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

December 5, 2017

Takayuki Tsunakawa
Fujitsu Limited
Who am I?

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

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

Variety

Volume

Velocity
Can PostgreSQL Handle Big Data?

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

Streaming
   PipelineDB (fork)

Persistent memory, FPGA, SIMD
   N/A

GPU
   PG-Strom (extension)
Why NoSQL Attracts Attention?

- 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

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

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

Data models in online shopping application

- **RDB**
  - customer
  - order

- **Graph**
  - recommendation

- **Document**
  - product catalog

- **Key-value**
  - Web session
  - shopping cart
  - user profile

- **Wide columnar**
  - Web access log
Multiple DBMSs Use

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

<table>
<thead>
<tr>
<th>Data model</th>
<th>DBMSs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational</td>
<td>MySQL, Redshift</td>
</tr>
<tr>
<td>Key-value</td>
<td>Memcached, Redis, Hollow (developed by Netflix)</td>
</tr>
<tr>
<td>Text search</td>
<td>Elasticsearch</td>
</tr>
<tr>
<td>Wide columnar</td>
<td>Cassandra</td>
</tr>
<tr>
<td>Time series</td>
<td>Atlas (developed by Netflix)</td>
</tr>
<tr>
<td>Event</td>
<td>Druid</td>
</tr>
</tbody>
</table>
Problems (1/2)

- 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)

- **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**
What is multi-model database?
Overview

- Support multiple data models in one DBMS

application

- RDB
- Graph
- Key-value
- Document

etc...
"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

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Supported data models</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArangoDB</td>
<td>key-value, document, graph</td>
</tr>
<tr>
<td>Cosmos DB</td>
<td>key-value, document, graph</td>
</tr>
<tr>
<td>CouchBase</td>
<td>key-value, document</td>
</tr>
<tr>
<td>DataStax(on Cassandra)</td>
<td>key-value, wide column, graph</td>
</tr>
<tr>
<td>MarkLogic</td>
<td>document, text/binary, RDF</td>
</tr>
<tr>
<td>OrientDB</td>
<td>key-value, document, graph, text/binary</td>
</tr>
</tbody>
</table>
## Trends of Major DBMSs

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

### Data model support in top 5 popular DBMSs

<table>
<thead>
<tr>
<th>DBMS</th>
<th>Key-value</th>
<th>Document</th>
<th>Wide column</th>
<th>Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>++</td>
<td>++</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>MySQL</td>
<td>++</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQL Server</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>MongoDB</td>
<td>+</td>
<td>++</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PostgreSQL as a Multi-model Database

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
How should we implement multi-model database?
## What is Data Model?

**Data model** = **Structure** + **Constraint** + **Operation**

### Data model

<table>
<thead>
<tr>
<th>Type</th>
<th>Structure</th>
<th>Constraint</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational</td>
<td>table, row, column</td>
<td>unique, referential,</td>
<td>scan, join, restriction,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>check, not null, ...</td>
<td>projection, ...</td>
</tr>
<tr>
<td>Key-value</td>
<td>key, value</td>
<td>unique</td>
<td>get, put</td>
</tr>
<tr>
<td>Graph</td>
<td>node, relationship,</td>
<td>unique, node existence</td>
<td>scan, join, restriction,</td>
</tr>
<tr>
<td></td>
<td>property, label</td>
<td></td>
<td>projection, pattern match,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Query Language and API

- 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

<table>
<thead>
<tr>
<th>Data model</th>
<th>languages/APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key-value</td>
<td>Redis API, Memcached API</td>
</tr>
<tr>
<td>Document</td>
<td>SQL/JSON path (SQL standard), MongoDB API</td>
</tr>
<tr>
<td>Graph</td>
<td>Cypher, Gremlin</td>
</tr>
<tr>
<td>RDF</td>
<td>SPARQL (W3C standard)</td>
</tr>
<tr>
<td>Array</td>
<td>SQL/MDA (Multi-Dimensional Array) (future SQL standard)</td>
</tr>
</tbody>
</table>
**Multi-model Approach 1**

- **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

Multi-model Approach 2

- 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

- **Graph model:** AgenGraph (fork)
  - [https://github.com/bitnine-oss/agensgraph](https://github.com/bitnine-oss/agensgraph)

- **Time series model:** TimescaleDB (extension)
  - [https://github.com/timescale/timescaledb](https://github.com/timescale/timescaledb)
Pluggable Data Model

- 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

- Mix queries for multiple data models in a query string
- Execute query in a specified region
  \[\text{in}_\text{region}(\text{region\_name, query string})\]
- Convert data across regions
  \[\text{cast\_region}(\text{source data, dest region name, dest container, dest schema})\]

```sql
-- 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;
```
Mixed-model Query Execution

Multi-model query plan

relation:sort

relation:join

relation:table/index scan
restaurant

graph:pattern match
IS_FRIEND_OF

graph:node scan
Person
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

```sql
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

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

- 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

<table>
<thead>
<tr>
<th></th>
<th>Jill</th>
<th>Jack</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jack</td>
<td>Jill</td>
<td>Jack</td>
</tr>
<tr>
<td>Jill</td>
<td>John</td>
<td>Jack</td>
</tr>
<tr>
<td>Jack</td>
<td>Jill</td>
<td>John</td>
</tr>
</tbody>
</table>

Native graph

- Friend
- Friend
- Friend
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

- Multi-model is necessary for broader use of PostgreSQL

**PostgreSQL 10**
- Build pluggable data model infrastructure
- Add/Improve popular data models: key-value, SQL/JSON, graph

**PostgreSQL 11**
- Add other (niche?) data models

**PostgreSQL 12**
- Add other (niche?) data models
Let’s do it together!

- 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
Fujitsu

shaping tomorrow with you
Questions?