SQL STANDARDS UPDATE

Keith W. Hare
SC32 WG3 Convenor
JCC Consulting, Inc.
October 20, 2017
Introduction

• What is SQL?
• Who Develops the SQL Standards
• A brief history
• SQL 2016 Published
• SQL Technical Reports
• What's next?
  • SQL/MDA
  • Streaming SQL
  • Property Graphs
• Summary
Who am I?

• Senior Consultant with JCC Consulting, Inc. since 1985
  • High performance database systems
  • Replicating data between database systems

• SQL Standards committees since 1988
  • Convenor, ISO/IEC JTC1 SC32 WG3 since 2005
  • Vice Chair, ANSI INCITS DM32.2 since 2003
  • Vice Chair, INCITS Big Data Technical Committee since 2015

• Education
  • Muskingum College, 1980, BS in Biology and Computer Science
  • Ohio State, 1985, Masters in Computer & Information Science
What is SQL?

• SQL is a language for defining databases and manipulating the data in those databases
• SQL Standard uses SQL as a name, not an acronym
  • Might stand for SQL Query Language
• SQL queries are independent of how the data is actually stored – specify what data you want, not how to get it
Who Develops the SQL Standards?

In the international arena, the SQL Standard is developed by ISO/IEC JTC1 SC32 WG3.

- Officers:
  - Convenor – Keith W. Hare – USA
  - Editor – Jim Melton – USA

- Active participants are:
  - Canada – Standards Council of Canada
  - China – Chinese Electronics Standardization Institute
  - Germany – DIN Deutsches Institut für Normung e. V.
  - Great Britain – British Standards Institution
  - Japan – SQL working group of JIS (Japan Industrial Standards)
  - Netherlands
  - USA – ANSI INCITS
Translation

• ISO/IEC
  • ISO – International Organization for Standardization
  • IEC – International Electrotechnical Commission
• JTC1 – Joint Technical Committee 1
  • Information Technology
• SC32 – SubCommittee 32 – Data Management and Interchange
• WG3 – Working Group 3 – responsible for database language standards
How does WG3 Work?

• Editor prepares base documents
• Experts contribute well written proposals identifying enhancements and/or corrections
• International Ballots to progress documents
  • Committee Draft (CD)
  • Draft International Standard (DIS)
  • International Standard (IS) – published
  • National Bodies review and create comments on balloted documents
• WG3 meets several times a year to review and discuss papers/comments
• Next several photos are from recent WG3 meetings
WG3 Meeting in Cape Town South Africa
WG3 Meeting in Okayama Japan
WG3 Meeting in Granville Ohio USA
SQL Standards – a brief history

- ISO/IEC 9075 Database Language SQL
  - SQL-87 (ANSI SQL-86)
  - SQL-89
  - SQL-92
  - SQL:1999
  - SQL:2003
  - SQL:2008
  - SQL:2011
  - SQL:2016

- 30 years of support and expansion of the standard
- Started with basic SQL model have extensive expansions
- But what are we doing now to meet emerging market needs?
SQL:2016

- Published 2016-12-14
- Major features:
  - Row Pattern Recognition
    - Regular Expressions across sequences of rows
  - Support for Java Script Object Notation (JSON)
    - Store, Query, and Retrieve JSON
  - Polymorphic Table Functions
    - Parameters and function return value can be tables whose shape is not known until query time
  - Additional analytics
    - Trigonometric and Logarithm functions
# SQL:2016 Parts

<table>
<thead>
<tr>
<th>Reference</th>
<th>Document title</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO/IEC 9075-1</td>
<td>Information technology -- Database languages -- SQL -- Part 1: Framework (SQL/Framework)</td>
</tr>
<tr>
<td>ISO/IEC 9075-2</td>
<td>Information technology -- Database languages -- SQL -- Part 2: Foundation (SQL/Foundation)</td>
</tr>
<tr>
<td>ISO/IEC 9075-3</td>
<td>Information technology -- Database languages -- SQL -- Part 3: Call-Level Interface (SQL/CLI)</td>
</tr>
<tr>
<td>ISO/IEC 9075-4</td>
<td>Information technology -- Database languages -- SQL -- Part 4: Persistent stored modules (SQL/PSM)</td>
</tr>
<tr>
<td>ISO/IEC 9075-9</td>
<td>Information technology -- Database languages -- SQL -- Part 9: Management of External Data (SQL/MED)</td>
</tr>
<tr>
<td>ISO/IEC 9075-10</td>
<td>Information technology -- Database languages -- SQL -- Part 10: Object language bindings (SQL/OLB)</td>
</tr>
<tr>
<td>ISO/IEC 9075-11</td>
<td>Information technology -- Database languages -- SQL -- Part 11: Information and definition schemas (SQL/Schemata)</td>
</tr>
<tr>
<td>ISO/IEC 9075-13</td>
<td>Information technology -- Database languages -- SQL -- Part 13: SQL Routines and types using the Java™ programming language (SQL/JRT)</td>
</tr>
<tr>
<td>ISO/IEC 9075-14</td>
<td>Information technology -- Database languages -- SQL -- Part 14: XML-Related Specifications (SQL/XML)</td>
</tr>
</tbody>
</table>
SQL Technical Reports – 19075

- SQL Standards committees have accumulated a great deal of descriptive material
- Useful information (non-normative) but does not belong in the actual standard.
- Started creating Technical Reports from this material
  - First was published in 2011
  - Three published in 2016 and 2017
  - Total of seven are now published
- The current list of Technical Reports is:
# SQL Technical Reports

<table>
<thead>
<tr>
<th>Reference</th>
<th>Document title</th>
<th>Publication Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO/IEC TR 19075-5</td>
<td>Information technology -- Database languages -- SQL Technical Reports -- Part 5: Row Pattern Recognition in SQL</td>
<td>2016-12-14</td>
</tr>
<tr>
<td>ISO/IEC TR 19075-6</td>
<td>Information technology -- Database languages -- SQL Technical Reports -- Part 6: SQL support for JSON</td>
<td>2017-03-29</td>
</tr>
<tr>
<td>ISO/IEC TR 19075-7</td>
<td>Information technology -- Database languages -- SQL Technical Reports -- Part 7: SQL Support for Polymorphic Table Functions</td>
<td>2017-03-29</td>
</tr>
<tr>
<td>ISO/IEC PDTR 19075-8</td>
<td>Information technology -- Database languages -- SQL Technical Reports -- Part 8: SQL Support for Multi-Dimensional Arrays</td>
<td>2018?</td>
</tr>
</tbody>
</table>
SQL Technical Reports

- Available from JTC1 Freely Available Standards page:
  - Search for 19075
  - Must agree to single use license
SQL Futures – Napkin of Things

- Block Chards
- High level Framework
- Bolt on
- Incremental accuracy
  - "Polyglot" Data Models
  - Safe Iteration
- Better Analytics / Approx. QA
  - Uncertainty of Purpose of Data
- Security / Privacy
- Streaming
- MDA 2017-2017
- Futures
What's next?

SC32 WG3 is adding support to the SQL standards in the following areas:

- Multi Dimensional Arrays – SQL/MDA
- Streaming SQL
- Property Graphs
SQL/MDA Multi-Dimensional Arrays

- Support for complex operations on multi-dimensional arrays within the context of an SQL database
- Official Designation is 9075-15 SQL/MDA – Multi-Dimensional Arrays
- Draft International Standard (DIS) ballot
  - Starts 2017-11-13
  - Ends 2018-02-05
- International Standard should be completed mid 2018
- Major components:
  - MDArray type
  - MDArray Operators
MDArray Type

- Multiple Named dimensions
- Minimum and Maximum for each dimension
- Single datatype
  - Could be most existing datatype – integer, float, character, etc.
  - An MDArray can not currently have a type of
    - MDArray
    - Binary/Character Large Object (LOB, BLOB, CLOB)
    - Support for additional types could be added in a future version
MDArray Operators

Operators iterate across all elements specified by parameters
• Reshape
• Scale
• Concatenate
• Mathematical operations
• Convert an array to a table
• Convert a table to an array

Return an array as a specific MIME type
• Minimal required set of MIME types
• Vendors will add MIME types to support problem domains
Why support Multi Dimensional Arrays?

• Power to process
  • massive images
  • seismic data
  • oceanographic data
  • etc.
• Process arrays on server, return only interesting parts
  • Reduce network load
• Transaction support
• Enable database management capabilities
Support for Streaming Data

- Initial discussion of requirements and specifications
- Support processing on:
  - Input from Internet of Things (IoT)
  - Video
  - Audio
  - Time Series Data
- Process data before or instead of storing
  - Continuous processing
  - Zero or more input streams
  - Process data
  - Zero or more output streams
Streaming Data – General Idea

• Map incoming stream(s) of data – subscribe
• Map outgoing stream(s) of data – publish
• Process stream(s) using all of the existing SQL capabilities, including
  • SQL Queries, Insert, Update, Merge
  • Compound Statements
  • Stored Procedures
  • Row Pattern Recognition
  • JSON
  • Polymorphic Table Functions
  • Multi Dimensional Arrays
• Potentially 2019/2020
Support for Graph Databases

• Graph data often represented as triples
  • <subject> <predicate> <object>
    <Hare> <speaking to> <database class>
    <Baumann> <instructor for> <database class>
    <Hare> <convenor of> <SC32/WG3>
    <SC32/WG3> <produces> <SQL Standards>
    <Melton> <Editor of> <SQL Standards>

• Would it be beneficial to have an SQL interface to graph data?
Existing Graph Database Language Efforts

- RDF Triples
  - W3C graph query language SPARQL
- Property Graphs
  - Linked Data Benchmark Council (LDBC) Query Language task force
  - openCypher Project
  - Vendors
  - INCITS DM32.2 Ad Hoc on SQL Extensions for Property Graphs
W3C graph query language SPARQL

- Resource Description Framework (RDF) Triples
- SPARQL (SPARQL Protocol and RDF Query Language)
- Well defined by W3C

https://www.w3.org/TR/sparql11-overview/
Linked Data Benchmark Council (LDBC) Query Language task force

- Property Graphs
- Tenth TUC (Technical User Committee) Meeting, TU Munich at VLDB2017
  - [http://wiki.ldbcouncil.org/display/TUC/Tenth+TUC+Meeting%2C+TU+Munich+at+VLDB2017](http://wiki.ldbcouncil.org/display/TUC/Tenth+TUC+Meeting%2C+TU+Munich+at+VLDB2017)
- Goals
  - Recommend a query language core that could be incorporated in future versions of industrial graph query languages.
  - Perform deep academic analysis of the expressiveness and complexity of evaluation of the query language
    - To ensure a powerful yet practical query language
openCypher Project

• Property Graphs
• The following is from the openCypher web page
  http://www.opencypher.org/

  The **openCypher project** aims to improve growth and adoption of
graph processing and analysis by providing an open graph query
language to any data store, tooling or application provider as a
mechanism to query graph data.
  • Provides a query language with full support
  • Makes graph processing and analysis easier to adopt
  • Grants vendor independence to all users
  • Eases graph integration with other data platforms

• Next several slides use the NEO4J openCypher implementation
Property Graph Example: Create Nodes

neo4j> `Create (:project:normative {project: "9075"});`
0 rows available after 17 ms, consumed after another 2 ms
Added 1 nodes, Set 1 properties, Added 2 labels
neo4j> `Create (:project:technicalreport {project: "19075"});`
0 rows available after 10 ms, consumed after another 0 ms
Added 1 nodes, Set 1 properties, Added 2 labels
neo4j> `Create (:project:technicalreport {project: "29075"});`
0 rows available after 4 ms, consumed after another 0 ms
Added 1 nodes, Set 1 properties, Added 2 labels
neo4j> `match (n:project) return n;`

```
+-----------------------------------------------+
| n                                             |
+-----------------------------------------------+
| (:normative:project {project: "9075"})        |
| (:technicalreport:project {project: "19075"}) |
| (:technicalreport:project {project: "29075"}) |
+-----------------------------------------------+
```

3 rows available after 12 ms, consumed after another 1 ms
Nodes, Labels, and Properties

In the example:

```sql
Create (:project:normative {project: "9075"});
```

- Nodes denoted by parenthesis ()
- Labels prefixed by colons
  - :project
  - :normative
  - Can have multiple labels
- Properties denoted by brackets {}
  - {project: "9075"}
  - Can have multiple properties
Query the Projects

```
neo4j> match (n:technicalreport) return n;
+-----------------------------------------------+
| n                                              |
| (:technicalreport:project {project: "19075"}) |
| (:technicalreport:project {project: "29075"}) |
+-----------------------------------------------+
2 rows available after 15 ms, consumed after another 1 ms

neo4j> match (n:normative) return n;
+----------------------------------------+
| n                                      |
| (:normative:project {project: "9075"}) |
+----------------------------------------+
1 row available after 17 ms, consumed after another 0 ms
```

2017-10-20
Create an SC and some WGs

neo4j> Create (:SC:committee {name:"SC32", title:"Data Management and Interchange"});
0 rows available after 19 ms, consumed after another 0 ms
Added 1 nodes, Set 2 properties, Added 2 labels
neo4j> create (:WG:committee {name:"WG1", title:"E-Business"});
0 rows available after 12 ms, consumed after another 0 ms
Added 1 nodes, Set 2 properties, Added 2 labels
neo4j> create (:WG:committee {name:"WG2", title:"Metadata"});
0 rows available after 2 ms, consumed after another 0 ms
Added 1 nodes, Set 2 properties, Added 2 labels
neo4j> create (:WG:committee {name:"WG3", title:"Database Languages"});
0 rows available after 1 ms, consumed after another 0 ms
Added 1 nodes, Set 2 properties, Added 2 labels
neo4j> create (:WG:committee {name:"WG4", title:"MultiMedia"});
0 rows available after 1 ms, consumed after another 0 ms
Added 1 nodes, Set 2 properties, Added 2 labels
neo4j>
Query the Committees

```sql
neo4j> match (n:committee) return n;
+--------------------------------------------------------------------------+
| n                                                                        |
+--------------------------------------------------------------------------+
| (:SC:committee {name: "SC32", title: "Data Management and Interchange"}) |
| (:WG:committee {name: "WG1", title: "E-Business"})                     |
| (:WG:committee {name: "WG2", title: "Metadata"})                       |
| (:WG:committee {name: "WG3", title: "Database Languages"})             |
| (:WG:committee {name: "WG4", title: "MultiMedia"})                     |
+--------------------------------------------------------------------------+
5 rows available after 1 ms, consumed after another 3 ms
```

```sql
neo4j> match (n:WG) return n;
+------------------------------------------------------------+
| n                                                          |
+------------------------------------------------------------+
| (:WG:committee {name: "WG1", title: "E-Business"})         |
| (:WG:committee {name: "WG2", title: "Metadata"})           |
| (:WG:committee {name: "WG3", title: "Database Languages"}) |
| (:WG:committee {name: "WG4", title: "MultiMedia"})         |
+------------------------------------------------------------+
```
Create a relationship between the WGs and SC32

```
neo4j> MATCH (n:WG), (s:SC {name:"SC32" })
    CREATE (s)-[Pof:parent_of]->(n)
RETURN s,Pof, n;
```

<table>
<thead>
<tr>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>(:SC:committee {name: &quot;SC32&quot;, title: &quot;Data Management and Interchange&quot;)</td>
</tr>
<tr>
<td>[:parent_of]</td>
</tr>
<tr>
<td>(:SC:committee {name: &quot;SC32&quot;, title: &quot;Data Management and Interchange&quot;)</td>
</tr>
<tr>
<td>[:parent_of]</td>
</tr>
<tr>
<td>(:SC:committee {name: &quot;SC32&quot;, title: &quot;Data Management and Interchange&quot;)</td>
</tr>
<tr>
<td>[:parent_of]</td>
</tr>
<tr>
<td>(:SC:committee {name: &quot;SC32&quot;, title: &quot;Data Management and Interchange&quot;)</td>
</tr>
<tr>
<td>[:parent_of]</td>
</tr>
</tbody>
</table>

4 rows available after 6 ms, consumed after another 5 ms
Created 4 relationships
neo4j>
```
Building Relationships

neo4j> MATCH (n:WG), (s:SC {name:"SC32" })
    CREATE (s)-[Pof:parent_of]->(n)
    RETURN s,Pof, n;

• Query for two sets of nodes
  • MATCH (n:WG), (s:SC {name:"SC32" })
  • Cartesian product

• Create relationship (edges):
  • CREATE (s)-[Pof:parent_of]->(n)
  • s, PoF, and n are arbitrary correlation names (binding variables)

• Return the data specified by correlation names
  • RETURN s,Pof, n
  • Command line interface does not adequately represent graph
Graphical display of Graph

- WG2
- SC32
- WG3
- WG4
- WG1
How does this fit with SQL?

Current discussions include:

- Property graph queries that return tabular data as a part of a tabular query
- Tabular queries that return property graph format data as a part of a property graph query
- Insert, update, and delete property graph nodes and edges (relationships)
- How is the data effectively stored?
  - Single “big” graph?
    - One graph per database
    - May be disconnected segments
  - Multiple “small” graphs?
    - Store a graph in a column for each row in a table
Why integrate Property Graphs with SQL?

• Users are using both SQL databases and Property Graph databases
• Application development is easier, better, quicker, faster if only one interface
When will it be real?

• Working Draft in 2018
• International Standard in 2019 or 2020
• Database vendors working on property graph support now
Summary

• SQL Standards have a long history
  • 30 years of experience integrating new technologies
  • Used for large number of existing applications
  • Unlikely to be replaced in the near future

• Recently completed SQL Enhancements
  • Row Pattern Recognition
  • JSON
  • Polymorphic Table Functions
  • Additional analytics

• SQL Technical Reports

• Future Enhancements
  • Multi Dimensional Arrays – SQL/MDA
  • Streaming SQL
  • Property Graphs
Questions?

• Select Questions From Audience;

• MATCH(a:Audience) -[q:Questions_for]->(s:Speaker) RETURN a, q, s;

Keith W. Hare
JCC Consulting, Inc.
600 Newark Granville Road
Granville Ohio 43023 USA
Keith@jcc.com
www.jcc.com