

# Big Data and the Multi-model Database

Heli Helskyaho, Elise Valin-Raki

# Introduction, Heli

- \* Graduated from University of Helsinki (Master of Science, computer science), currently a doctoral student, researcher and lecturer (databases, Big Data, Multi-model Databases, methods and tools for utilizing semi-structured data for decision making) at University of Helsinki
- \* Worked with Oracle products since 1993, worked for IT since 1990
- \* Data and Database!
- \* CEO for Miracle Finland Oy
- \* Oracle ACE Director
- \* Ambassador for EOUC (EMEA Oracle Users Group Community)
- \* Public speaker and an author
- \* Winner of Devvy for Database Design Category, 2015
- \* Author of the book Oracle SQL Developer Data Modeler for Database Design Mastery (Oracle Press, 2015), co-author for Real World SQL and PL/SQL: Advice from the Experts (Oracle Press, 2016)

ORACLE®

# Oracle SQL Developer Data Modeler for Database Design Mastery

Design, Deploy, and Maintain World-Class Databases  
on Any Platform

**Heli Helskyaho**

Oracle ACE Director

Forewords by C.J. Date and Tom Kyte

Oracle  
Press®

ORACLE®

# Real World SQL & PL/SQL

Advice from the Experts

Arup Nanda

Brendan Tierney

Heli Helskyaho

Martin Widlake

Alex Nuijten

Oracle  
Press®

# Introduction, Elise

- \* Oracle Database Lead for Fennia Insurance Company
- \* Master of Science degree from Helsinki High School of Economics (with major in Management Science)
- \* IT sector since 1987, different positions
- \* OUGF (Oracle User Group Finland):
  - \* Exa-SIG founder
- \* Oracle ACE Associate
- \* DBA

# References

- \* [1] Marcello Buoncristiano, Giansalvatore Mecca, Elisa Quintarelli, Manuel Roveri, Donatello Santoro, Letizia Tanca: Database Challenges for Exploratory Computing. SIGMOD Record 44(2): 17-22 (2015).
- \* [2] Stratos Idreos, Olga Papaemmanouil, Surajit Chaudhuri: Overview of Data Exploration Techniques. SIGMOD Conference 2015: 277-281.
- \* [3] Zhen Hua Liu, Dieter Gawlick: Management of Flexible Schema Data in RDBMSs - Opportunities and Limitations for NoSQL. 7th Biennial Conference on Innovative Data Systems Research (CIDR '15) January 4-7, 2015, Asilomar, California, USA.
- \* [4] Z. H. Liu, B. Hammerschmidt, D. McMahon, Y. Liu, H.J.Chang: Closing the functional and Performance Gap between SQL and NoSQL. SIGMOD Conference 2016: 227-238
- \* [5] Z. H. Liu, B. Christoph Hammerschmidt, D. McMahon: JSON data management: supporting schema-less development in RDBMS. SIGMOD Conference 2014: 1247-1258
- \* [6] <https://docs.oracle.com/database/122/ADJSN/json-dataguide.htm#ADJSN-GUID-219FC30E-89A7-4189-BC36-7B961A24067C>

# The history of data models, briefly

- \* Solutions for *storing and retrieving data*
- \* The *network* data model, the *hierarchical* data model
- ...
- \* The *relational* data model
- \* The *relational database* management system (RDBMS)
  - \* has been de facto as a data model solution for decades
  - \* based on solid theory
  - \* standardized environment for storing and retrieving data

# The evolution...

- \* of software development and programming languages leads to new demands for data models



# Objects

- \* The need to save and retrieve *objects* easily using object-oriented programming languages
- \* -> Object databases
- \* -> Later these features were added to a RDBMS making it an ORDMS (Object-Relational DataBase Management System)
  
- \* Supported in Oracle Database



# XML (Extensible Markup Language)

- \* To transfer data in a standardized way
- \* The need to save and retrieve *XML documents* easily
- \* XML Databases
  - \* Later these features were added to the RDBMS
- \* Supported in Oracle Database (XMLType and its methods, SQL functions, indexing, registering XML Schema,...)

# Spatial

- \* The need to save and retrieve *spatial data* easily
- \* Spatial Databases
  - \* Later these features were added to the RDBMS
- \* Supported in Oracle Database

# New demand for a Data Model: FSD

- \* Relational Data: "schema first, data later"
- \* Flexible Schema Data (FSD): "data first, schema later/never"

# NoSQL

- \* An "umbrella" for different solutions, each for a certain purpose/problem
  - \* Document Stores
  - \* Key-value pair stores
  - \* Column stores
  - \* Graph stores
- \* No ACID (Atomicity, Consistency, Isolation, and Durability) but maybe BASE (Basically Available, Soft state, Eventual consistency).
- \* Scale-up vs scale-out

# Document Stores

- \* JSON for Javascript programming, lighter version of XML
- \* The need to save and retrieve *JSON (Javascript Notation) documents* easily
- \* Supported in Oracle Database (CLOB, NCLOB)

# Key-value pair stores

- \* something between semi-structured and text, key is structured and value is text
- \* Supported in Oracle Database (+Oracle NoSQL)

# Column stores

- \* Data stored in columns, not rows
- \* Supported in Oracle Database (in-memory column store)



# Graph stores

- \* About relationships
  - \* Nodes/vertices, links/edges and properties
    - \* Each vertex and each edge has a collection of key-value properties
  - \* A graph can be created from a relational model quite easily
- \* Supported in Oracle Database
- \* To me the most interesting of these NoSQL solutions....
- \* But worth a presentation of its own

# NoSQL

- \* Slowly these features have been added to the RDBMS...
- \* But can this continue?
  - \* New feature-> new db-> added to RDBMS

# Gartner, Critical Capabilities for Operational Database Management Systems, Dec 9<sup>th</sup> 2015

## Key Findings

- \* Operational DBMS use cases have *expanded beyond traditional transactions* — still the leading revenue generator — to encompass three additional use cases: distributed variable data; lightweight events and observations; and hybrid transactional/analytical processing (HTAP).
- \* Relational operational DBMS products and emerging NoSQL and multimodel offerings are entering the market to address some of these new use cases, creating opportunities for best-fit engineering, but threatening established company standards for information management.
- \* Incumbent vendors are responding by expanding the capabilities of their relational database management systems (RDBMSs) into multimodel territory, but hedging their bets by also adding specialized products to their portfolios, *creating both multimodel and multiproduct choices — and confusion in the minds of buyers.*

# Gartner, Critical Capabilities for Operational Database Management Systems, Dec 9<sup>th</sup> 2015

## Recommendations

- \* *Identify the use cases you have under consideration, and assess your current operational DBMS products' fit, costs, deployment options and skills requirements.*
- \* *Evaluate potential alternative product selections based on best-fit engineering that maps to your use cases — don't pay for features you don't and will not need. Use this Critical Capabilities document alongside its companion Magic Quadrant to identify candidates.*
- \* *Issue RFPs (using Gartner's RFP Toolkit templates) and conduct proof of concept (POC) exercises to confirm and validate candidates.*
- \* *Update your standards, training and support organizations to manage, qualify and accommodate additional requirements if your technology portfolio is expanding.*

# Gartner, Critical Capabilities for Operational Database Management Systems, Dec 9<sup>th</sup> 2015

## Strategic Planning Assumptions

- \* By 2017, all leading operational DBMSs *will offer multiple data models*, relational and NoSQL, in a single DBMS platform.
- \* Through 2018, 50% of small operational DBMS vendors will *disappear* due to acquisitions, mergers or business failures.
- \* By 2017, the "NoSQL" *label will cease* to distinguish DBMSs, reducing its value and resulting in the label falling out of use.

# The Big Data...

- \* Brings even more than just the need for more new data models...

# What is Big Data?

- \* There is *no size* that makes a data to be "Big Data", it always depends on the capabilities
- \* The data is "**Big**" when traditional processing with traditional tools is not possible due to the amount or the complexity of the data
  - \* You cannot open an attachment in email
  - \* You cannot edit a photo
  - \* etc.



# The three V's

- \* **Volume**, the size/scale of the data
- \* **Velocity**, the speed of change, analysis of streaming data
- \* **Variety**, different formats of data sources, different forms of data; structured, semi-structured, unstructured

# The other V's

- \* **Veracity**, the uncertainty of the data, the data is worthless or harmful if it's not accurate
- \* **Viability**, validate that hypothesis before taking further action (and, in the process of determining the viability of a variable, we can expand our view to determine other variables)
- \* **Value**, the potential value
- \* **Variability**, refers to data whose meaning is constantly changing, in consistency of data; for example words and context
- \* **Visualization**, a way of presenting the data in a manner that's readable and accessible

# Volume and Velocity

- \* Are easy to understand and there are plenty of technical solutions to solve the problems but...

# Variety leads to...

- \* -> diversity, complexity, ...
- \* Users can not query all of their data using a *single* high level declarative query language. They have to
  - \* use different query languages for querying different data
  - \* implement their own join algorithms to join between relational data and FSD
- \* Also many specialized systems lack *essential* advanced functionalities, such as ACID and fine grain security, that are the “norm” in RDBMSs

# Data Exploration

- \* Data exploration is to efficiently extract *knowledge* from data, even though you do *not* know what you are looking for.
- \* *Exploratory computing*: a conversation between a user and a computer.
  - \* The system provides active support
  - \* The exploration process is investigation, exploration-seeking, comparison-making, and learning
  - \* Wide range of different techniques is needed (the use of statistics, data analysis, query suggestion, advanced visualization tools, etc.)

# Not a new thing...

- \* J. W. Tukey. Exploratory data analysis. Addison-Wesley, Reading, MA, **1977**:
  - \* “with *exploratory data analysis* the researcher explores the data in many possible ways, including the use of graphical tools like boxplots or histograms, gaining knowledge from the way data are displayed”

# Old problems and new problems

- \* More and more data (volume)
- \* Different data models and formats (variety)
- \* Loading in progress while data exploration going on (velocity)
- \* Not all data is reliable (veracity)
- \* We do not know what we are looking for (value, viability, variability)
- \* Must support also non-technical users (journalists, investors, politicians,...) (visualization)
- \* All must be done *efficiently and fast*



# Different Layers

- \* the user interaction/user interface
- \* the middleware
- \* the database layer/engine

# Challenges for the User Interaction

- \* The user interface should help the user to find information she/he was *not* aware of and to find even *more* information based on the information already found.
- \* The data is in different formats, it is a large amount of different kinds of data and analyzing it takes time.
- \* The conversation must be fluently responsive.
- \* Not all users are IT professionals
- \* The query result visualization to understand the data and to communicate with the exploration software would be valuable

# User Interaction, Query Visualization

- \* The role of visualization for data analytics is huge (a picture tells more than 1000 words)
- \* visualization tools that assist users in navigating the underlying *data structures*
- \* tools that incorporate new types of interactions such as *collaborative annotations and searches*
- \* Query *result reduction* (aggregation, sampling and/or filtering operations)
- \* Query *result visualization*

# User Interaction, User Involvement

- \* the relevance of an answer for a specific user
  - \* predict users' *interests* and *anticipations* in order to issue the most relevant answer
- \* Some possible techniques
  - \* *Grouping* the users (subgroups)
  - \* *classify* different notions of relevance
  - \* devise strategies to *customize* the system response to the user behavior

# Middleware

- \* Building various optimizations on top of the database layer/engine
  - \* Can be used to improve the efficiency of the data exploration *without* changing the underlying architecture
- \* The exploration tasks are computationally heavy and can be speedup in the middleware layer with
  - \* *Data Prefetching* (and background execution)
  - \* *Query Approximation*

# Middleware, Data Prefetching

- \* *caching* data sets which are likely to be used
- \* the *trade-off* between introducing new results and re-using cached ones
- \* the main *challenge* is identifying the data set with the highest utility

# Middleware, Query Approximation

- \* The system offers *approximate* answers
- \* Allows users to get a quick sense of whether a particular query reveals anything interesting about the data
- \* Need solutions that process queries on *sampled* data sets to provide fast query response times
- \* the *trade-off* between results accuracy and query performance



# Database Engine/Layer

- \* A database is all about *saving* and *retrieving* data
- \* The way the data is *stored* defines the best possible ways of accessing it, the way data is *accessed* defines the best way to store it...
  - \* We might not know what data we will be loading
  - \* We do not know what we are looking for
    - > therefore are unable to define the "best" way of storing
- \* how can the architecture of database systems be redesigned to aid data exploration tasks?

# Saving

- \* Efficiently, Reliably
- \* Understanding the data while saving
  - \* finding the PK to be able to join to other data
  - \* Taxonomies to understand that first name = fname?
  - \* Metadata Management
  - \* ...
- \* Categorizing
  - \* Some data is more valuable than other
  - \* some data is more reliable than other
  - \* not all the data want to be stored in the same, operational database
  - \* data classification (reliable/no reliable)

# Saving

- \* The data need to be *classified* before storing and the qualification might affect where and on which format the data is stored.
  - \* Maybe transforming?
- \* Usually the disk space in RDBMS is expensive
  - \* -> new ways to store the data (Hadoop,...)
- \* What data to save close to what and why
- \* today's spam is tomorrow's ham (possibility to re-categorize)
- \* How to handle date datatype?
- \* ...

# Oracle Database

- \* Quite often the solution for saving a new data model is CLOB, BLOB or external table
- \* But we need more than that...

# Database Engine/Layer, Solutions

1. Adaptive Indexing
2. Adaptive Loading
3. Adaptive Storage
4. Flexible Architecture
5. Architectures tailored for approximation processing

# Adaptive Indexing

- \* creating indexes *incrementally* and *adaptively* during query processing based on the columns, tables and value ranges that queries request
- \* Indexes are built gradually; as more queries arrive indexes are continuously fine-tuned

# Adaptive Loading

- \* Not all data is *needed*; users might want to leave some parts of the data unloaded
- \* Not all data is *available*; users might start querying a database system before all data is loaded

# Adaptive Storage

- \* There is no one perfect solution for storage layout for all the data.
- \* Adaptive Storage can be used for different needs of storage layouts
- \* Examples
  - \* OctopusDB (a central log and Storage Views)
  - \* H<sub>2</sub>O (supports multiple storage layouts and data access patterns, decides during query processing, which design is best for classes of queries)
  - \* In-memory columnar stores (for example Oracle)
  - \* real-time materialized views (Oracle 12c rel 2)



# Flexible Architecture

- \* can tune the architecture to the task at hand
  - \* by having a declarative interface for the physical data layouts
  - \* by having a declarative interface for the whole engine
  - \* using organic databases to continuously match incoming data and queries
    - \* Start with a schema that is "good-enough", schema evolution while loading and querying

# Architectures tailored for approximation processing

- \* An architecture where storage and access patterns are tailored to efficiently support sampling-based query processing
  - \* efficient access and updates of sampled data sets
  - \* In the DB core
  - \* For example Oracle Database 12c: Adaptive Dynamic Sampling (for the optimizer)
  - \* DICE (Distributed and Interactive Cube Exploration) faceted exploration of data cubes; combines sampling with *speculative execution* in the same engine
- \* Design new engines is one of the open challenges.

# Techniques for Large Data Size

- \* *Summarizing*: fully pre-computing is not possible and not smart
  - \* The size of data summarized must be large enough to estimate statistical parameters and distributions, but manageable from the computation viewpoint
- \* *Data mining techniques*
- \* *Histograms*, presenting a histogram algebra for efficiently executing queries on the histograms instead of on the data, and estimating the quality of the approximate answers
- \* Effective *pruning techniques*, not all node-paths are interesting from the user viewpoint, and the ones that fail to satisfy this requirement should be discarded as soon as possible.

# Different Data Models

- \* Two possibilities:
  - \* Polyglot persistence
  - \* Multimodel database

# Polyglot Persistence

- \* Different kinds of *data models* are best dealt with *different data stores*
- \* Data is stored using multiple data storage technologies to support better the usage of the data (applications, components)

# Oracle and Polyglot Persistence

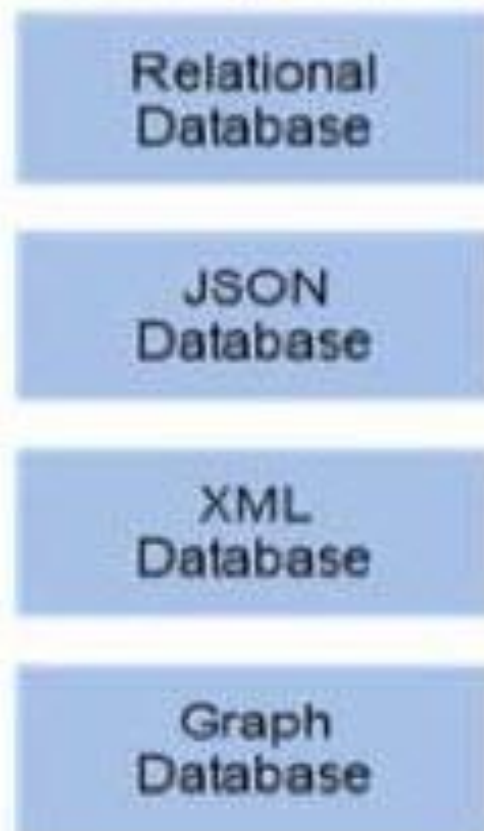
- \* Even using Oracle technology you can build a polyglot solution:
  - \* Berkeley DB as a Key-Value store
  - \* Oracle NoSQL Database as a Key-Value and sharded database
  - \* OracleTimesTen as an In-Memory Database
  - \* Essbase for analytic processing
  - \* ...

# What's wrong with polyglot

- \* Different query languages
- \* Different frontends/user interfaces
- \* How to join data?
- \* Is data consistent?
- \* Is data secured?
- \* Many different skills needed
- \* ...

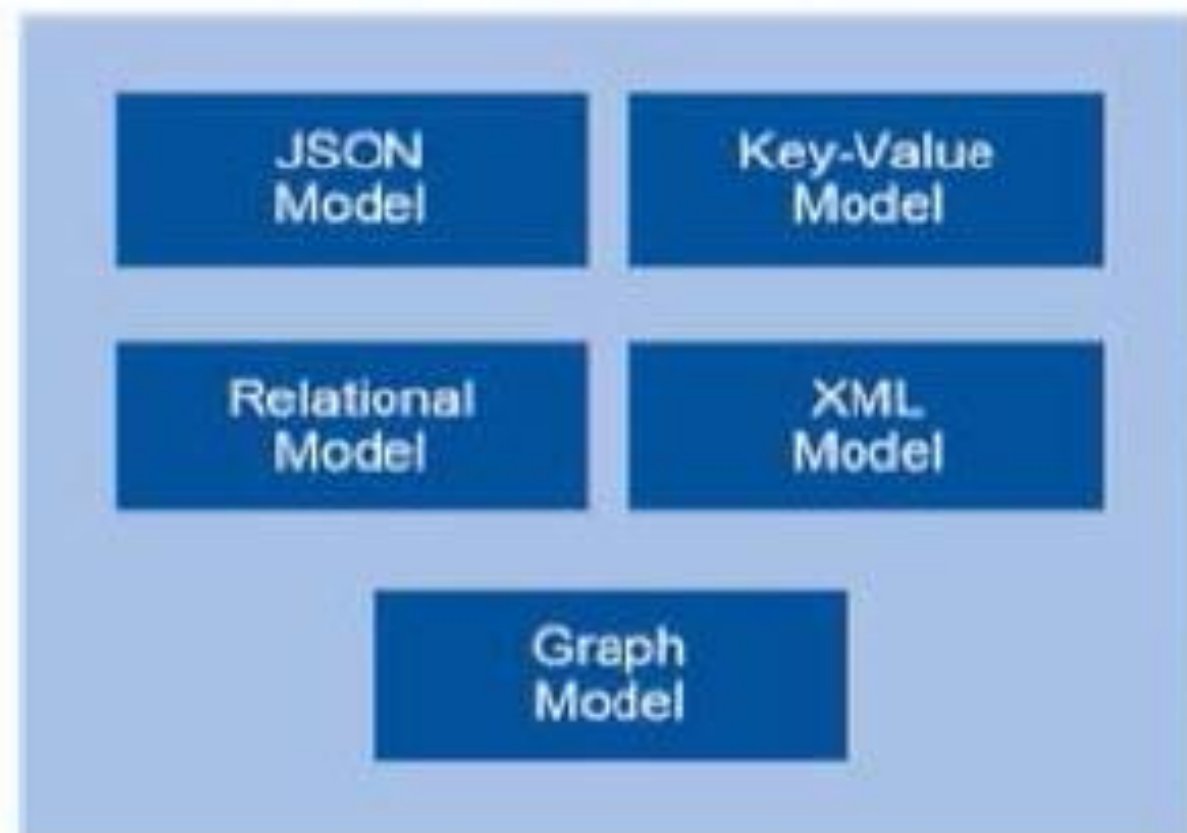
# Polyglot vs. Multimodel

## Polyglot Multiple Databases With Single Model



JSON = JavaScript Object Notation

## Multimodel Single Database With Multiple Models



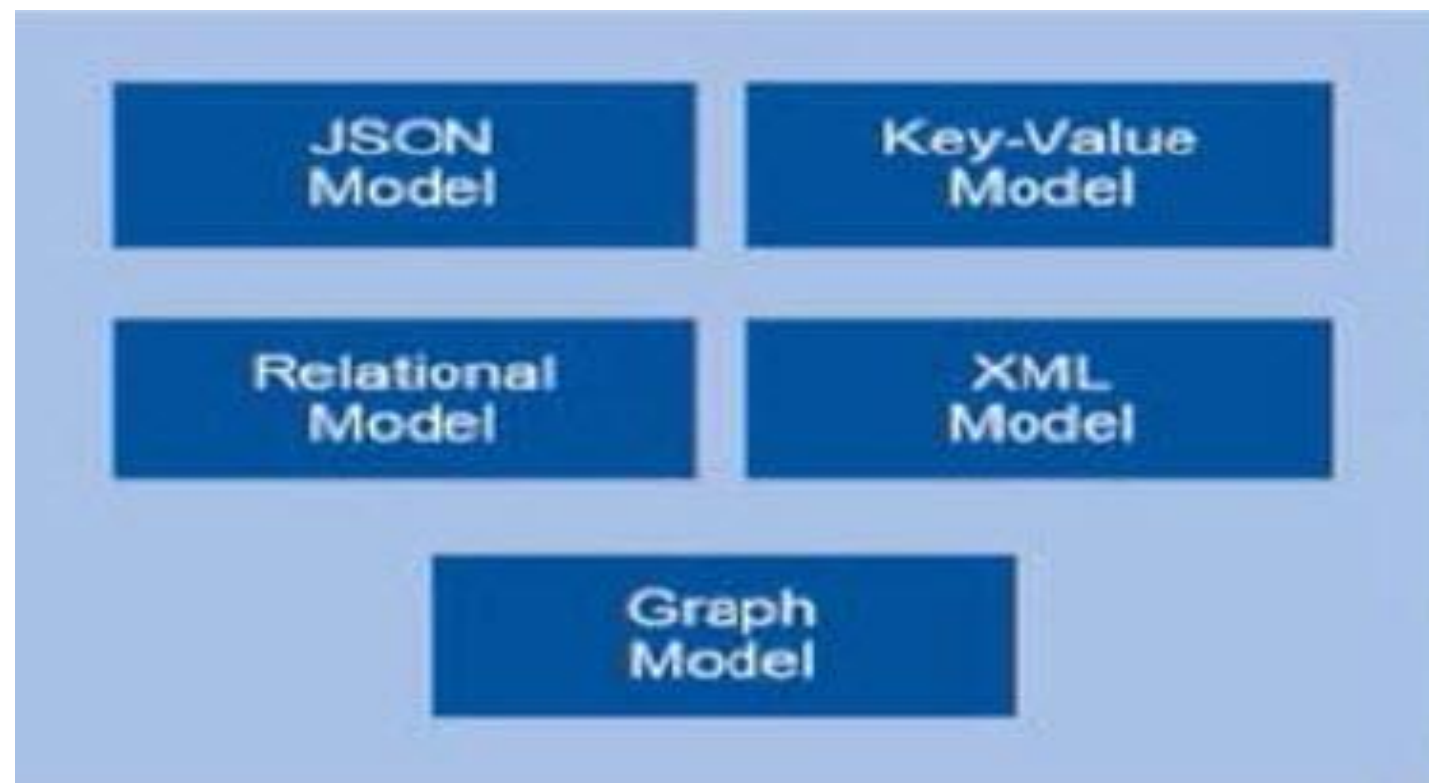
Source: The Rise of Polyglot Persistence Demands Your Consideration, Gartner (January, 2015)



# Multimodel database

- \* A multimodel database is designed to support multiple data models against a single, integrated backend
- \* One query language
- \* All data available
- \* ACID (or any level needed), security, backup/recovery, ...
- \* All the "good" from relational added with all the "good" from everywhere else

# How to save and how to retrieve the data?



# Interesting solutions in Oracle Database

# JSON DataGuide in Oracle Database 12cR2

- \* save as it is, use it with SQL
  - \* Now only JSON, something more in the future?
- \* “write without schema, read with schema”

# JSON DataGuide in Oracle Database 12cR2

- \* Enabling FSDM (Flexible Schema Data Management) in a RDBMS
- \* Storing JSON objects as aggregated, self-described entities without shredding them into relational rows and columns
- \* Indexing JSON using a schema-agnostic strategy to support ad-hoc queries that search **both** schema and values together
- \* Querying JSON using SQL as the inter-document query language and SQL/JSON path language as the intra-document query language

# Creating a DataGuide

- \* Two forms of JSON DataGuide
  - \* Persistent
    - \* a component of a schema agnostic JSON search index
  - \* Transient
    - \* a new SQL aggregate function over any JSON collections that can be computed from SQL declaratively

# Creating a DataGuide, Persistent

- \* DataGuide information is part of the JSON search index infrastructure
- \* CREATE SEARCH INDEX FOR JSON
  - \* Can be created for index only, DataGuide only, both
  - \* The default is both
  - \* database privilege CTXAPP needed
- \* The content of DataGuide is automatically updated whenever the index is synchronized.
- \* To obtain the DataGuide information stored in a JSON search index use PL/SQL function `DBMS_JSON.get_index_dataguide`

# JSON search index and Data Guide

- \* When the index is synced, additive *updates* to the document set are automatically reflected in the persisted DataGuide information
- \* *Deletions* are **not** updated
  - \* you must create the index again if the data of deletion is needed
- \* can also include statistics (how frequently each JSON field is used in the document set)
  - \* Explicitly gathered
  - \* Not updated automatically



# Creating a DataGuide, Transient

- \* DataGuide is created by scanning JSON documents (use SQL aggregate function `json_dataguide`)
- \* Not updated automatically

# Formats of DataGuide

- \* There are two formats for a DataGuide (both defined as CLOB):
  - \* Flat
    - \* to query DataGuide information such as field frequencies and types
  - \* Hierarchical
    - \* to create a view, or to add virtual columns, using particular fields that you choose on the basis of DataGuide information
- \* (DBMS.JSON.FORMAT\_FLAT or DBMS.JSON.FORMAT\_HIERARCHICAL)

# JSON DataGuide, views

- \* Relational *views* on top of a set of JSON files for querying the data as it were relational
  - \* still only saved once and as a JSON document
- \* You can create multiple views based on the same JSON document set, projecting different fields

# Creating views over JSON data

- \* Based on a Hierarchical DataGuide
  - \* The fields projected are those in the *DataGuide*
  - \* You can edit the DataGuide to include only the fields that you want to project.
  - \* DBMS\_JSON.create\_view
- \* Based on a Path Expression
  - \* You can use the information in a DataGuide enabled JSON search index to create a database view
  - \* Columns project JSON fields from your *documents*
  - \* The fields projected are the scalar fields and the scalar fields in the data targeted by a specified SQL/JSON path expression.

# DataGuide and a Virtual Column

- \* Based on DataGuide information for a JSON column, you can project scalar fields as virtual columns to the same table that contains this JSON column
  - \* From
    - \* hierarchical DataGuide or
    - \* DataGuide enabled JSON search index
  - \* `DBMS_JSON.add_virtual_columns`,  
`DBMS_JSON.drop_virtual_columns`
- \* To improve the performance you can
  - \* Build an index on it
  - \* Gather statistics on it for the optimizer
  - \* Load it into the In-Memory Column Store

# Why DataGuide is interesting?

- \* It enables FSDM (Flexible Schema Data Management) in a RDBMS
- \* And as a solution might be used in other cases for different semi-structured data models in the future...

# In-Memory

- \* Save as it is, copy to a faster data model kept in-memory
  - \* Now only columnar store

# In-Memory

- \* Very interesting for retrieving data...
- \* Saving data as its original data model and showing a copy of it on an in-memory version of another data model when needed
- \* Original data as it is and a copy of it created in another data model when needed
  - \* Fast (relative term 😊 )
  - \* Synchronized automatically
  - \* No additional storage needed
  - \* The version of a data model is used that works best for the need
- \* Maybe other data models could be stored in-memory too?  
Graph?



# Conclusion

- \* So many data models
  - \* The network data model
  - \* The hierarchical data model
  - \* The relational data model
  - \* Objects
  - \* XML
  - \* Spatial
  - \* NoSQL models
  - \* ...
- \* Flexible Schema Data (FSD): "data first, schema later/never"

# Conclusion

- \* Several V's related to Big Data...
  - \* Volume
  - \* Velocity
  - \* Variety
  - \* Veracity
  - \* Viability
  - \* Value
  - \* Variability
  - \* Visualization
  - \* ...

# Conclusion

- \* Data exploration is about finding new information and *knowledge* from the data *without* knowing what you are looking for. And it brings a lot of pressure for the performance...in all levels:
  - \* the user interaction/user interface
  - \* the middleware
  - \* the database layer/engine

# Conclusion

- \* Various data models, solutions
  - \* Polyglot
  - \* Multimodel database
- \* Promising solutions in Oracle Database
  - \* JSON DataGuide in Oracle Database 12cR2
  - \* In-Memory
- \* Plenty of research going on in this area and results needed soon...

# THANK YOU!

QUESTIONS?

Heli:

Email: [heli@miracleoy.fi](mailto:heli@miracleoy.fi)

Twitter: @HeliFromFinland

Blog: [Helifromfinland.com](http://Helifromfinland.com)

Elise:

Email: [elise.valin-raki@fennia.fi](mailto:elise.valin-raki@fennia.fi)

Twitter: @ValinRaki

Blog: [evrellai.wordpress.com](http://evrellai.wordpress.com)