PGConf.ASIA 2017

FUJITSU

shaping tomorrow with you

Road to a Multi-model Database -- making PostgreSQL the most popular and versatile database

December 5, 2017

Takayuki Tsunakawa
Fujitsu Limited

Who am I?

FUJITSU

Takayuki Tsunakawa

- PostgreSQL contributor
- PostgreSQL Enterprise Consortium member
  (PostgreSQL Ecosystem Wiki maintainer)

- Develop/Maintain/Support
  FUJITSU Software Enterprise Postgres
  (PostgreSQL-based product)
- Support open source PostgreSQL in various products

Agenda

FUJITSU

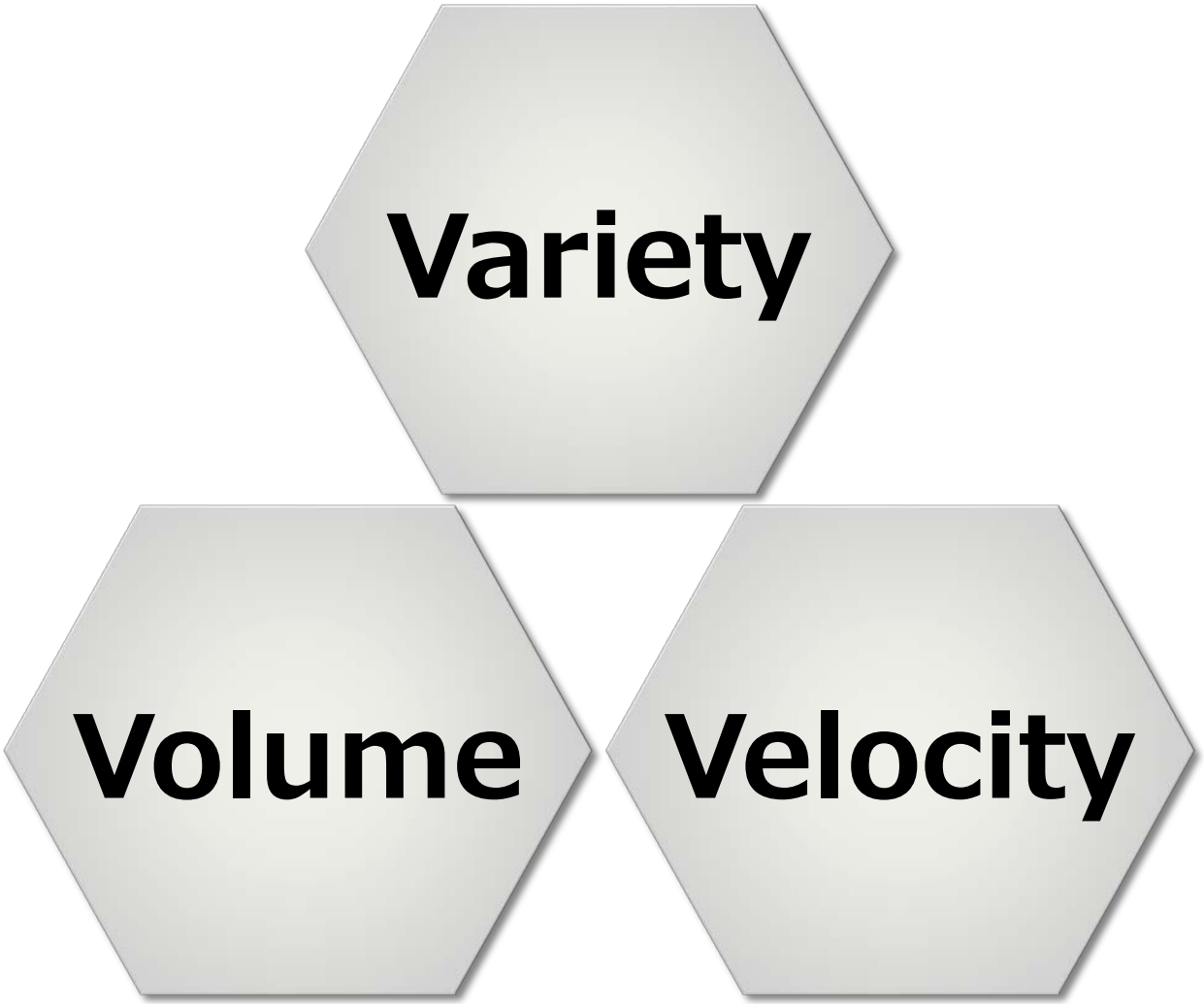
- Why is multi-model necessary? (background)
- What is multi-model database?
- How should we implement it?

FUJITSU

Why is multi-model necessary?

Big Data

FUJITSU



Can PostgreSQL Handle Big Data?

FUJITSU

Variety

Key-value model
hstore type

Document model
jsonb type

Volume

Partitioning
PostgreSQL 10~

Scaleout

Postgres-XL
(fork)

Citus
(extension)

Velocity

In-memory columnar
In developing

GPU
PG-Strom
(extension)

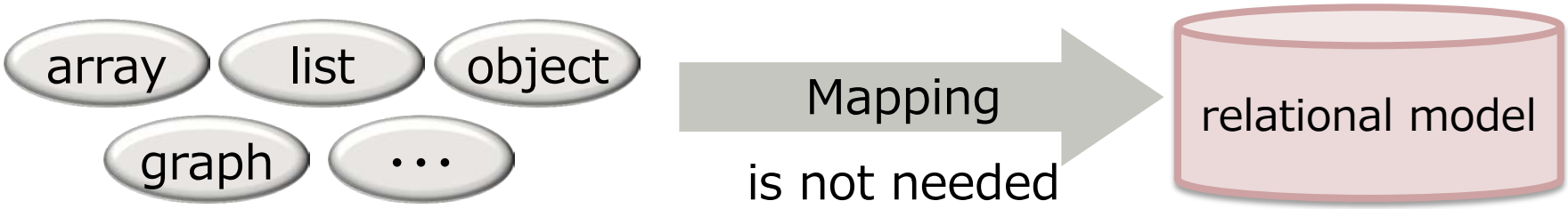
Streaming
PipelineDB
(fork)

Persistent memory, FPGA, SIMD
N/A

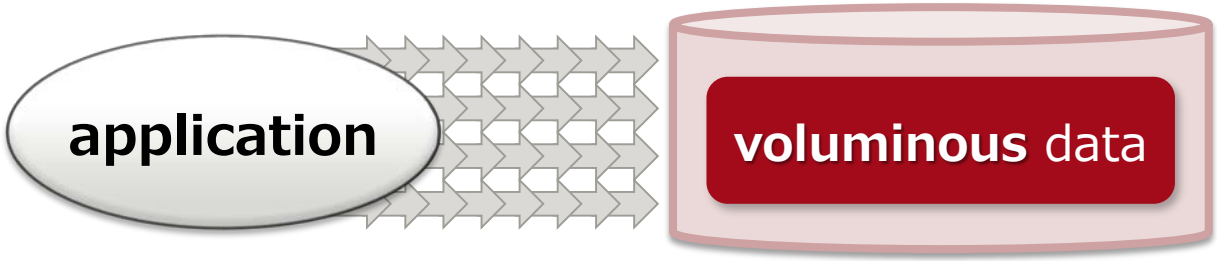
Why NoSQL Attracts Attention?

FUJITSU

- Developer productivity with flexible data model
  - Can handle various data types as-is (array, list, object, graph, etc.)
  - No need to map to relational model (eliminate ORM)



- High scalability
  - Can store and process voluminous data
  - Can handle many requests simultaneously



- Fault tolerance

Data Models

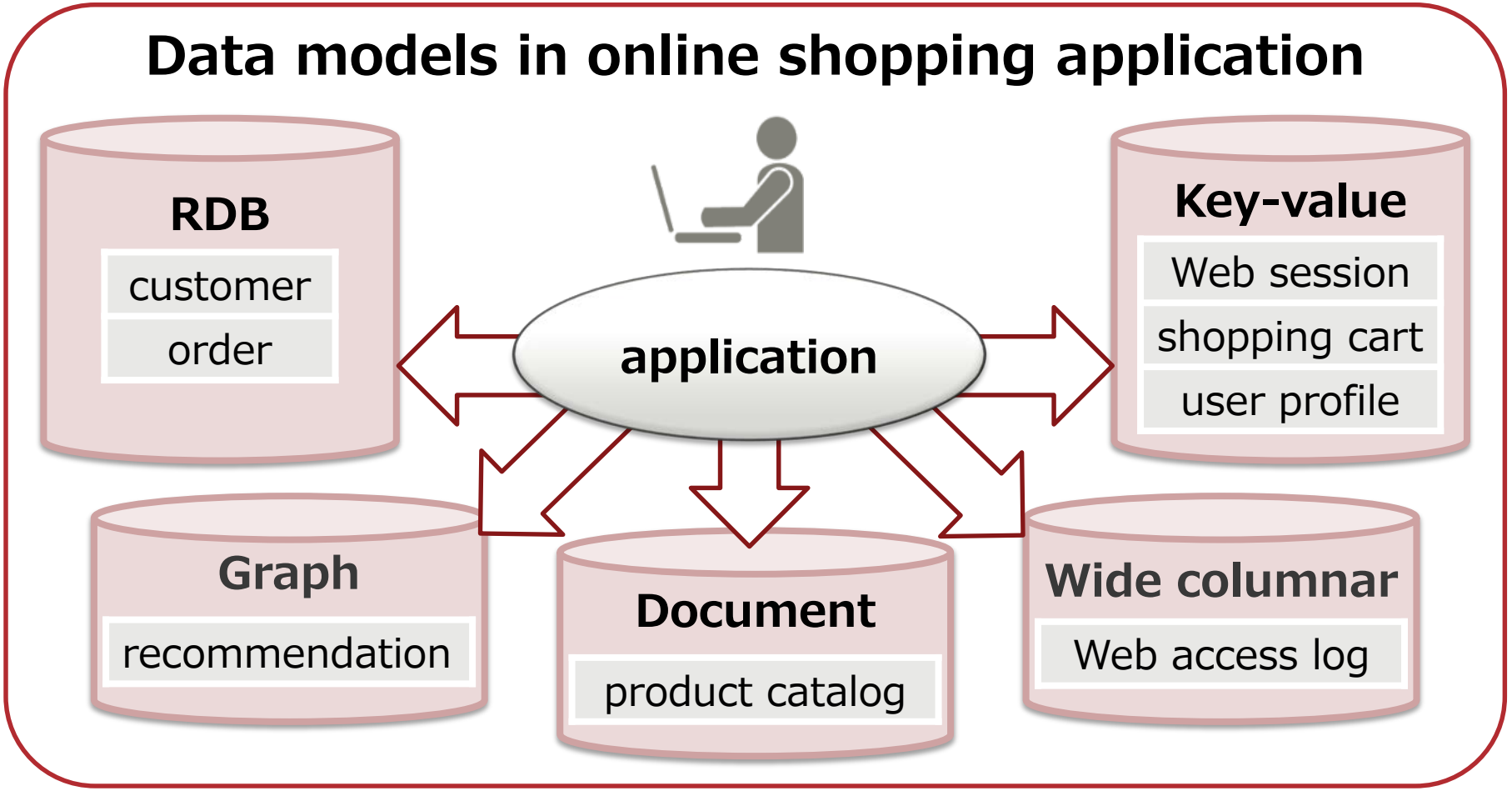
FUJITSU

Data model	Representative DBMSs
Relational	Oracle, MySQL, SQL Server, PostgreSQL
Key-value	Redis, Memcached
Document	MongoDB, CouchBase, MarkLogic
Graph	Neo4j
Wide columnar	Cassandra, Hbase
RDF	MarkLogic, Virtuoso, Oracle
Text search	Elasticsearch, Apache Solr
Time series	InfluxDB
Multi-dimensional array	rasdaman, SciDB
Event	Event Store, NEventStore
Object	InterSystems Cache

Polyglot Persistence

FUJITSU

- Use multiple DBMSs in one system/application
- Spread by Martin Fowler



Multiple DBMSs Use

FUJITSU

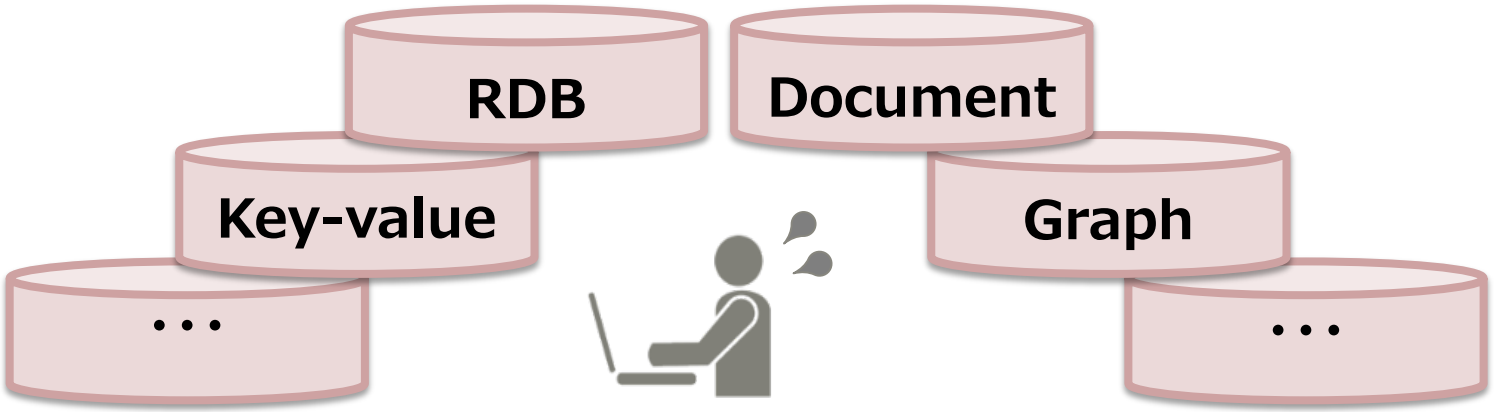
- Leading tech companies use many DBMSs (ex. Netflix)

Data model	DBMSs
Relational	MySQL, Redshift
Key-value	Memcached, Redis, Hollow (developed by Netflix)
Text search	Elasticsearch
Wide columnar	Cassandra
Time series	Atlas (developed by Netflix)
Event	Druid

Problems (1/2)

FUJITSU

- Data silo to prevent cross-sectional data analysis
  - Time-consuming and laborious ETL
  - Complex logic in application (fetch, join, aggregation, sort)
- Data consistency among DBMSs
  - Distributed transaction is not available in all DBMSs
- Infrastructure cost increase due to duplication of data



Problems (2/2)

FUJITSU

- Operational complexity
  - Product/OSS software management, support/service contracts
  - Infrastructure provisioning (server, storage, network)
  - Deployment, patching, testing, configuration, version control
  - Security: user management, access control, encryption, auditing
  - Monitoring and diagnosis, performance tuning, troubleshooting
  - HA: backup/recovery, local failover, disaster recovery

- Steep learning curve for developers
  - DBMS-specific non-SQL API and SQL-like query language
  - Transaction control, consistency model, application tuning

- Lack of skilled personnel

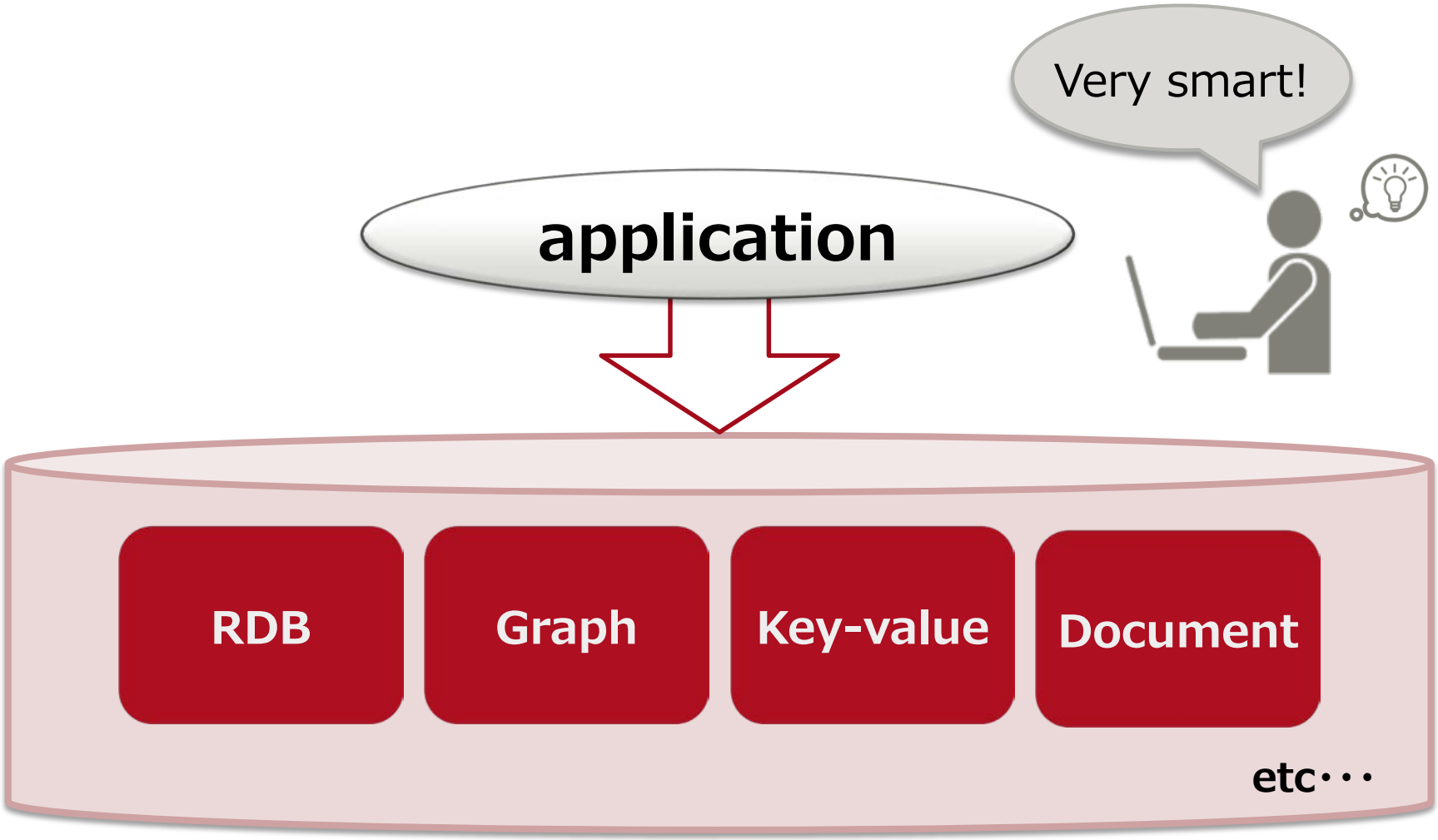
FUJITSU

What is multi-model database?

Overview

FUJITSU

- Support multiple data models in one DBMS



Merits

FUJITSU

"All-in-one" is convenient,
just like a smartphone

Smooth data utilization
with less data integration

Higher developer productivity

Lower cost
for infrastructure and DBA

Multi-model Database Examples

FUJITSU

DBMS	Supported data models
ArangoDB	key-value, document, graph
Cosmos DB	key-value, document, graph
CouchBase	key-value, document
DataStax(on Cassandra)	key-value, wide column, graph
MarkLogic	document, text/binary, RDF
OrientDB	key-value, document, graph, text/binary

Trends of Major DBMSs

FUJITSU

- Major RDBMSs are adding data models
- NoSQL DBMSs are also adding data models

Data model support in top 5 popular DBMSs

DBMS	Key-value	Document	Wide column	Graph
Oracle		++		+
MySQL	++	+		
SQL Server		+		+
MongoDB	+	++		+
PostgreSQL	+	+		

PostgreSQL as a Multi-model Database

FUJITSU

- Why based on RDBMS?

RDBMS has

- ✓ Mature storage engine and transaction management
- ✓ Smart optimizer
- ✓ Prevalent RDBMS gives more people the chance to use

- Why based on PostgreSQL?

PostgreSQL has

- ✓ Extensibility as a data platform
- ✓ Liberal community open to niche data models

FUJITSU

How should we implement multi-model database?

What is Data Model?

FUJITSU

Data model = Structure + Constraint + Operation

Data model

	Structure	Constraint	Operation
Relational	table, row, column	unique, referential, check, not null, …	scan, join, restriction, projection, …
Key-value	key, value	unique	get, put
Graph	node, relationship, property, label	unique, node existence	scan, join, restriction, projection, pattern match, …

Query Language and API

FUJITSU

- Adopt standard and well-known languages/APIs per data model
  - Developer productivity: leverage skill/know-how/asset
  - Rich information for learning
  - Standard compliance and popularity for ecosystem

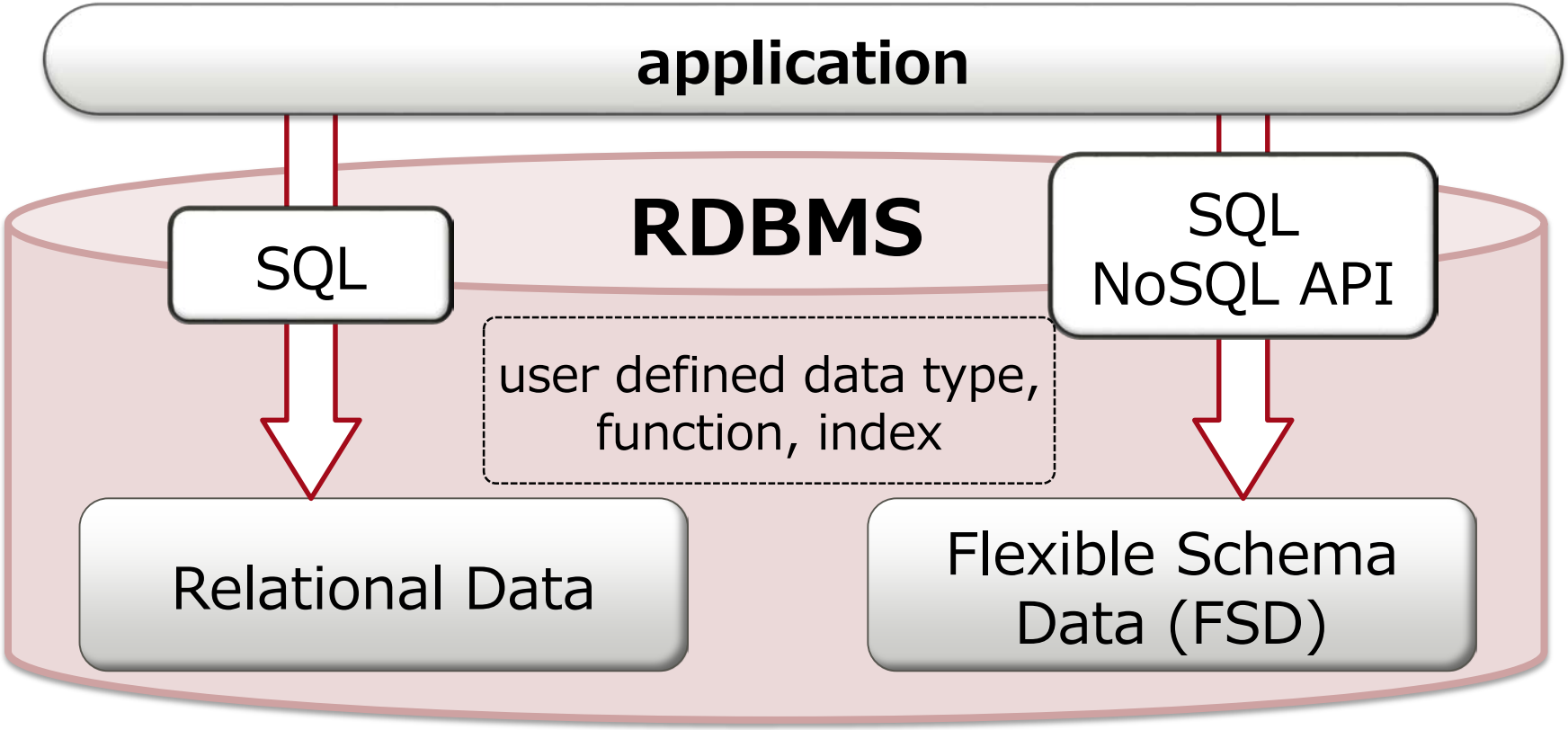
- Examples

Data model	languages/APIs
Key-value	Redis API, Memcached API
Document	SQL/JSON path (SQL standard), MongoDB API
Graph	Cypher, Gremlin
RDF	SPARQL (W3C standard)
Array	SQL/MDA (Multi-Dimensional Array) (future SQL standard)

Multi-model Approach 1

FUJITSU

- Flexible Schema Data (FSD)
  - Leverage RDBMS's user defined data type, function, and index
  - Store/access data in a table column with functions in SQL
  - Used for XML, JSON, geospatial data

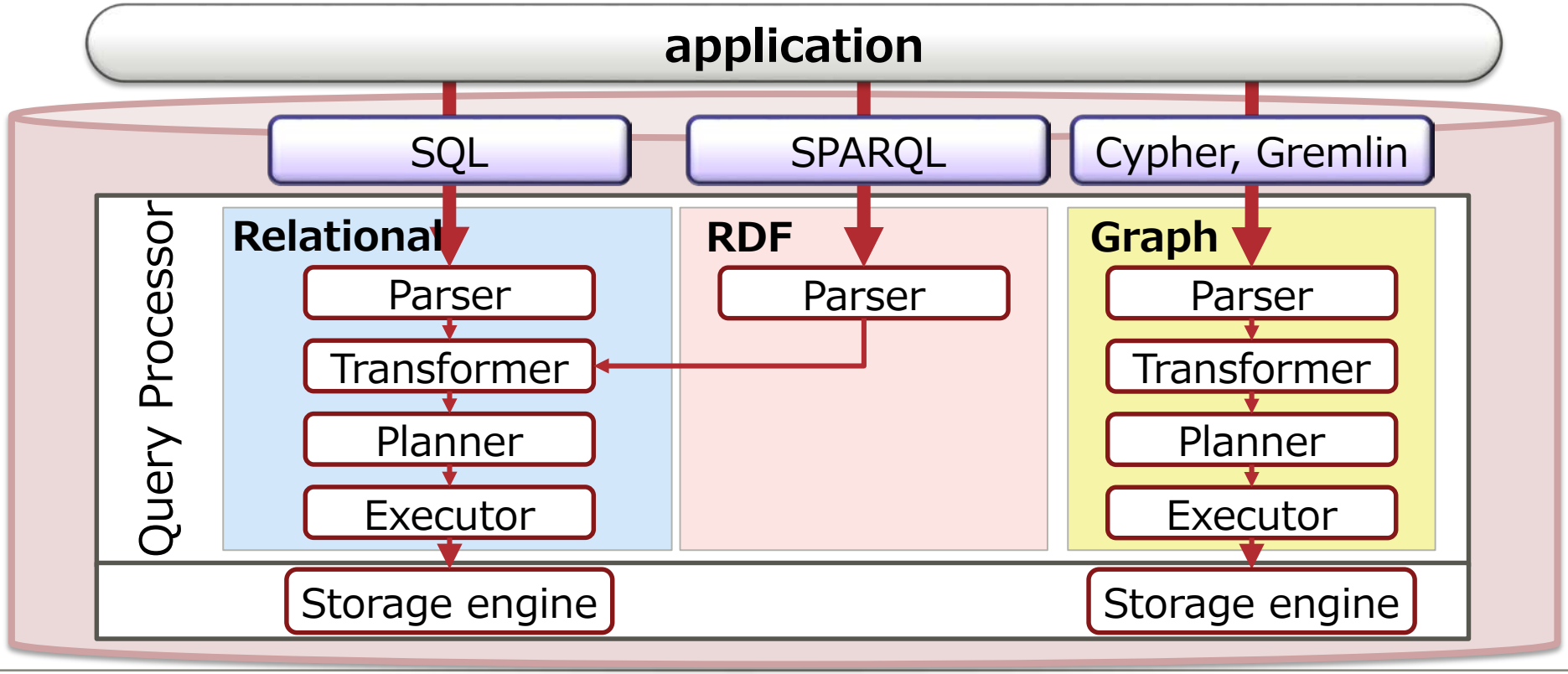


reference : http://cidrdb.org/cidr2015/Papers/CIDR15_Paper5.pdf

Multi-model Approach 2

FUJITSU

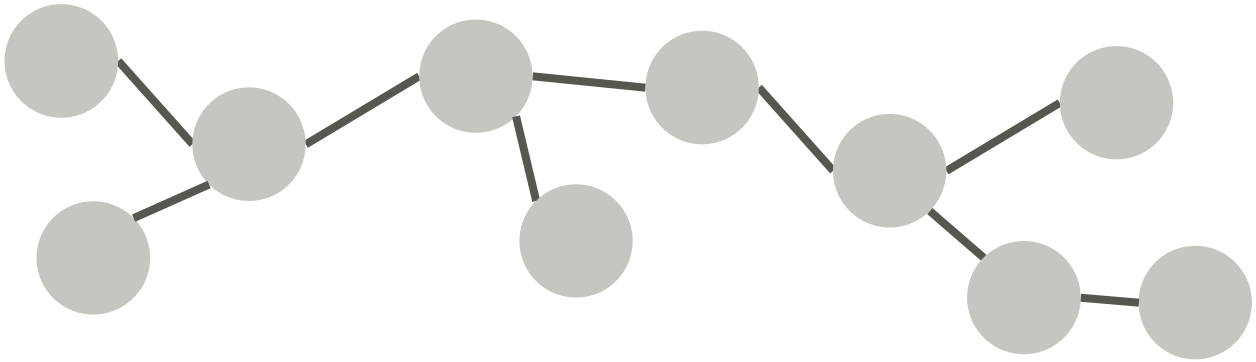
- Independent data model components
  - Query language and API for each data model
  - Data is optionally separated from relational data
  - Use for Graph, RDF, time series, event…
- Independence ensures performance for each data model



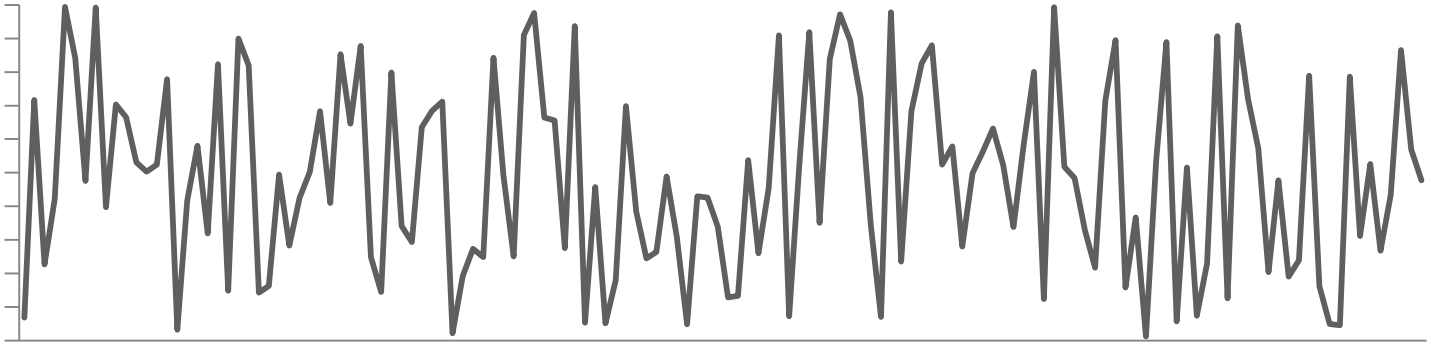
Examples Based on Approach 2

FUJITSU

- Graph model: AgensGraph (fork)
  - https://github.com/bitnine-oss/agensgraph



- Time series model: TimescaleDB (extension)
  - https://github.com/timescale/timescaledb

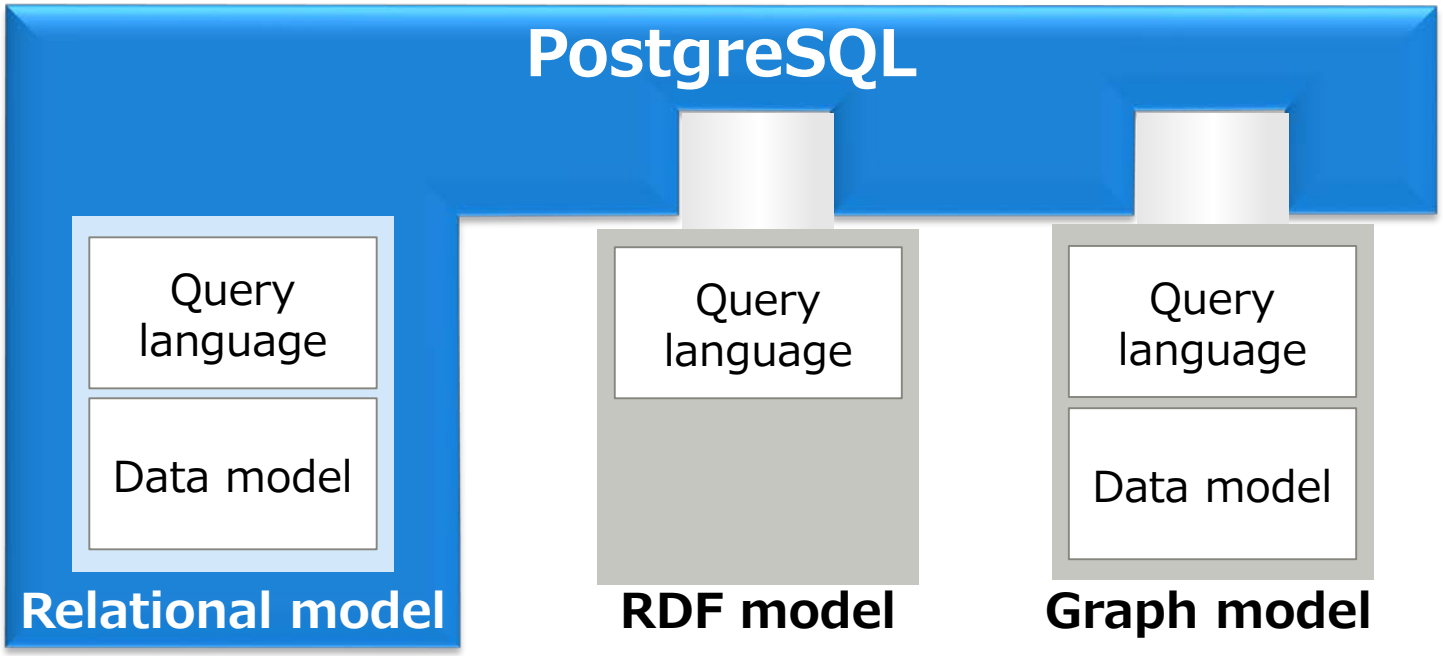


Pluggable Data Model

FUJITSU

- Want to facilitate data model development
- Introduce 3 pluggable objects
  - Query language : generate parse tree from query string
  - Data model : generate query plan from parse tree and run it
  - Region : combination of query language and data model

Data model as an extension



Multi-model Query

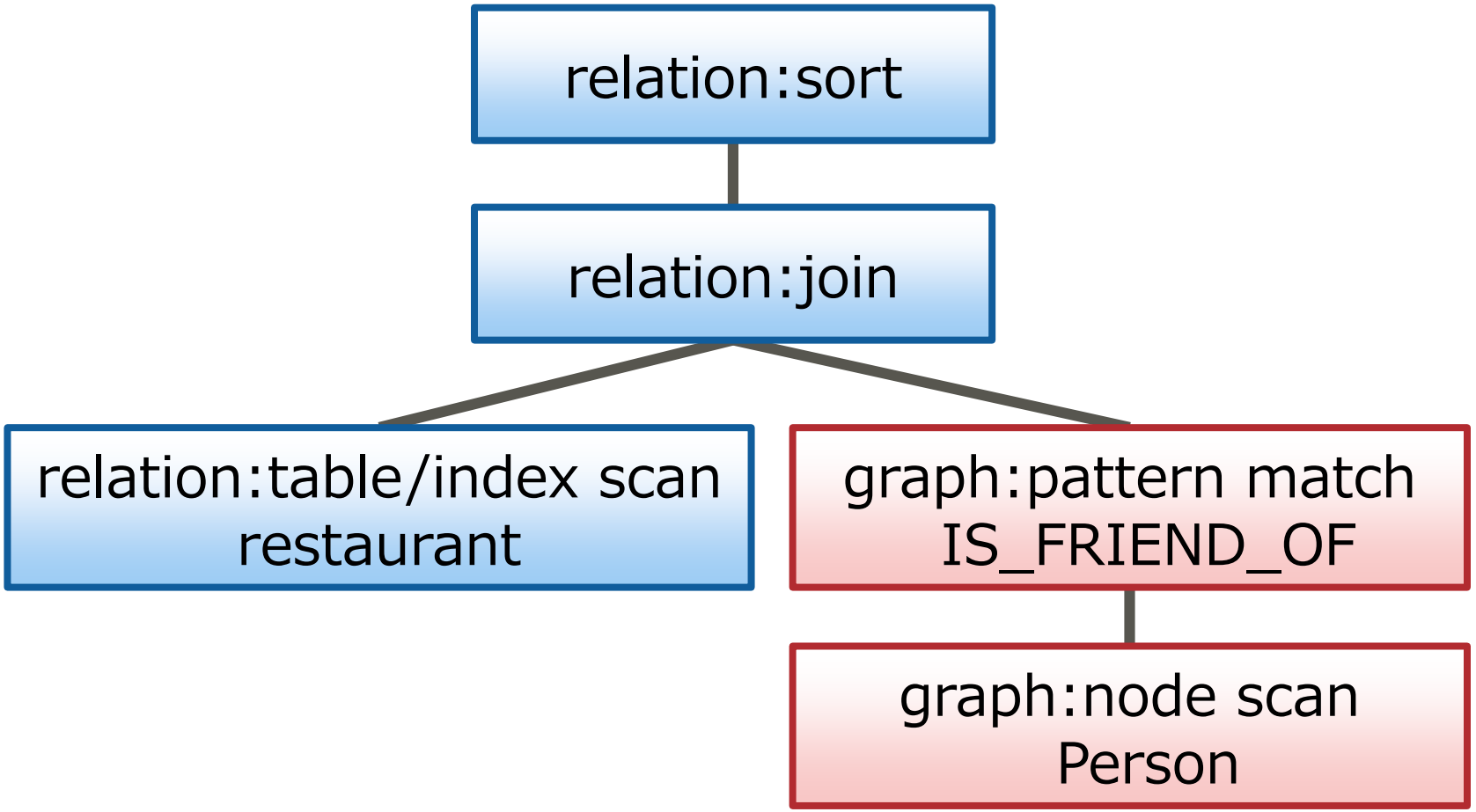
FUJITSU

- Mix queries for multiple data models in a query string
  - Execute query in a specified region
    in_region(region_name, query string)
  - Convert data across regions
    cast_region(source data, dest region name,
    dest container, dest schema)

```
-- Among Chinese restaurants in Tokyo,
-- list up to 5 top ones among friends' friends
SELECT r.name, g.num_likers FROM restaurant r,
  cast_region(
    in_region('graph_cypher',
      'MATCH (:Person {name:"Taro"})-[:IS_FRIEND_OF*1..2]-(friend),
      (friend)-[:LIKES]->(restaurant:Restaurant)
      RETURN restaurant.name, count(*)'),
  'relational', 'g', '(name text, num_likers int')
WHERE r.name = g.name AND r.city = 'Tokyo' AND r.cuisine = ‘chinese'
ORDER BY g.num_likers DESC LIMIT 5;
```

FUJITSU

Multi-model query plan



Document Model

FUJITSU

- PostgreSQL supports JSON since 2012, but…
- Different SQL/JSON was standardized in SQL:2016
  - Store JSON data in character/binary column
  - Intuitive function and SQL/JSON path language
  - Powerful JSON_TABLE function to map JSON to relational data
- Support for SQL/JSON is being developed in community

Query in current PostgreSQL

```
SELECT
  JSON_VALUE(jcol, '$.name') AS name,
  JSON_QUERY(jcol, '$.skills') AS skills
FROM emp
WHERE
  JSON_EXISTS(jcol, '$.projects[*] ?
    (@.category == "IoT")');
```

Query in SQL/JSON

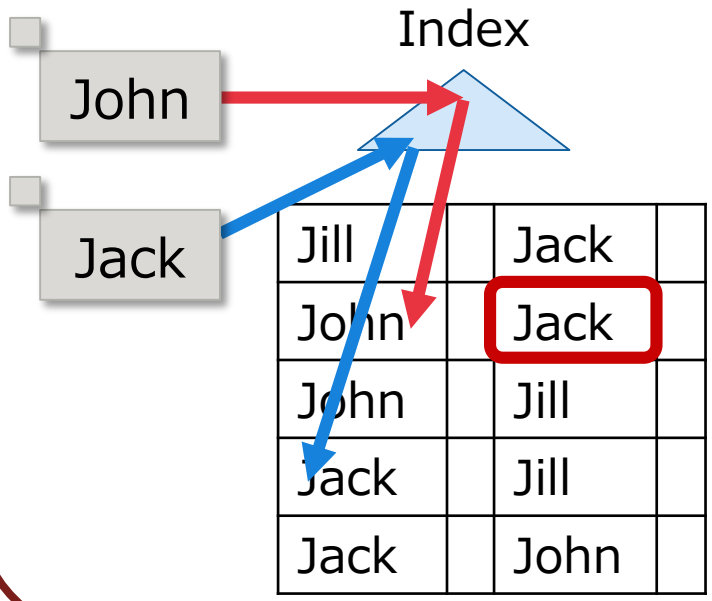
```
SELECT
  jcol ->> 'name' AS name,
  jcol -> 'skills' AS skills
FROM emp
WHERE
  jcol @>
    '{ "projects": [{ "category": "IoT" }] }';
```

Graph Model

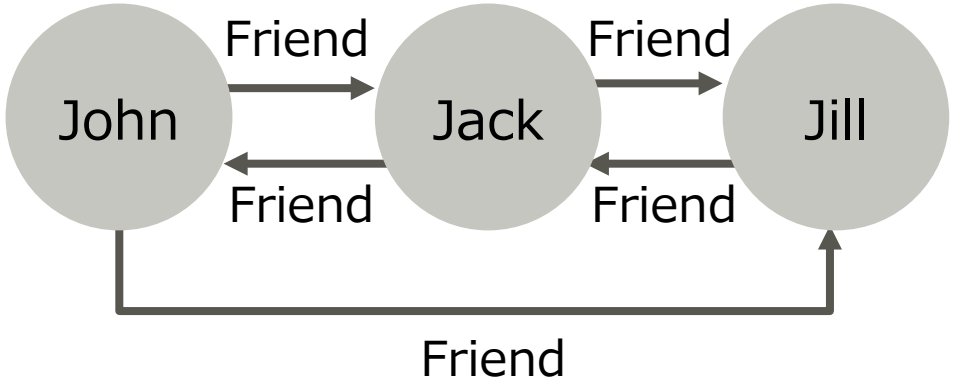
FUJITSU

- The key is performance in storage engine
  - RDB is slow to traverse graph due to index scan
  - Eliminate index scan using direct pointers between records
  - Node traversal cost drops from O(n) to O(1)

Graph in RDBMS

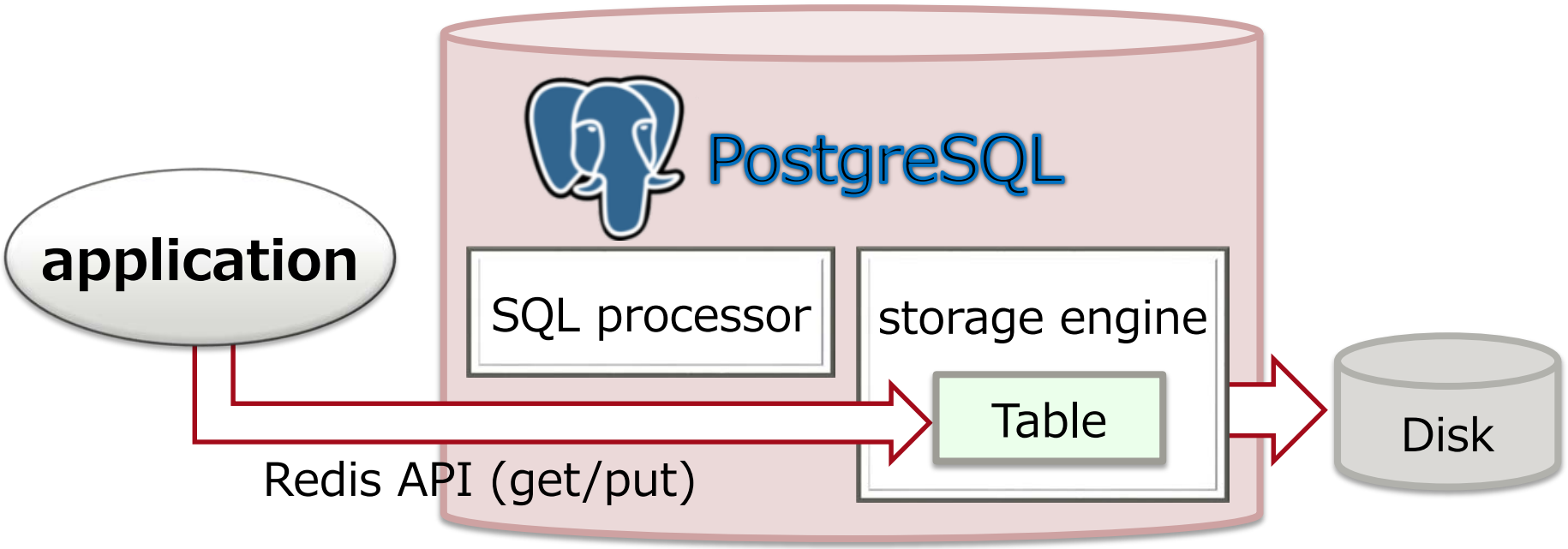


Native graph



Key-value Model

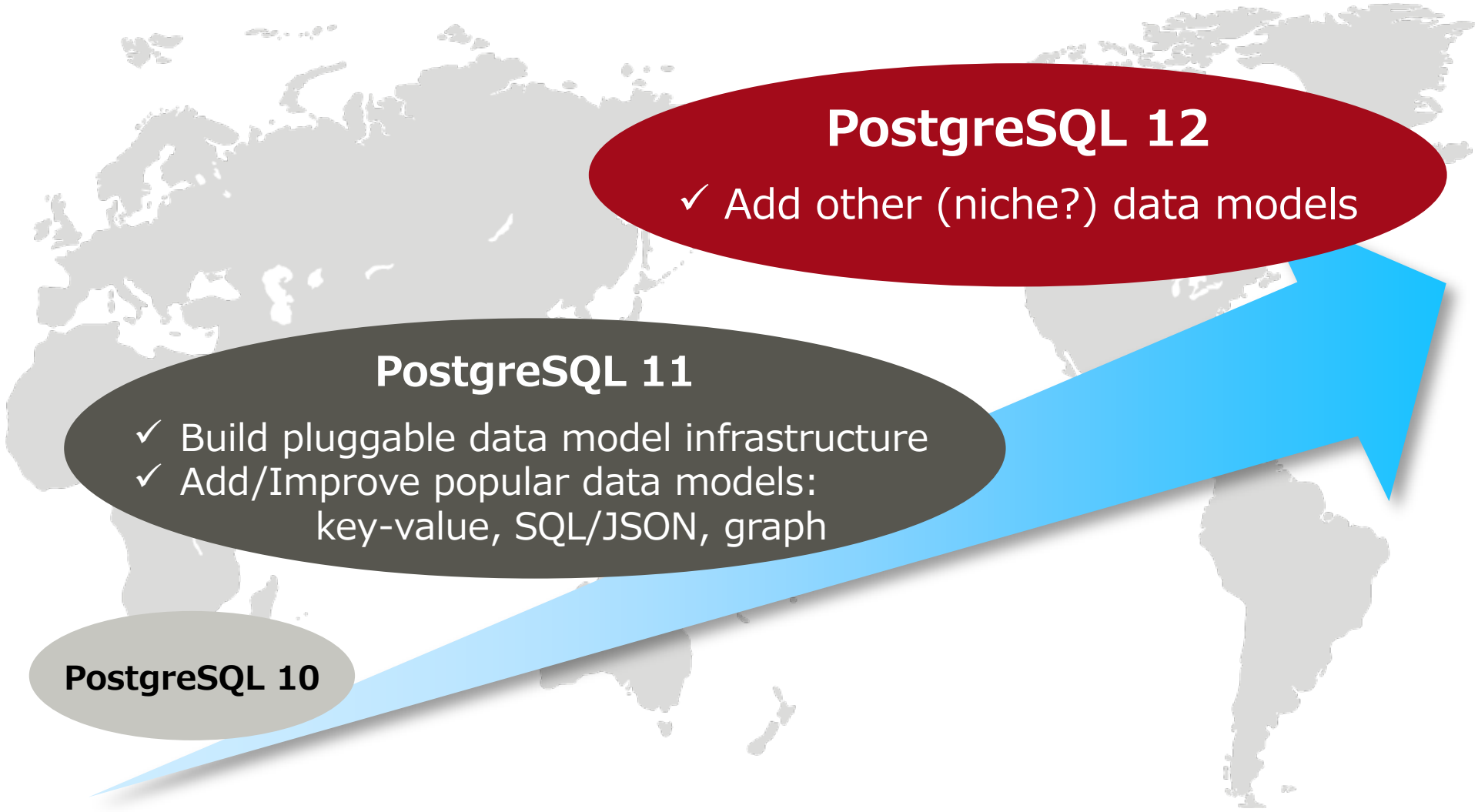
- PostgreSQL has hstore data type, but
  - Less performant than expected
  - Unfamiliar API
- Solution: Redis in the background worker
  - Maximal performance by bypassing SQL processor
  - Familiar, developer-friendly Redis API



Conclusion

FUJITSU

- Multi-model is necessary for broader use of PostgreSQL

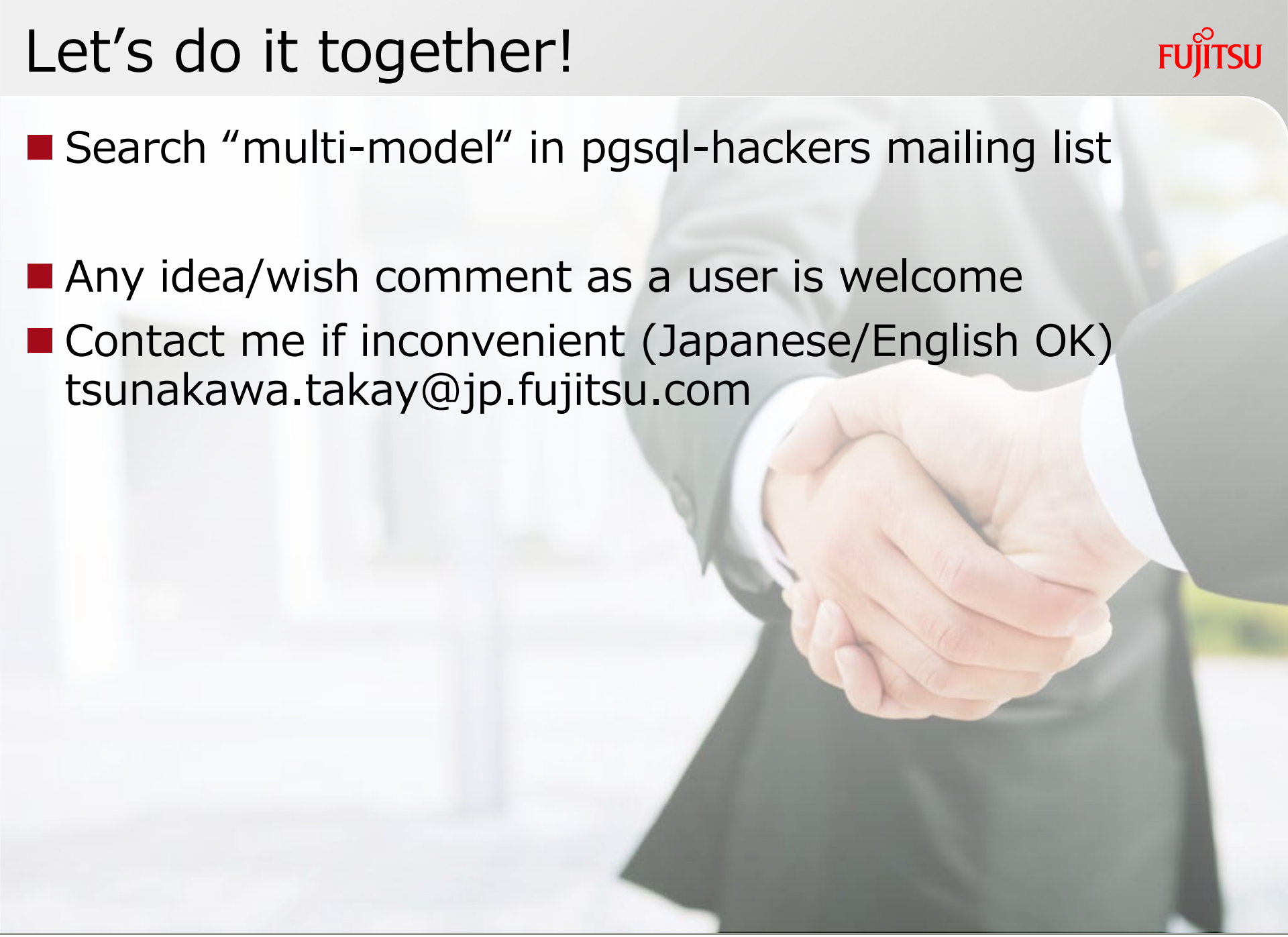


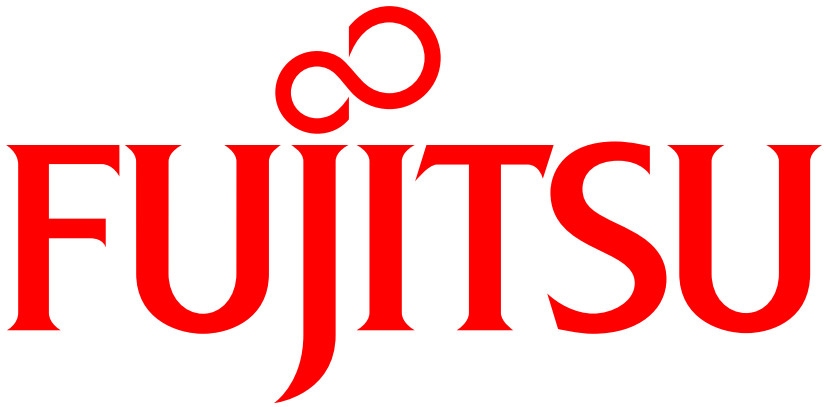
Let’s do it together!

FUJITSU

- Search “multi-model“ in pgsql-hackers mailing list

- Any idea/wish comment as a user is welcome
- Contact me if inconvenient (Japanese/English OK)
  tsunakawa.takay@jp.fujitsu.com





shaping tomorrow with you