB Replication Server provides the flexibility of allowing you to run the replication system's components on separate machines on a network.

https://www.enterprisedb.com/docs/en/6.0/repguide/EDB_Postgres_Replication_Server_Users_Guide.1.11.html

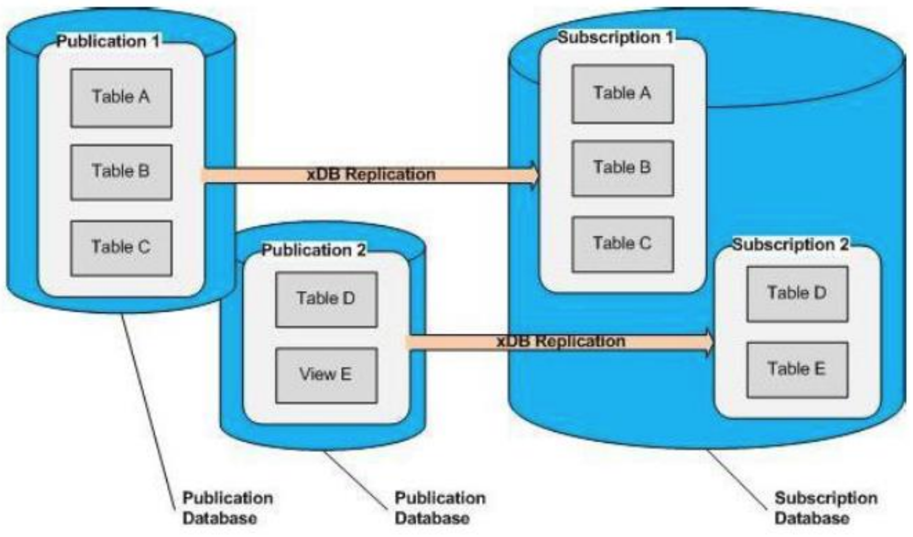


Figure 2-3 - Publications in two databases replicating to one subscription database

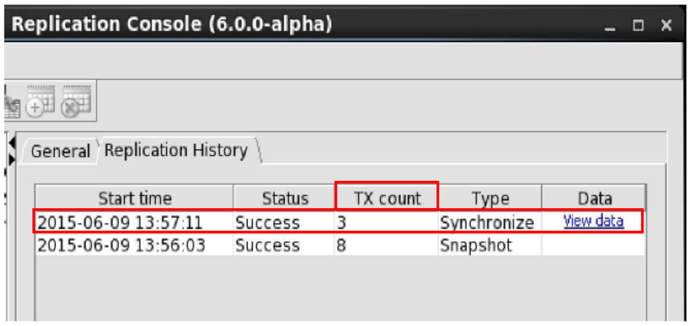
2.1.7 Localized Data Access

In a geographically dispersed application, local access to the database can be provided to regions of clients. Having the database servers physically close to clients can reduce latency with the database. Multi-master replication allows you to employ a WAN connected network of master databases that can be geographically close to groups of clients, yet maintain data consistency across master databases.

http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_Users_Guide_v6.0-3.pdf

41. The EnterpriseDB system comprises a synchronizer that obtains from the differencing engine any differences that are needed to make the one or more client copies of the at least a portion of the data store current, and transmits the differences to the one or more client copies of the at least a portion of the data store by way of the communication network. For example, differences are transmitted to the client for execution to update the client database.

Step 2: Click the Replication History tab to show a history of replications for this table.


Replication Console (6.0.0-alpha)

General | Replication History

Start time	Status	TX count	Type	Data
2015-06-09 13:57:11	Success	3	Synchronize	View data
2015-06-09 13:56:03	Success	8	Snapshot	

Figure 7-42 - Table replication history tab

Though changes made to the source tables since the last replication occurred are applied to the target tables using SQL `INSERT`, `UPDATE`, and `DELETE` statements, the actual SQL statements run against the target tables are not the same SQL statements that were run against the source tables.

When synchronization replication occurs, the publication server executes JDBC batches of SQL statements (also referred to as *transaction sets*) against the target tables. The batches contain an `INSERT` statement for each shadow table row recording an insert operation, an `UPDATE` statement for each shadow table row recording an update operation, and a `DELETE` statement for each shadow table row recording a delete operation. Each batch is executed in one transaction.

http://get.enterprisedb.com/docs/EDB_Postgres_Replication_Server_User
s_Guide_v6.0-3.pdf

42. By making, using and offering a system for distributing differences corresponding to one or more change events, EnterpriseDB is infringing the claims of the ’236 Patent, including but not limited to claim 1. EnterpriseDB has committed these acts of

infringement without license or authorization.

43. EnterpriseDB has injured PDS and is liable to PDS for direct and indirect infringement of the claims of the ‘236 Patent pursuant to 35 U.S.C. § 271(a), (b), and (c).

44. As a result of Defendant’s infringement of the ’236 Patent, PDS has suffered harm and seeks monetary damages in an amount adequate to compensate for infringement, but in no event less than a reasonable royalty for the use made of the invention by EnterpriseDB, together with interest and costs as fixed by the Court.

PRAYER FOR RELIEF

Plaintiff respectfully requests the following relief from the Court:

1. That Defendant has directly and indirectly infringed the Patents-in-Suit;

2. That Defendant be ordered to pay damages to PDS, together with costs, expenses, pre-judgment, interest and post-judgment interest as allowed by law;

3. That the Court enter judgment against Defendant, and in favor of PDS in all respects; and

4. For any such other and further relief as the Court deems just and equitable.

JURY TRIAL DEMANDED

5. Pursuant to Rule 38 of the Federal Rules of Civil Procedure, PDS requests a trial by jury of any issues so triable by right.

Dated: June 30, 2017

Respectfully submitted,

STAMOULIS & WEINBLATT LLC

/s/ Stamatios Stamoulis

Stamatios Stamoulis #4606
stamoulis@swdelaw.com
Richard C. Weinblatt #5080
weinblatt@swdelaw.com
Two Fox Point Centre
6 Denny Road, Suite 307
Wilmington, DE 19809
(302) 999-1540

Attorneys for Plaintiff