JSON processing in Apache Ignite as cache of RDBMS

Lazarev Nikita, Borisenko Oleg
ISP RAS

Plan

- JSON processing in RDBMS
- Implementing of JSON processing in Apache Ignite
- Apache Ignite as cache to RDBMS
- Testing
- Results

This work is funded by the Minobrnauki Russia (grant id RFMEFI60417X0199, grant number 14.604.21.0199)

Introduction

JSON documents:

- Describes objects with substructure
- Flexible scheme
- Used in WEB
- Supported in some RDBMS
- Not included in SQL standard

Introduction

1. SQL tables may contain in the columns JSON type
2. Functions allowing developers generate JSON documents directly in SQL queries
3. Transformation of string data types to JSON and reverse operations
4. CAST-operator both into JSON and out of it
5. Operations to check correctness of the documents
6. Operations to work with documents directly in SQL
7. Indexing of JSON documents

Introduction

Feature	Oracle	MySQL	MS SQL Server	PostgreSQL
1	Stored in strings	Yes	Stored in strings	Yes, Binary storage is possible
2	Incompletely	Yes	Yes	Yes
3	Incompletely	Yes	Yes	Yes
4	Incompletely	Yes	Incompletely	Yes
5	Yes	Yes	Yes	Yes
6	Yes	Yes	Incompletely	Yes
7	Full text search	No	No	For binary representation

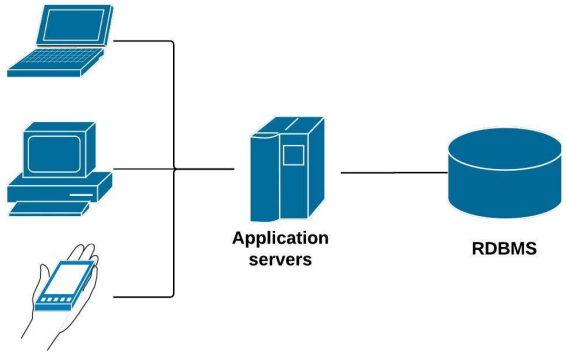
Introduction

PostgreSQL provides:

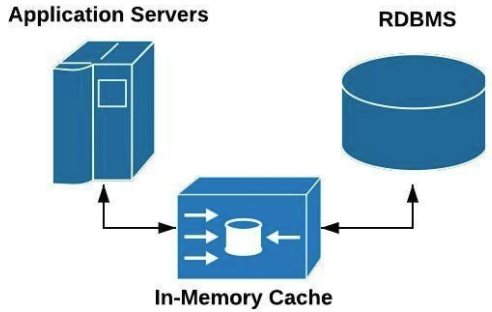
- json, jsonb data types
- json and jsonb operators:
  - “->”, “->>”, “#>”, “#>>” - get document element as document or text
- jsonb operators:
  - “@>”, “<@” - comparison of documents
  - “?”, “?|”, “?&” - key existence
  - “||”, “-”, “#-” - document modifications
- Addition JSON processing functions
- Aggregate functions
- GIN Index for jsonb

Introduction

- The cost of RAM decreases
- Increase of data processing performance
- Horizontal scaling



- CAP-theorem
- Power supply dependency



Apache Ignite

Apache Ignite is the open source version of GridGain Systems product. It’s memory-centric distributed database, caching, and processing platform designed to work on large volumes of data

- Read-/Write-Through and Write-behind operations
- ANSI SQL:1999 with support of the most dialects
- ACID compliance
- Guaranteed data consistency
- Dynamic scalability
- All the local data and indexes can be duplicated on external storage
- JCache interface, JDBC, ODBC
- Java, C++, .Net/C#, Scala, Node.JS

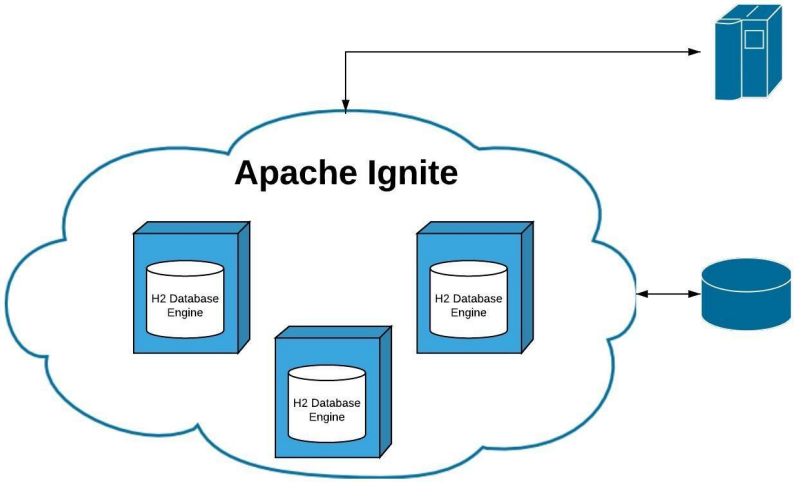
Implementing of JSON processing

For SQL querying on lower layer used H2 Database Engine:

- SQL Parsing
- Local data storage

Apache Ignite has functions:

- Data distribution
- Consistency
- Local results collection



Implementation of JSON processing

- Used Java library FasterXML/Jackson for JSON documents serialization (deserialization) and processing on lower layer
- Implemented only text storage of documents
- Implemented PostgreSQL operators for json and jsonb
- In Apache Ignite implemented JSON processing only in SELECT queries in cache mode

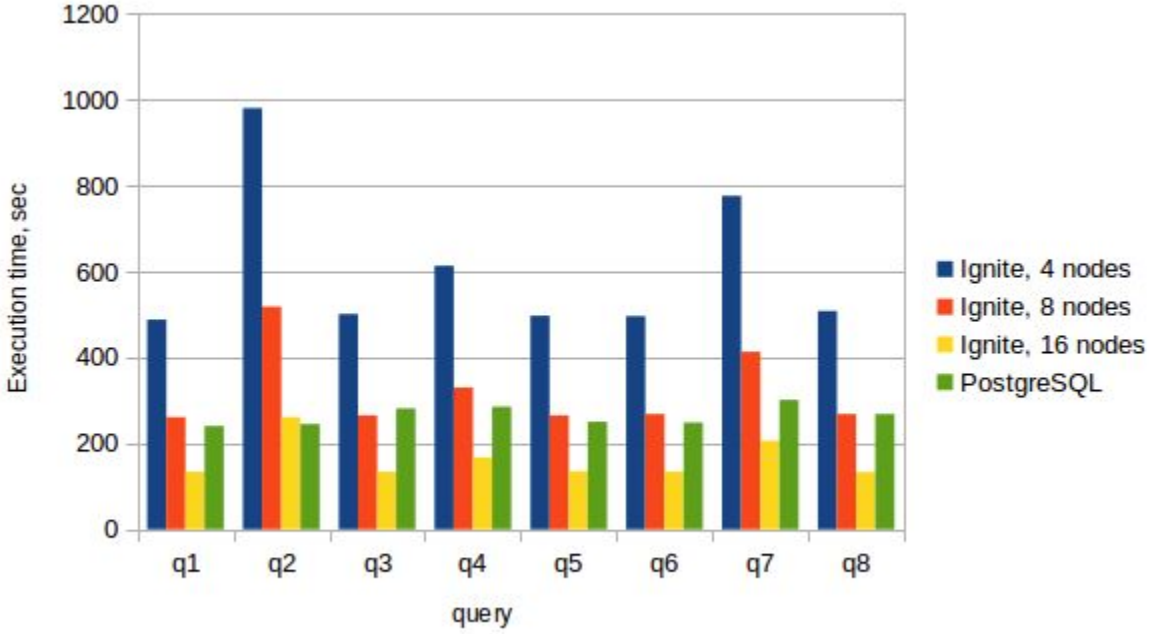
RDBMS caching

- In Apache Ignite implemented CacheStoreAdapter to synchronization with PostgreSQL
- Tested cluster configurations are 4, 8 and 16 nodes. Each one is a virtual machine with 92GB of RAM, 3 processor cores at 2.1 GHz (Intel Xeon Gold 6152) running under Xen hypervisor and each virtual machine has dedicated physical server-class SSDs
- Testing based on pg_nosql_benchmark

RDBMS caching

1. SELECT count(id) FROM table WHERE data->>’brand’ = ‘ACME’;
2. SELECT count(id) FROM table WHERE data::JSONB ? ‘name’ AND data->>’name’ = ‘AC3 Case Red’;
3. SELECT count(id) FROM table WHERE data::JSONB ?& array[‘type’, ‘name’, ‘price’];
4. SELECT count(id) FROM table WHERE data::JSONB ?& array[‘type’, ‘name’, ‘price’] AND data->>’type’ = ‘phone’;
5. SELECT count(id) FROM table WHERE (data->’limits’->’voice’->>’n’)::DECIMAL > 400;
6. SELECT count(id) FROM table WHERE (data#>>’{limits, voice, n}’)::DECIMAL> 400;
7. SELECT count(id) FROM table WHERE data::JSONB?'color' AND data->>'color' = 'black' AND data::JSONB?'price' AND (data->>'price')::DECIMAL = 12.5;
8. SELECT count(id) FROM table WHERE data::JSONB@>'{"color":"black", "price":12.5}';

Results


1200
1000
800
600
400
200
0
Execution time, sec
q1
q2
q3
q4
q5
q6
q7
q8
query
Ignite, 4 nodes
Ignite, 8 nodes
Ignite, 16 nodes
PostgreSQL

Conclusion

- Implemented JSON processing in H2 Database Engine
  - Text storage
  - PostgreSQL operators for document processing
- Implemented JSON processing in Apache Ignite with RDBMS caching
  - Not implemented Write-Through operation
  - Must be loaded full database
  - High overhead on document serialization
  - Good scaling