Postgres-XC

July 12th, 2011
Koichi Suzuki
NTT DATA INTELLILINK CORPORATION

Overview of Postgres-XC

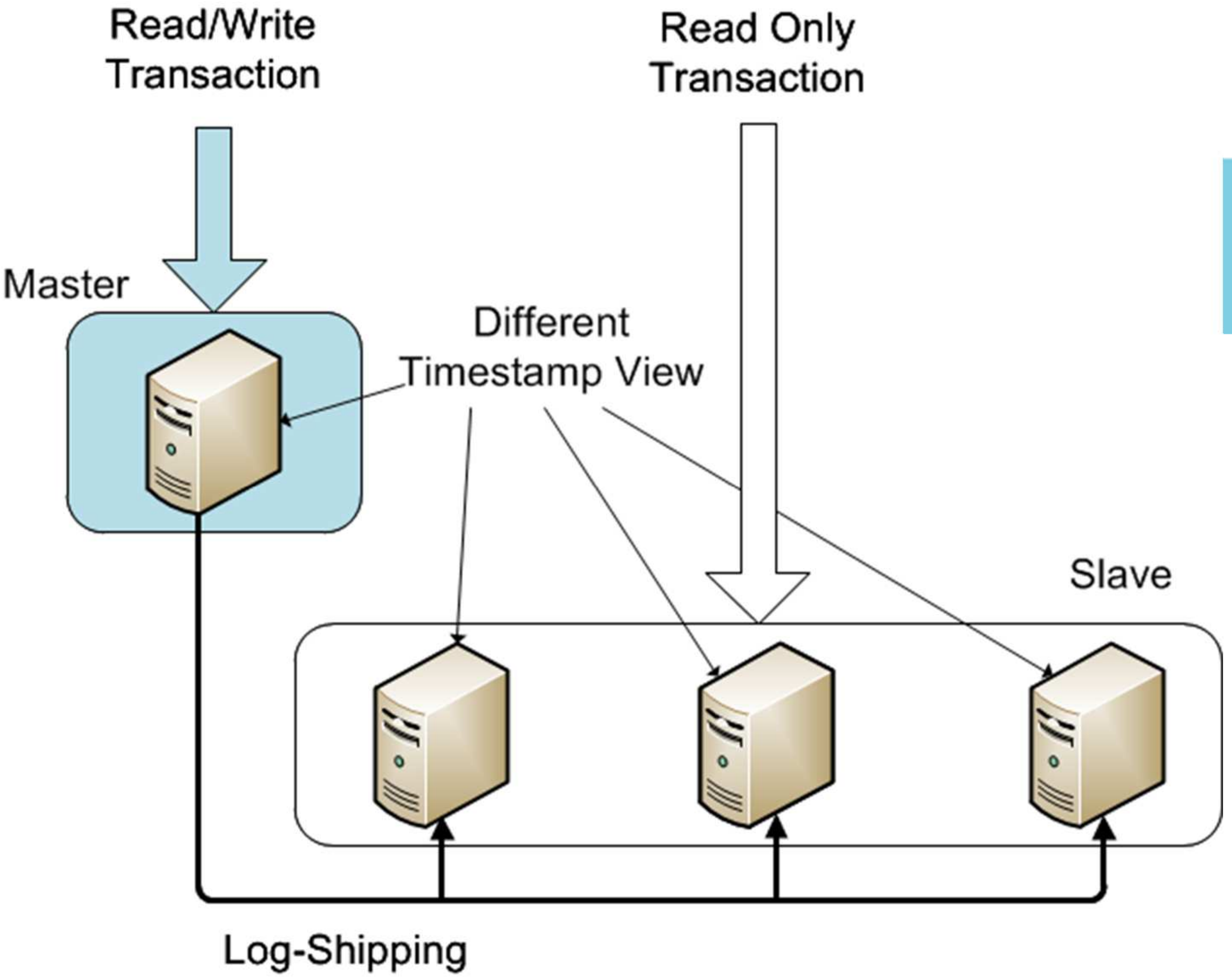
Symmetric PostgreSQL cluster

- No Master
- No Slave
  - No READ ONLY slaves
  - Every node can issue both READ/WRITE
- Transparent Transaction Management

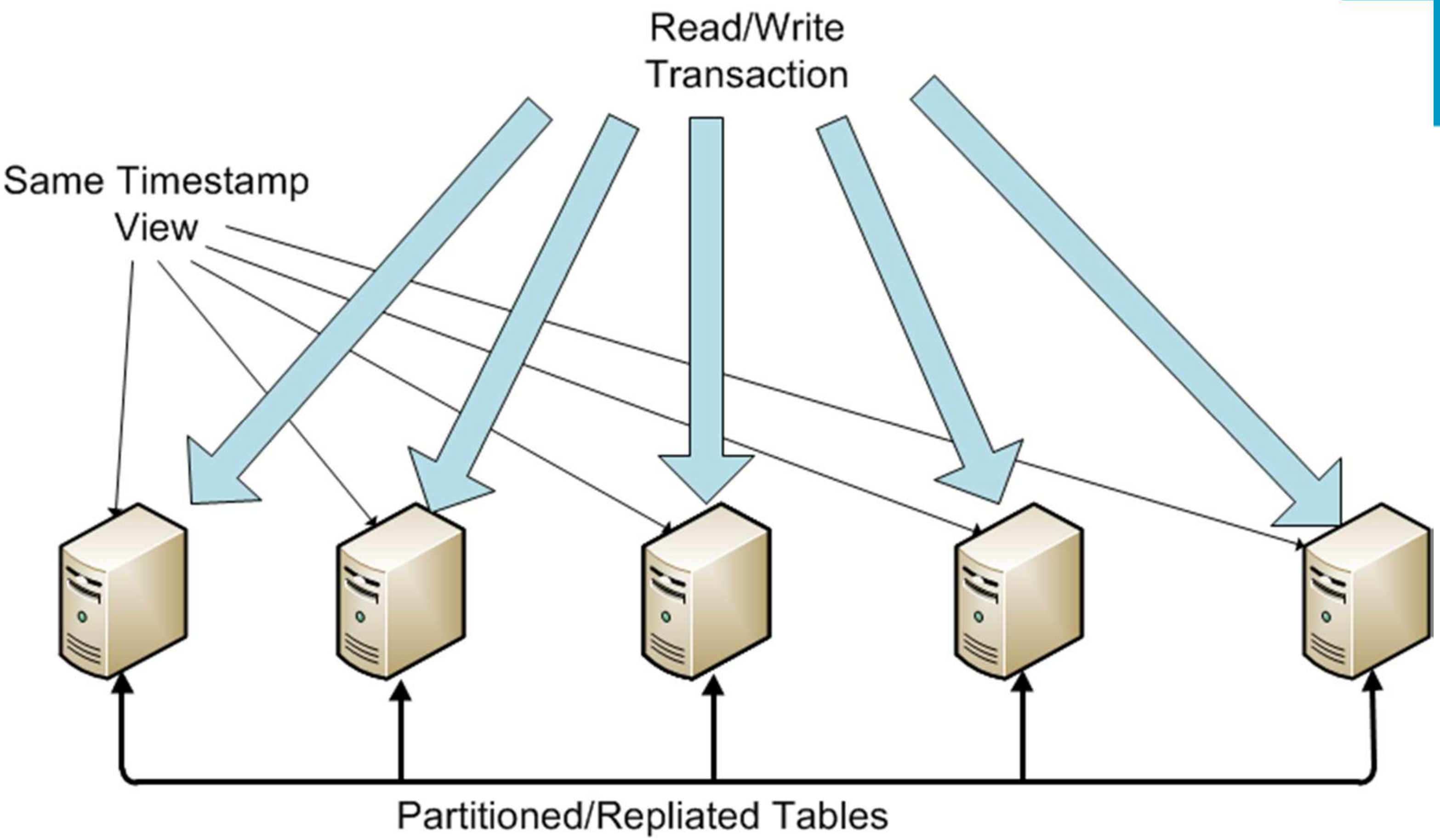
Now Version 0.9.5

- Generally available next calendar year

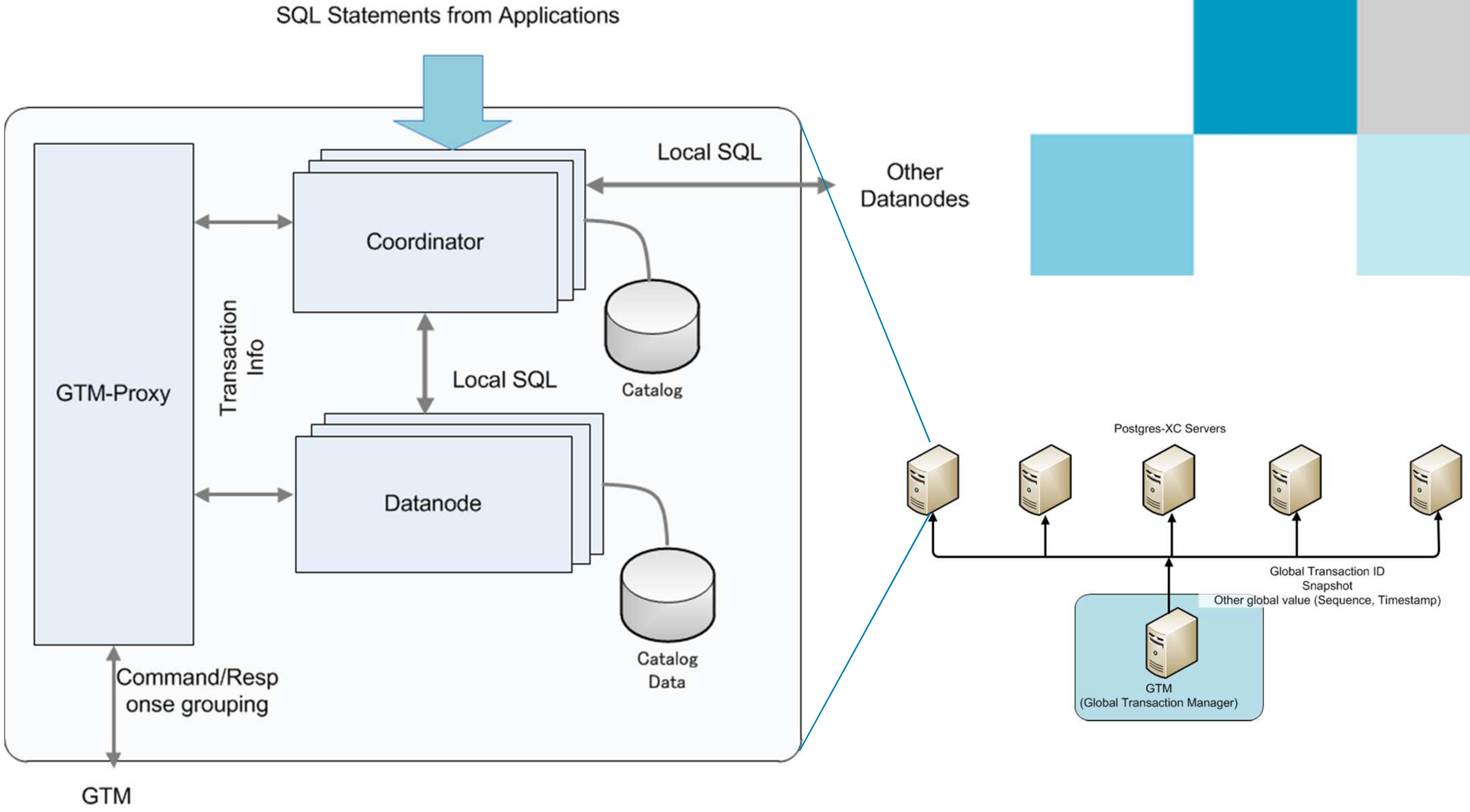
PostgreSQL Master/Slave with Log Shipping



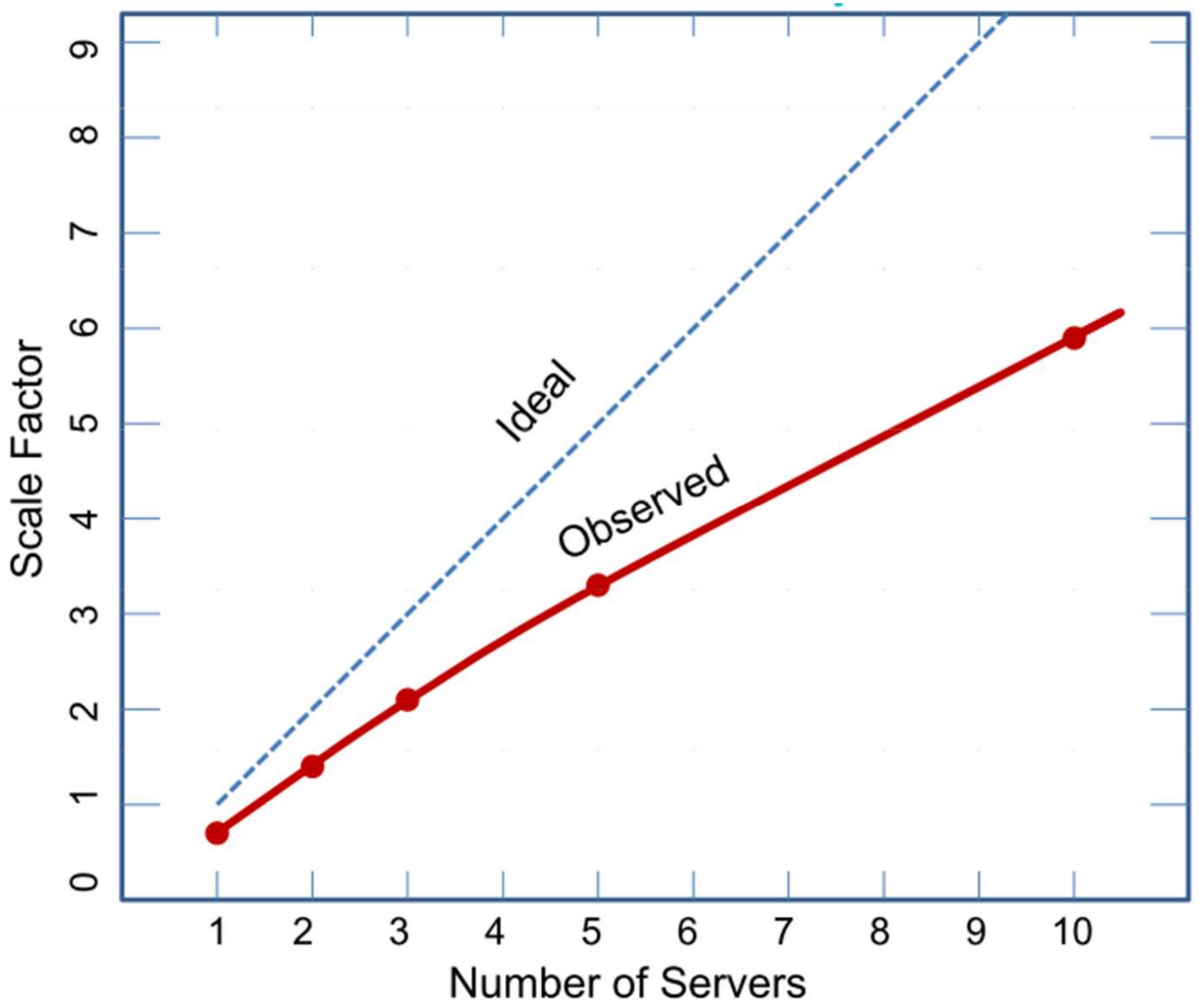
Postgres-XC Symmetric Cluster



Server Configuration and GTM-Proxy



Scalability



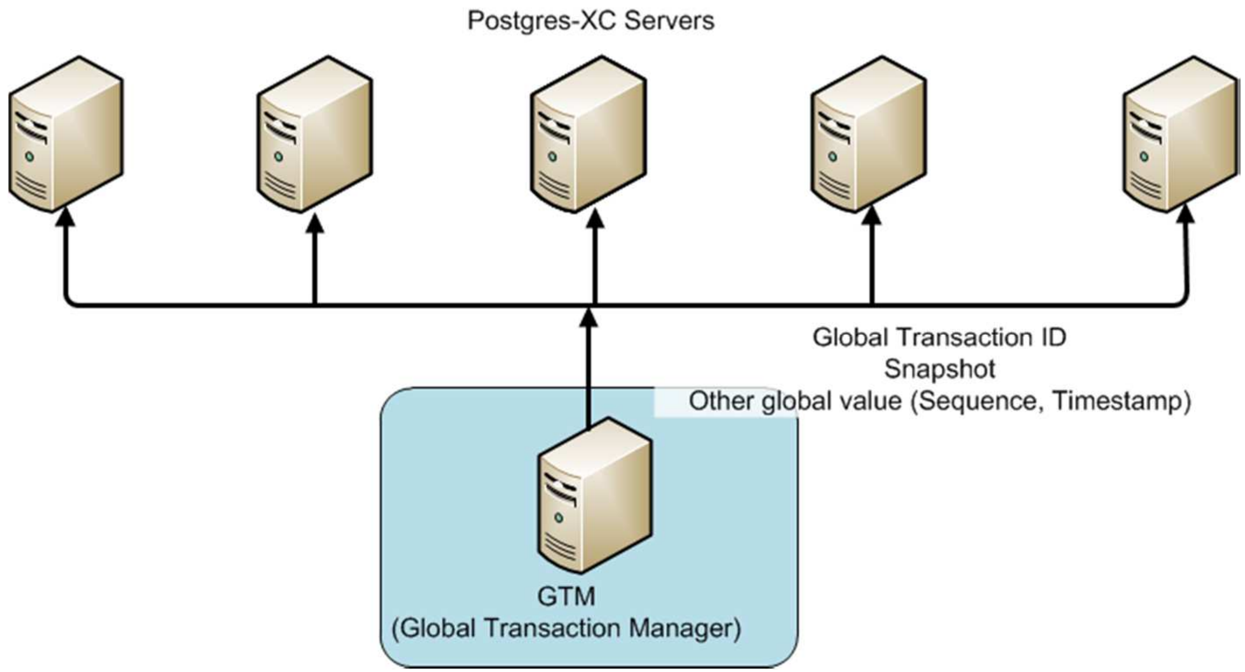
DBT-1 (Rev)

Current Status

- Now V 0.9.5 is available
- License changed to PostgreSQL license
  - Free to bring outcome back to PostgreSQL

GTM: Key for Transaction Transparency

- Consistent Transaction ID (GXID) throughout the system
- Provide global snapshot for consistent visibility from any server


Postgres-XC Servers
Global Transaction ID
Snapshot
Other global value (Sequence, Timestamp)
GTM
(Global Transaction Manager)

Requirements Since Last Year …

Solution for GTM as SPOF

- GTM Standby

Support same SQL statements as original PostgreSQL

- Functions
- Views
- Cross-node joins
- Role/User/Tablespace
- Transparent DDLs
- Many others

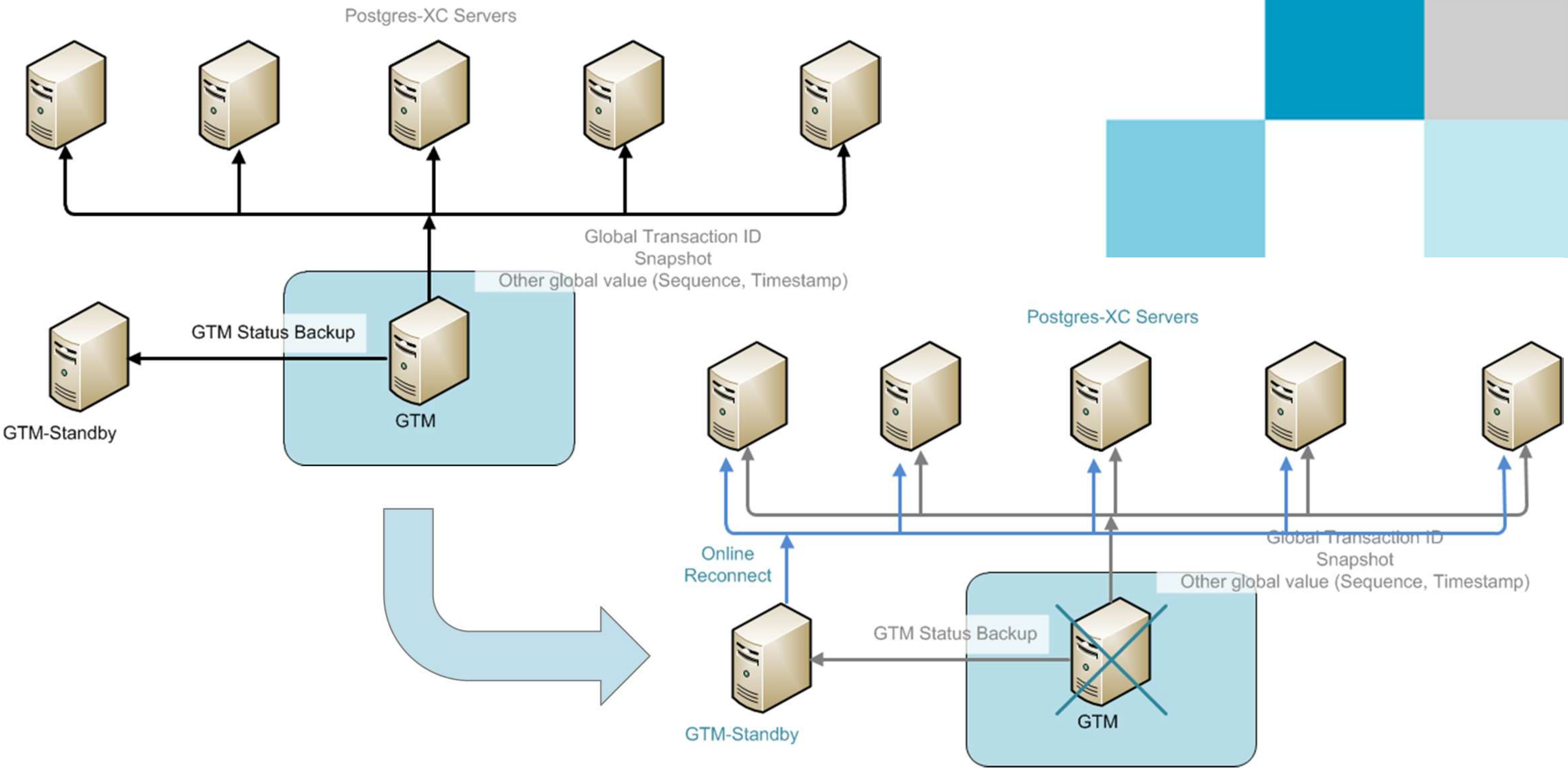
Other High Availability Feature such as

- Data Node Standby
- Consistent Backup and Recovery

Flexible Node Configuration

- On-line addition/Removal

GTM Standby


Postgres-XC Servers
Global Transaction ID
Snapshot
Other global value (Sequence, Timestamp)
GTM Status Backup
GTM-Standby
GTM
Postgres-XC Servers
Online
Reconnect
Global Transaction ID
Snapshot
Other global value (Sequence, Timestamp)
GTM Status Backup
GTM-Standby
GTM

GTM Standby Requirements

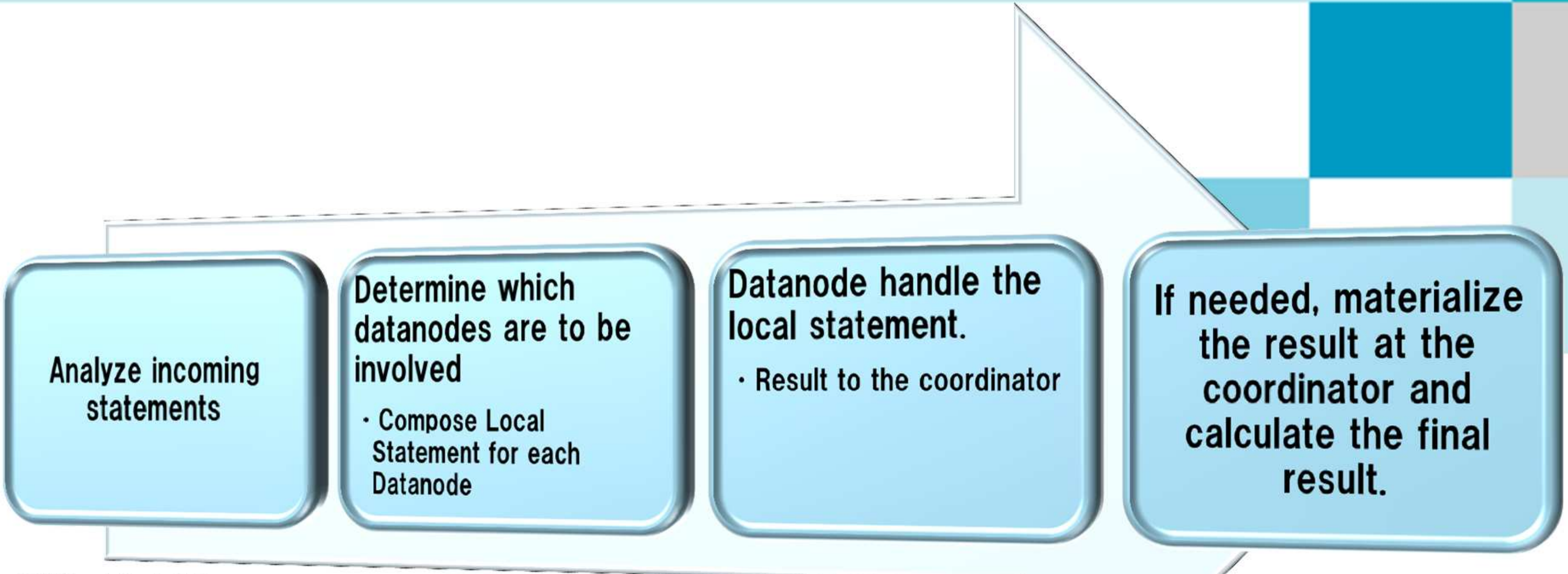
Online Promote and Reconnect

- Invisible from applications
  - Can be visible from GTM-Proxy
- Transactions should be able to continue to run

GTM-Standby: Current Status

- Infrastructure Available: V 0.9.5
- Improvement in progress
  - Connect to GTM at anytime
    - At present, GTM-Standby should be the first to connect to GTM
  - Get rid of any chance of backup information loss
    - Backup first
    - Negotiate the last message at reconnect
  - Performance
    - Backup grouping and decrease response
- Improvement scheduled at the next release

Postgres-XC Statement Extension



Analyze incoming statements

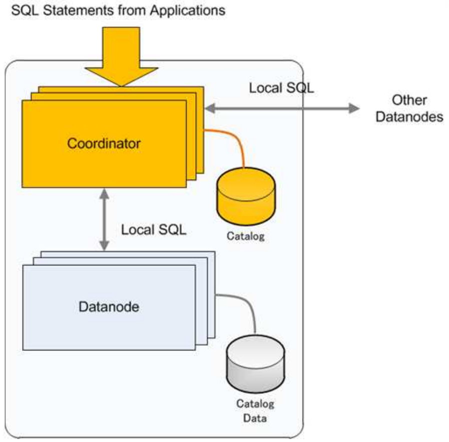
Determine which datanodes are to be involved

- Compose Local Statement for each Datanode

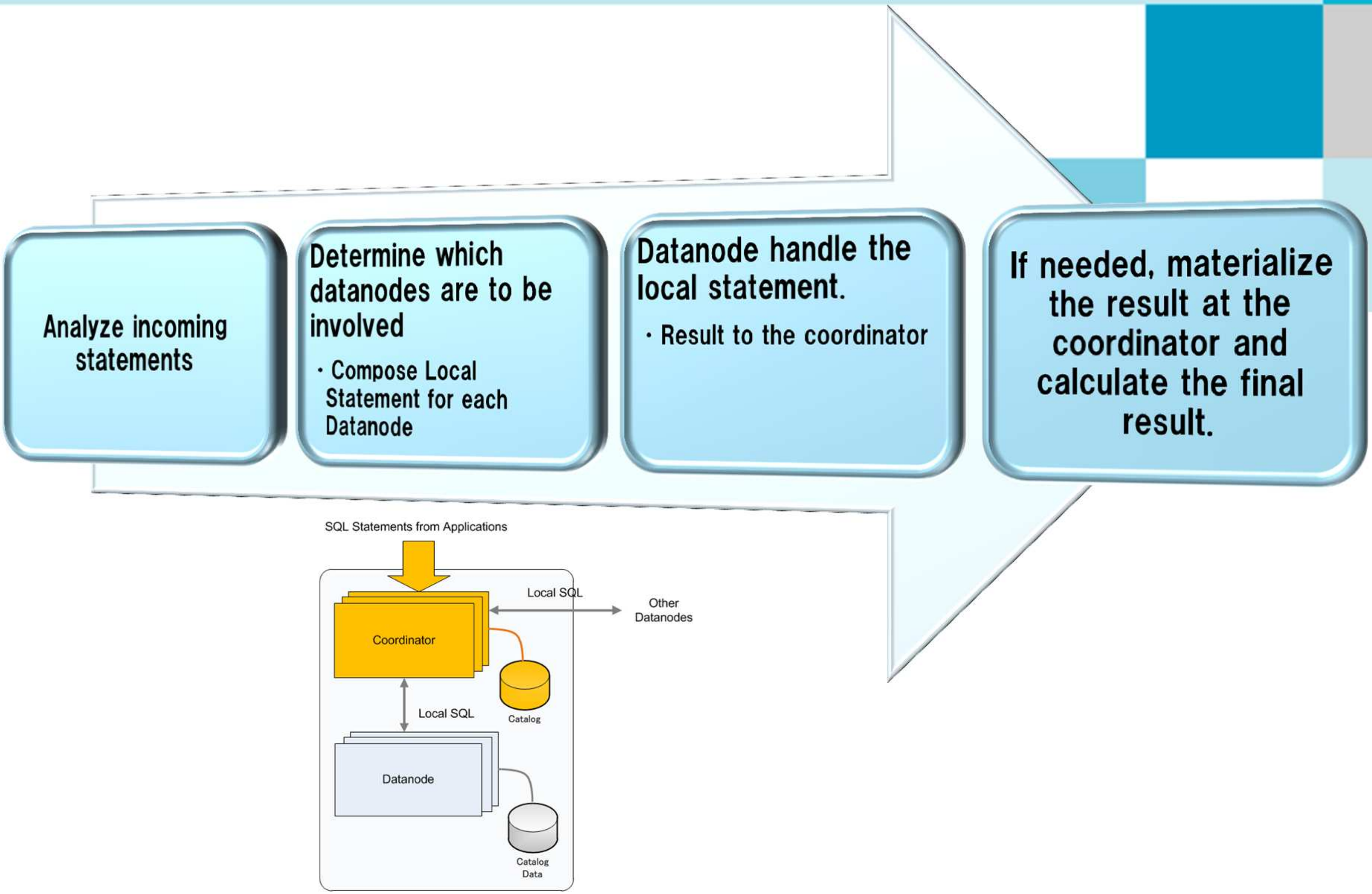
Datanode handle the local statement.

- Result to the coordinator

If needed, materialize the result at the coordinator and calculate the final result.



Postgres-XC Statement Extension



Analyze incoming statements

Determine which datanodes are to be involved

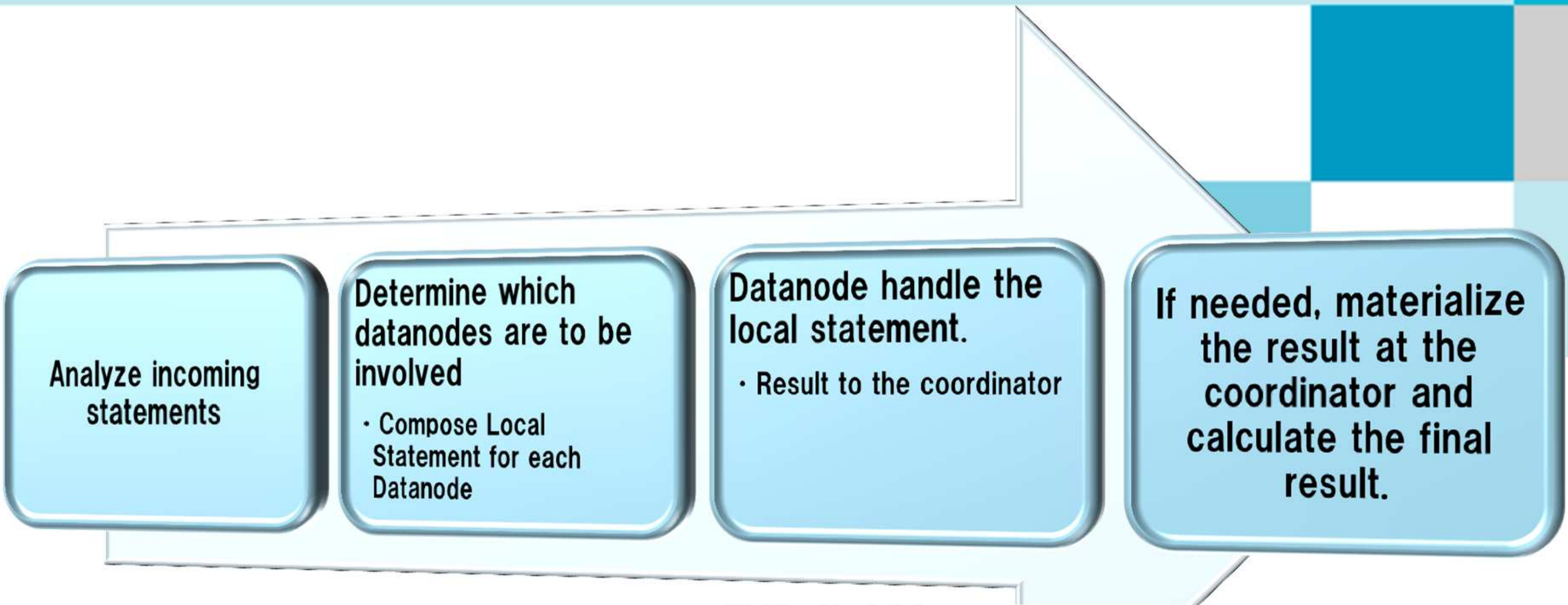
- Compose Local Statement for each Datanode

Datanode handle the local statement.

- Result to the coordinator

If needed, materialize the result at the coordinator and calculate the final result.

Postgres-XC Statement Extension



Analyze incoming statements

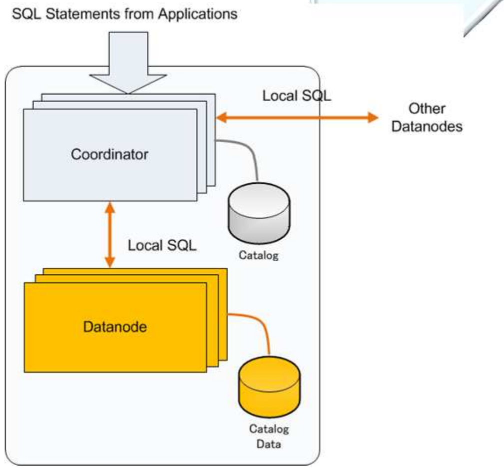
Determine which datanodes are to be involved

- Compose Local Statement for each Datanode

Datanode handle the local statement.

- Result to the coordinator

If needed, materialize the result at the coordinator and calculate the final result.



Postgres-XC Statement Extension

Analyze incoming statements

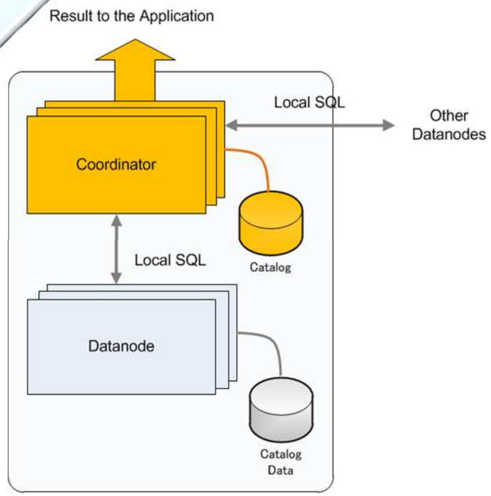
Determine which datanodes are to be involved

- Compose Local Statement for each Datanode

Datanode handle the local statement.

- Result to the coordinator

If needed, materialize the result at the coordinator and calculate the final result.



Optimizing Statements（V 0.9.5）

Push-down as many clause as possAZible

・Join

・WHERE Clause

・Aggregate

・Functions（when used in WHERE clause)

・Column projection

Uses the following information

・If each table is replicated or partitioned

・Partition key

・Partition algorism (Hash/Modulo/Round Robin)

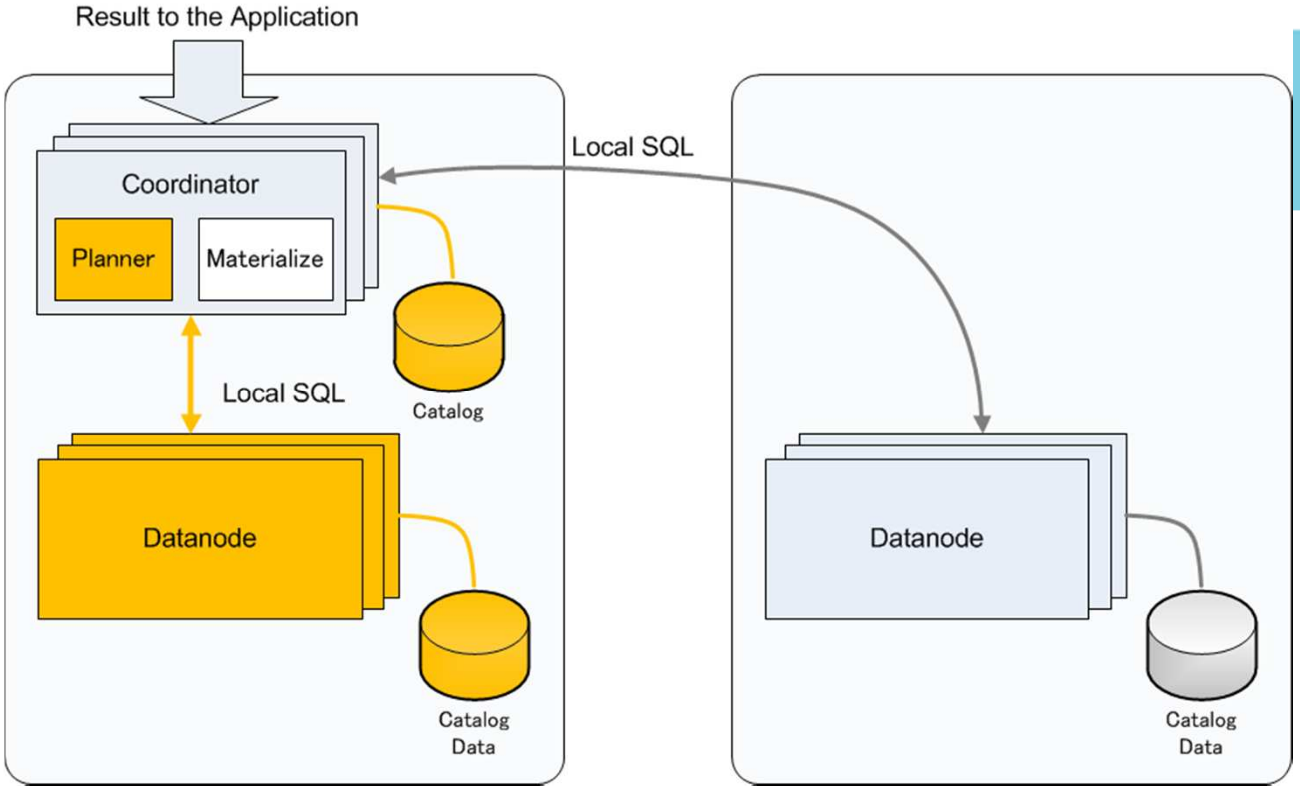
Future Improvement

Candidate

- Use statistic info.
- Use Semi-Join to determine joining rows
- Direct join tuple transfer among datanodes
- Much more …

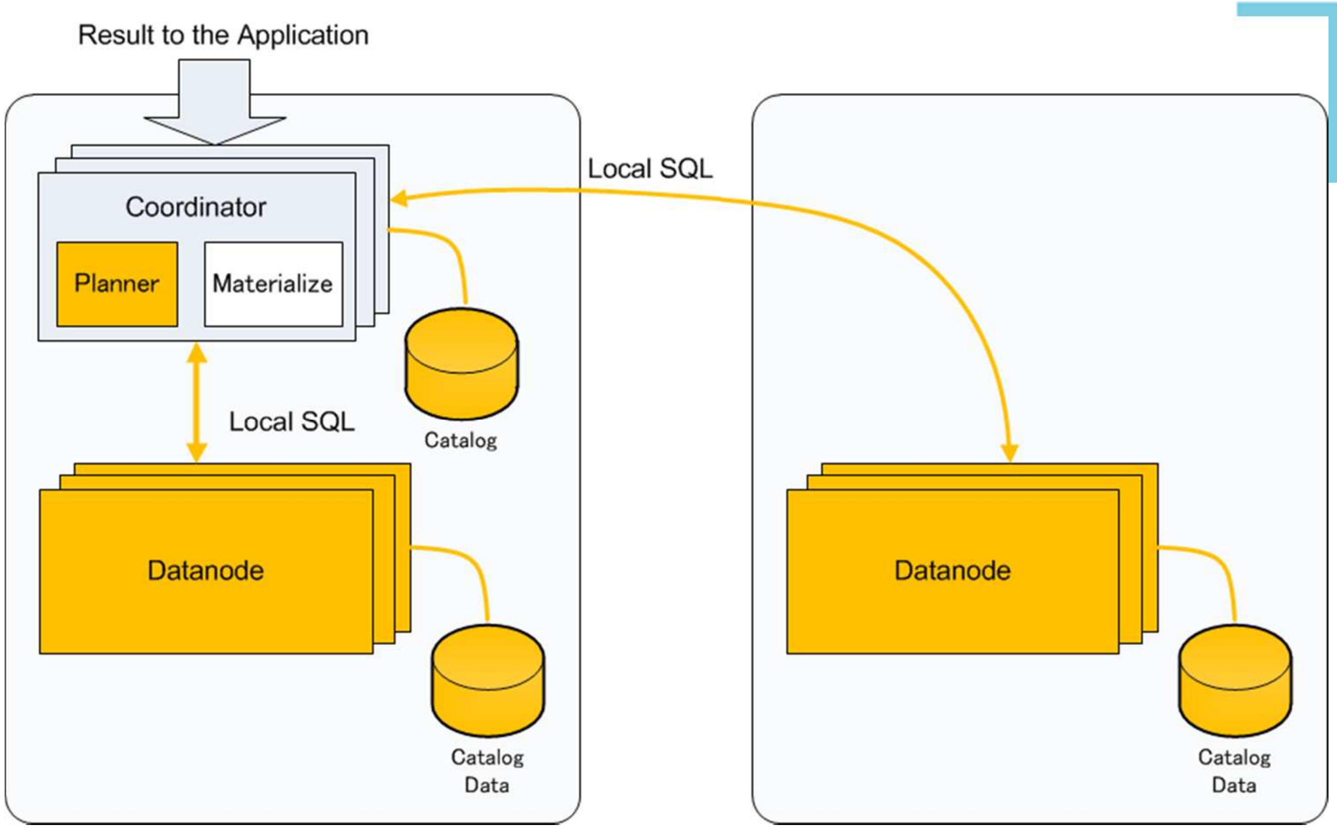
XC Optimization Examples（Join-1)

- Both Tables Are Replicated



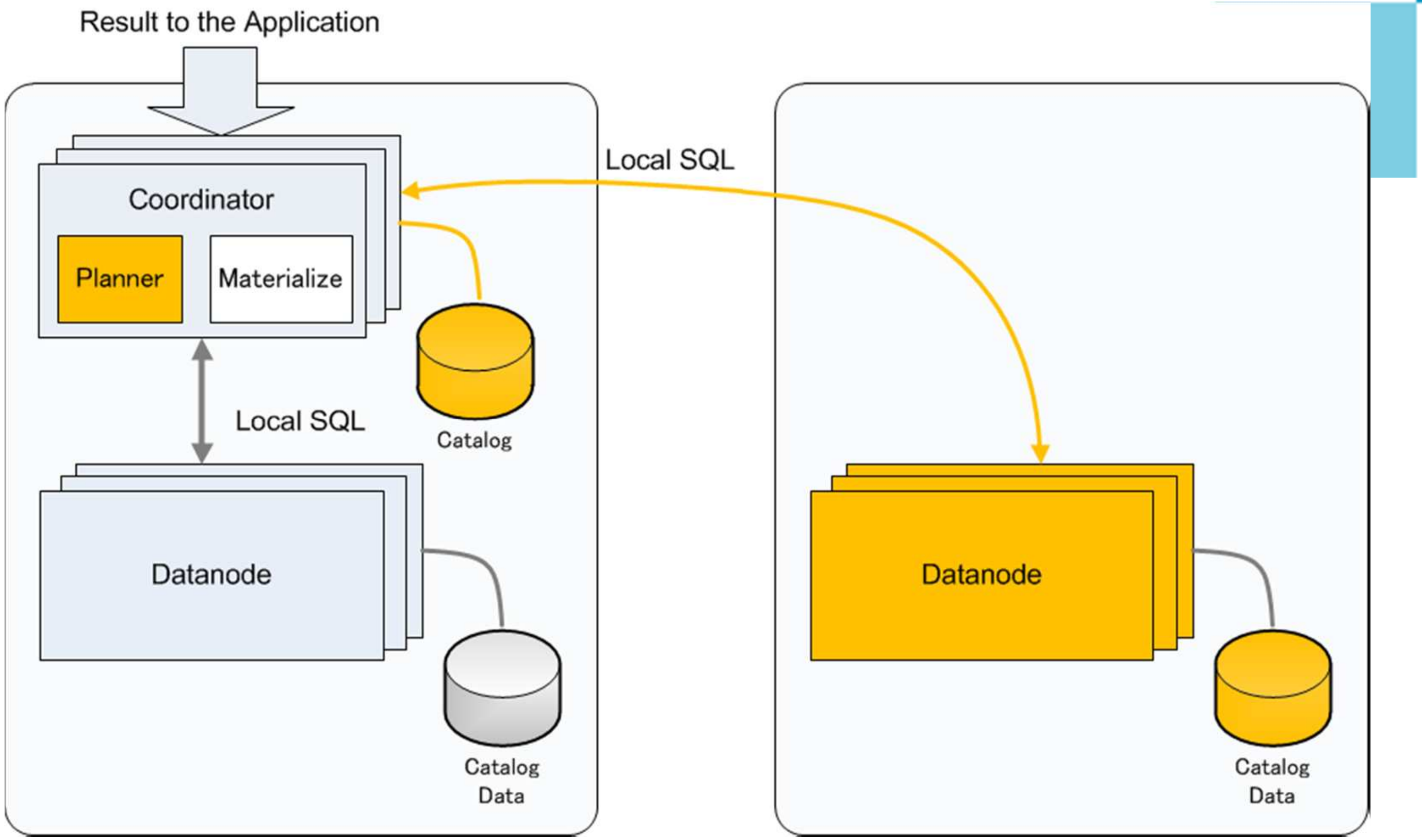
XC Optimization Examples (Join-2)

- Replicated Table and Partitioned Table
  - Cannot determine which datanode to go



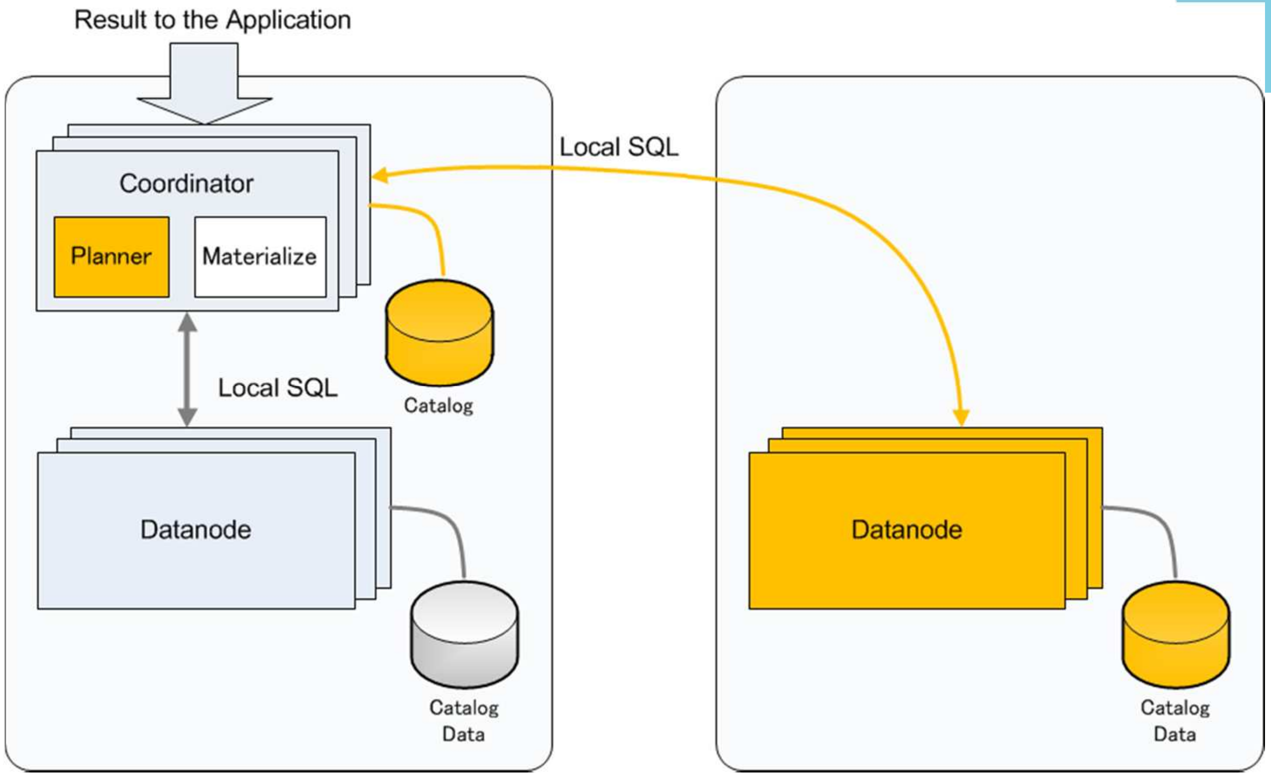
XC Optimization Examples（Join-3)

- Replicated Table and Partitioned Table
  - Can determine which datanode to go from WHERE clause



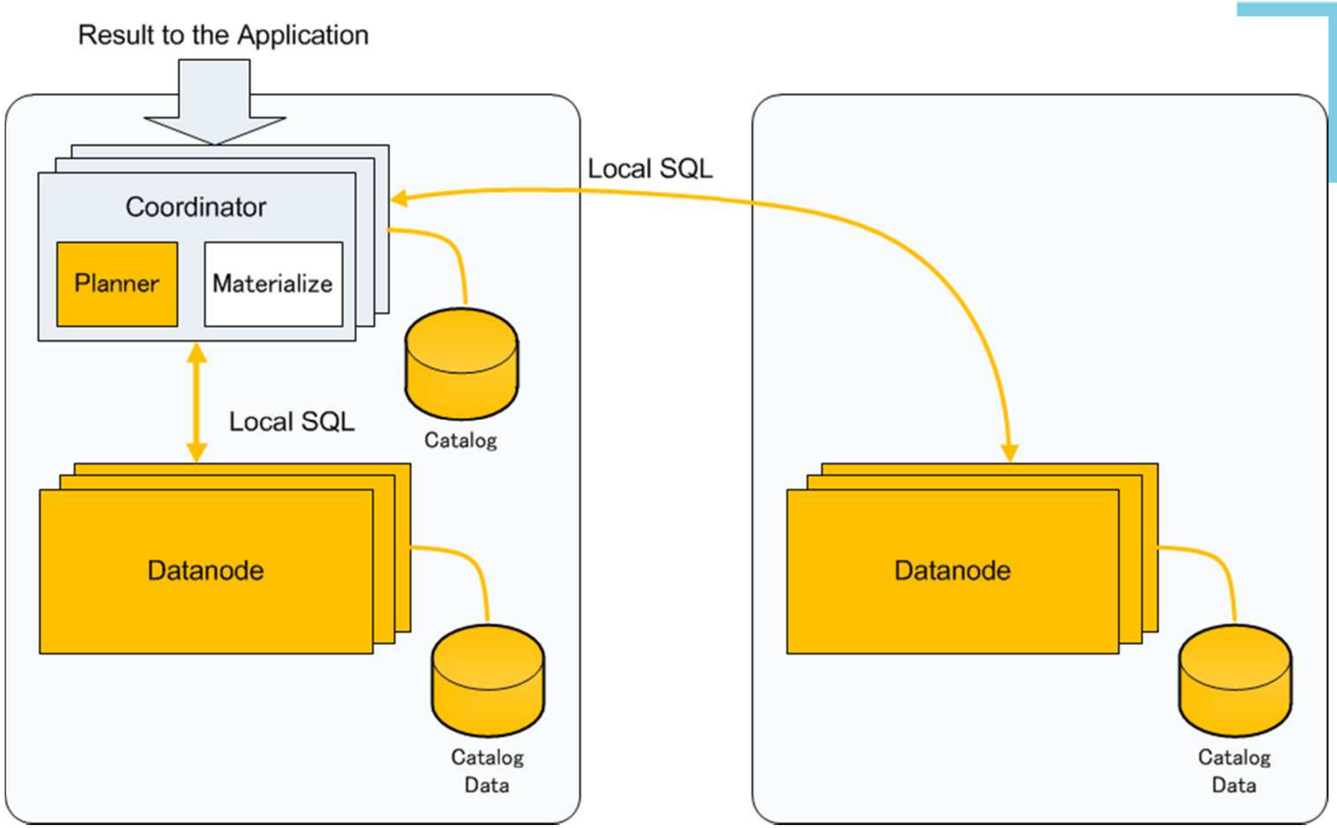
XC Optimization Examples（Join-4)

- Partitioned Table and Partitioned Table
  - Both Join columns are distribution (partitioning) column
  - Where clause can determine which datanode to go



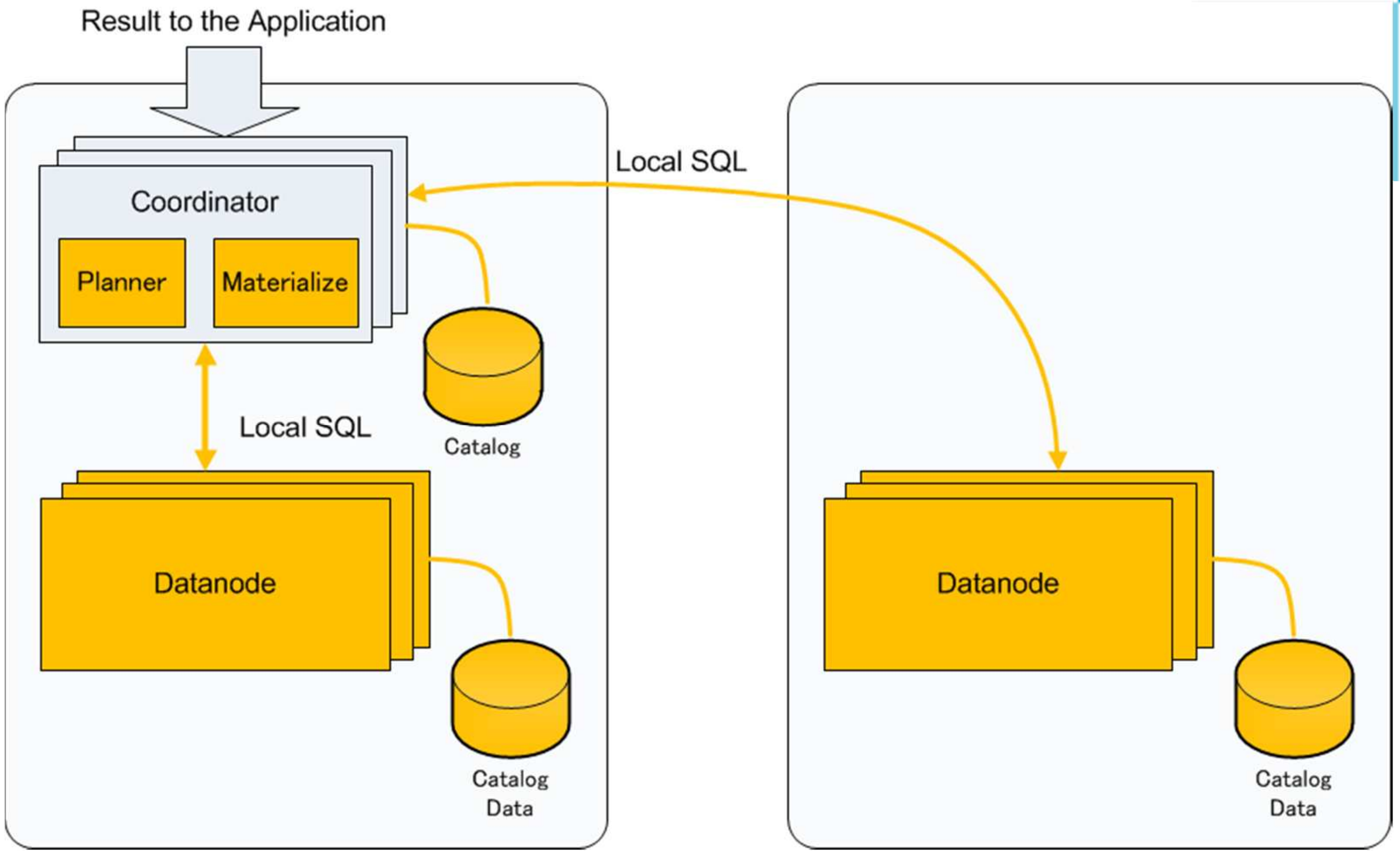
XC Optimization Examples（Join-5)

- Partitioned Table and Partitioned Table
  - Both Join columns are distribution (partitioning) column



XC Optimization Examples（Join-6)

- Partitioned Table and Partitioned Table
  - One of Join columns are not distribution (partitioning) column



XC Statement Handling Summary

- Now can handle wide variety of PostgreSQL statement.
- Still in progress
  - HAVING
  - PREPARE, EXECUTE, CURSOR
    - Eliminate restrictions
  - WITH/WITH RECURSIVE
  - General Subqueries
  - Functions with more than one statement
  - SELECT INTO (CREATE TABLE AS)
  - Triggers
  - Temp tables
- Challenges
  - Global constraint
  - More Optimization
  - More Parallelism
- Miscellaneous
  - LISTEN/NOTIFY/UNLISTEN

Multi-Statement Planner

Backup and Recovery (PITR) Requirement

- Transaction status should be consistent
  - Each transaction must be either:
    - Committed in all the involved node
    - Running or aborted in all the involved node
- Write such timing in WALs of all the coordinators and datanodes.
- Application can provide such timing as “BARRIER”
  - CREATE BARRIER *barrier_id*
    - Wait partially-committed-transactions completes commit,
    - Block other transaction’s commit,
    - Write BARRIER record to WALs of all the coordinators/datanodes.
  - When running PITR, specify barrier_id in recovery.conf

Demonstration

Further Development Topics/Schedule (1)

- Support more variety of statements:
  - HAVING, PREPARE, EXECUTE, CURSOR, TRIGGER
    - By the end of this year
  - SAVEPOINT
  - Multi-statement planner for WITH, WITH RECURSIVE, general functions, general subqueries, SELECT INTO, CREATE TABLE AS
    - By the end of this year

Further Development Topics/Schedule (2)

- Datanode high-availability
  - Backup with synchronous streaming replication
    - Synchronous replication needed to maintain data integrity among datanodes.
- Cluster operation
  - Online server addition/removal
- Challenging
  - Global constraint
    - Unique/Reference integrity among partition,
    - Exclusion constraint among partition
  - LOB
- Others needs additional test
  - dblink
  - SQL/MED

Postgres-XC to PostgreSQL

- Snapshot cloning
  - Parallel pg_dump
  - Parallel query execution (local/cluster)
- SQL/MED extension
  - Column projection pushdown
  - Join pushdown
  - Function pushdown
- Federation
  - Materialization
  - Cross-node join
  - Cross-node aggregation

Many candidate features.
Need more members for quick actions.

New Developer Wanted

- Writing Code
  - New distributed/parallel query handling/optimization
  - HA capabilities
  - Utilities
    - Installation
    - Configuration
    - Operation
  - Bug fixes
  - Back port to PostgreSQL
- Build
  - Creating binaries/distribution packages
- Test
  - Performance evaluation with various benchmarks
  - Finding bugs
  - New feature proposals
- Pilot application
  - Practical applications

Project resources

- Development site
  - http://sourceforge.net/projects/postgres-xc/
  - http://sourceforge.net/apps/mediawiki/postgres-xc/
- Project home
  - http://postgres-xc.sourceforge.net/
- Mailing List
  - http://postgres-xc.sourceforge.net/mailinglist.html

Contact us

Thank you very much!

Koichi Suzuki

NTT DATA INTELLILINK Corporation

Pacific Marks Tsukishima,1-15-7,
Tsukishima, Chuo-ku,
Tokyo 104-0052, Japan

TEL : +81 3 5843 6800
E-mail : koichi@intellilink.co.jp
koichi.szk@gmail.com
URL : http://www.intellilink.co.jp/ *only in Japanese

Global IT Innovator

NTT DATA GROUP

NTT DATA