On UNIQUE KEYS in SQL/JSON Implementations

Experimenting with JSON support in DBMS products

Martti Laiho
DBTechNet
www.dbtechnet.org
formerly at Haaga Helia
email: martti.laiho@gmail.com

Fritz Laux (retired)
Fakultät Informatik
Reutlingen University
D-72762 Reutlingen, Germany
email: fritz.laux@reutlingen-university.de

Martti Laiho

Martti Laiho was educated at Helsinki University and after a career of 20 years in a data service centre of pension funds, is now a retired senior lecturer from Haaga-Helia UAS Helsinki where he taught SQL, data access programming, and database administration over 30 years.

He is a cofounder of DBTechNet.org, a European initiative of academics and industry to improve and promote database education. He has been project manager in various EU-funded projects of DBTechNet member institutions arranging workshops and acting as lead writer of tutorials on database technologies.

His main interest has been transaction programming using various programming languages on services provided by the mainstream DBMS systems. Recently he has been focusing on JSON in MongoDB and SQL/JSON implementations in RDBMS systems.

Prof. Dr. Fritz Laux (retired)

Fritz Laux received a Ph.D. (Dr. rer. nat.) in Mathematics and worked as researcher at the Institute of Astrophysics at the University of Tübingen. Later, he worked as a system analyst in industry and was responsible for project management, database modelling and tuning. As software-architect he designed commercial information systems for international companies and Swiss government agencies. From 1986 - 2015 he was a full professor of database and information systems. Together with colleagues he established the departments of Business Informatics and Media- and Communication Informatics. His teaching record includes
Computer Networks, Object Oriented Technologies, Database and Information Systems, Transaction Processing, Post-relational Databases, Data Warehousing, Business Intelligence, and Data Mining. During his 29 years of academic work, he supervised hundreds of students, among them many which became successful as managers, researcher and academics. In 2012 he received a research award from his university. He retired in 2015, but is still active as researcher.

His research interests include
Information modelling and data integration
Transaction management and optimistic concurrency control
Business intelligence and knowledge discovery

Prof. Laux has published a number of papers in peer reviewed conferences and journals. Together with Malcolm Crowe he made contributions on Big Live Data and developed an hypergraph model with schema support, called Typed Graph Model (TGM). Recently Malcolm Crowe implemented TGM on top of his Pyrrho DBMS.

He served as program chair, member of program committees, and reviewer for several conferences and has been actively supporting IARIA by contributing papers to DBKDA and PATTERNS conferences that received DBKDA 2009 and DBKDA 2010 Best Paper Awards. He is a panellist, keynote speaker, and member of the DBKDA advisory board.

Prof. Laux is a founding member of DBTech.net (http://www.dbtechnet.org/), an initiative of
European universities and IT-companies to set up a transnational collaboration scheme for Database teaching. Together with colleagues from 5 European countries he worked on projects supported by the European Union on state-of-the-art teaching and hands-on labs on database technology towards harmonizing the professional and university level of database education in Europe. He is an IARIA fellow and member of the German Computer Society (Gesellschaft für Informatik).

RDBMS systems, developments, future ..

- Relational Model as theoretical basis for data and query languages
- SQL standard unifying implementations and knowledge portability
- Robust data platforms in terms of
  - Reliability, transactionality, security, performance, distribution, ..
- Data types extending the relational data model of SQL
  - CLOB – character large objects
  - BLOB – binary large objects
  - XML – SQL/XML, Xpath, ..
    - path expressions, filter expressions, ..
  - JSON – SQL/JSON adapting JavaScript Object Notation from IETF RFC 8259
    - text-based, language-independent data interchange format

A sample JSON object

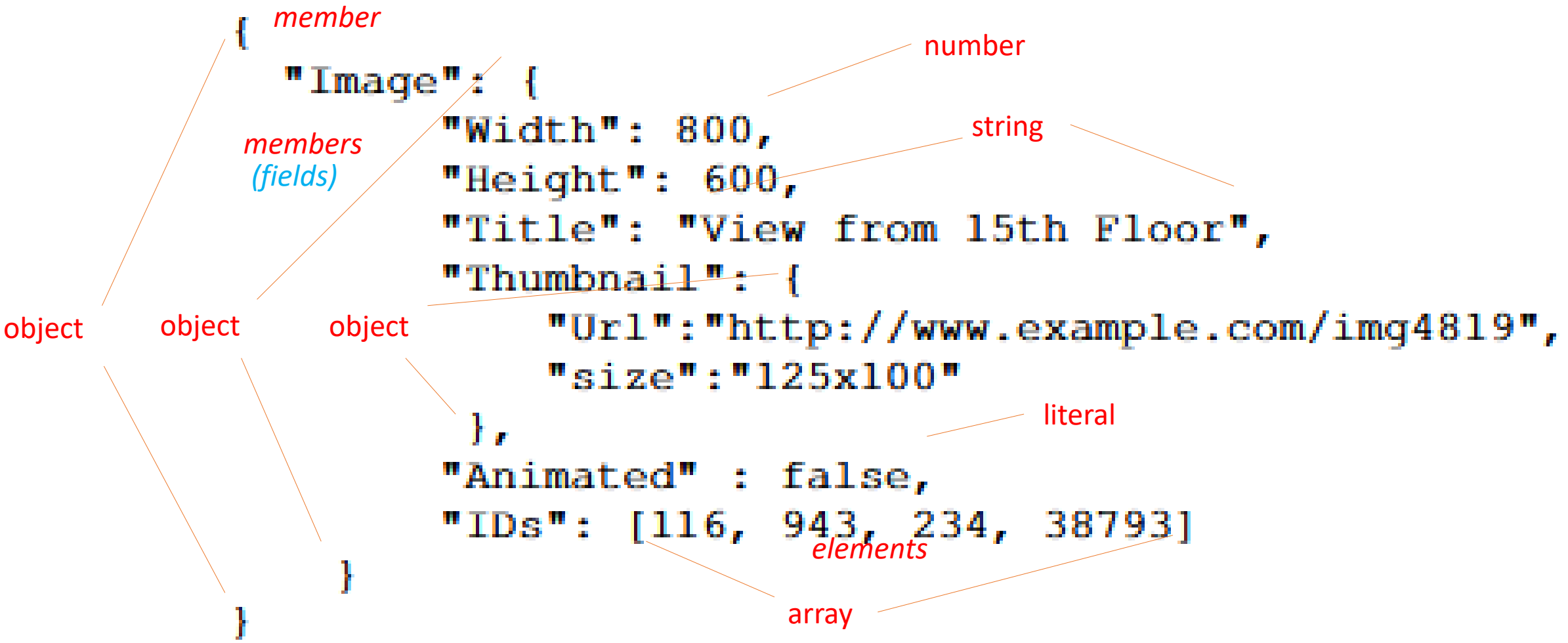
- using text-based, case-sensitive language - independent of programming languages

```
{
  "Image": {
        "Width": 800,
        "Height": 600,
        "Title": "View from 15th Floor",
        "Thumbnail": {
            "Url":"http://www.example.com/img4819",
            "size":"125x100"
         },
        "Animated" : false,
        "IDs": [116, 943, 234, 38793]
    }
}
```

On SQL/JSON standard

- Data Model adapted from IETF RFC 8259
- Storage
  - Left to implementations – CLOB, BLOB, native JSON/BSON?
- JSON Query functions
  - IS JSON – validation predicate with options
    - IS JSON WITH UNIQUE KEYS
    - IS JSON WITHOUT UNIQUE KEYS - left as default 
  - JSON_EXISTS – JSON matching predicate
  - JSON_VALUE – extracting a single value
  - JSON_QUERY – extraction a query of values
  - JSON_TABLE – mapping JSON values to SQL table format
  - Aggregate functions ...
  - *JSON_KEYS* – directory of member keys in JSON object
    - missing in standard, but implemented in MySQL/MariaDB 
- Data Manipulation
  - Left to implementations => “Babel effect” messy 
  - “to-do” developments ..?

JSON - structure, concepts, name/value pairs ("key":value)



A JSON test document of scalar and structured data types

Document object

(top-level) Members:
- *IETF RFC: unordered collection*
  but in practice: append only list
- *found by key only*

Primitive (scalar) examples:

Structured (nested) examples:

```
'{
 "mem1": 123,
 "mem2": "123",
 "mem3": true,
 "mem4": null,
 "mem5": [ 123, "123", true, false, null, [1, 2], {} ],
 "mem6": { "m61":1, "m62":"123", "m63":true, "m64":null,
           "m65": [2, 3], "m66": {} }
}'
```

Array of elements:
- *ordered list of values*
- *without keys*
- *found by position only*
- *append only*
- *delete by set to null*

object with members

empty object

Sample JSON column “J” in an SQL table

Adapted from SQL/JSON Part 2 proposal

```
CREATE TABLE T(
K  INT NOT NULL PRIMARY KEY,
J  -- BLOB|CLOB|VARCHAR() in Db2
   -- JSON|BLOB|CLOB|VARCHAR2() in Oracle
   JSON -- in MySQL/MariaDB
   JSONB – in PostgreSQL
   -- NVARCHAR() in SQL Server, JSON expected?
   IS JSON WITH UNIQUE KEYS -- different implementations
);
```

On WITH UNIQUE KEYS clause

- if missing, the default is WITHOUT UNIQUE KEYS
  allowing duplicate key names within members of an object
- Below we will test the implications

On the WITHOUT UNIQUE KEYS clause

IETF RFC 8259 on JSON objects:
"The names within an object SHOULD be unique."
"When the names within an object are not unique, the behavior of software that receives such an object is unpredictable. Many implementations report the last name/value pair only. Other implementations report an error or fail to parse the object, and some implementations report all of the name/value pairs, including duplicates."

SQL/JSON part 2 of **ISO/IEC JTC1/SC32 WG3:**
"Thus it is recommended that JSON objects should have unique key name, but it is not required. We decided that there may exist JSON texts that do not enforce this uniqueness constraint, and also that users may be interested in enforcing the uniqueness constraint. Therefore we propose two options on the IS JSON predicate, to either enforce the constraint or not. The two choices are illustrated as follows:

```
T.C IS JSON WITH UNIQUE KEYS
T.C IS JSON WITHOUT UNIQUE KEYS
```

Since enforcing a constraint is costly, the default is not to check, that is, T.C IS JSON is equivalent to T.C IS JSON WITHOUT UNIQUE KEYS."

*By adopting the IETF RFC's "SHOULD" statement into SQL/JSON extension of SQL standard, the WG3 forgets the **reliability requirement in RDBMS context.** The "costly implementation" does not justify the decision made by WG3.*
According to documentations, Oracle and PostgreSQL have implemented the constraint.

Testing implications of allowing duplicate key names

```
INSERT INTO T (K, J) VALUES (3, '{ "duplica":"First", "duplica":"Second", "duplica":"Third","duplica":"Last"}');
```

Note: implementations manage the list of members ordered

Db2 LUW:

```
SQL16406N  JSON data has non-unique keys.
```

error

Oracle 23ai:

```
SQL Error: ORA-40473: duplicate key names 'duplica' in JSON object
```

error

SQL Server XE:

```
SELECT LEFT(K, 4) AS K, JSON_VALUE(J, '$.duplica') AS Duplica FROM T1 WHERE K=3;
  K    Duplica
  ---- ----------
  3    First
  (1 row affected)
```

First Second Third Last

MariaDB:

```
SELECT LEFT(K, 4) AS K, JSON_VALUE(J, '$.duplica') AS Duplica FROM T1 WHERE K=3;
+------+---------+
| K    | Duplica |
+------+---------+
| 3    | First   |
+------+---------+
1 row in set (0.000 sec)
```

First Second Third Last

All data in these members is hidden!

PostgreSQL:

```
(SELECT J FROM T1 WHERE K=3);
                 j
-----------------
{"duplica": "Last"
(1 row)
```

First Second Third Last

All data in these members is lost!

Conclusions

- The tested database products behave differently
- Application developers trusting in default behaviour of the used RDBMS may build applications, which will lose data from databases due to wrong default decision on WITHOUT UNIQUE KEYS by ANSI WG3 and RDBMS implementors
- Great care is required to prevent duplicate keys in storing members of the same object, either by using the WITH UNIQUE KEYS option or applying corresponding constraint
- The SQL/JSON standard and implementations should be fixed accordingly to protect data integrity in databases

References

- T. Bray (ed.), ”The JavaScript Object Notation (JSON) Data Interchange Format”, URL: https://dl.acm.org/doi/pdf/10.17487/RFC8259, Dec. 2017, (Accessed Dec 1, 2025)
- G. Turutin and M. Puzevich, ”PostgreSQL JSONB-based vs. Typedcolumn Indexing: A Benchmark for Read Queries”, IEEE Dataport, October 29, 2025, doi:10.21227/fxws-3a11
- Z. H. Liu, B. Ch. Hammerschmidt, D. Mcmahon, H. J. Chang, Y. Lu, J. Spiegel, A. Colunga Sosa, S. Suresh, G. Arora, and V. Arora, ”Native JSON Datatype Support: Maturing SQL and NoSQL convergence in Oracle Database”. Proc. VLDB Endow. 13(12): 3059-3071 (2020)
- N.N., IBM, ”Uniqueness controls for key names”, Apr. 2023, URL: https://www.ibm.com/support/pages/json-uniqueness-controls-keynames
- D. Petkovic, “SQL/JSON Standard: Properties and Deficiencies”, Datenbank-Spektrum, Volume 17, pp 277-287, Oct 24 2017, URL:https://link.springer.com/content/pdf/10.1007/s13222-017-0267-4.pdf
- D. Petkovic, ”Implementation of JSON Update Framework in RDBMSs”, International Journal of Computer Applications Foundation of Computer Science (FCS), NY, USA, Volume 177 - Number 37, 2020, DOI: 10.5120/ijca2020919881
- N. N., ISO/IEC 21778:2017 ”Information technology - The JSON data interchange syntax”, URL: https://www.vde-verlag.de/iecnormen/download-kostenloses-dokument/225146/ (Accessed Dec 1, 2025)
- N. N., ISO/IEC 19075-6:2021 ”Information technology — Guidance for the use of database language SQL Part 6: Support for JSON”, Edition 1,URL: https://www.iso.org/standard/78937.html (Accessed Dec 1, 2025)
- N.N., ISO/IEC 9075-2:2023 ”Information technology — Database languages SQLPart 2: Foundation (SQL/Foundation)”, URL: https://www.iso.org/standard/76584.html, Edition 6, 2023, URL: https://www.iso.org/standard/76584.html