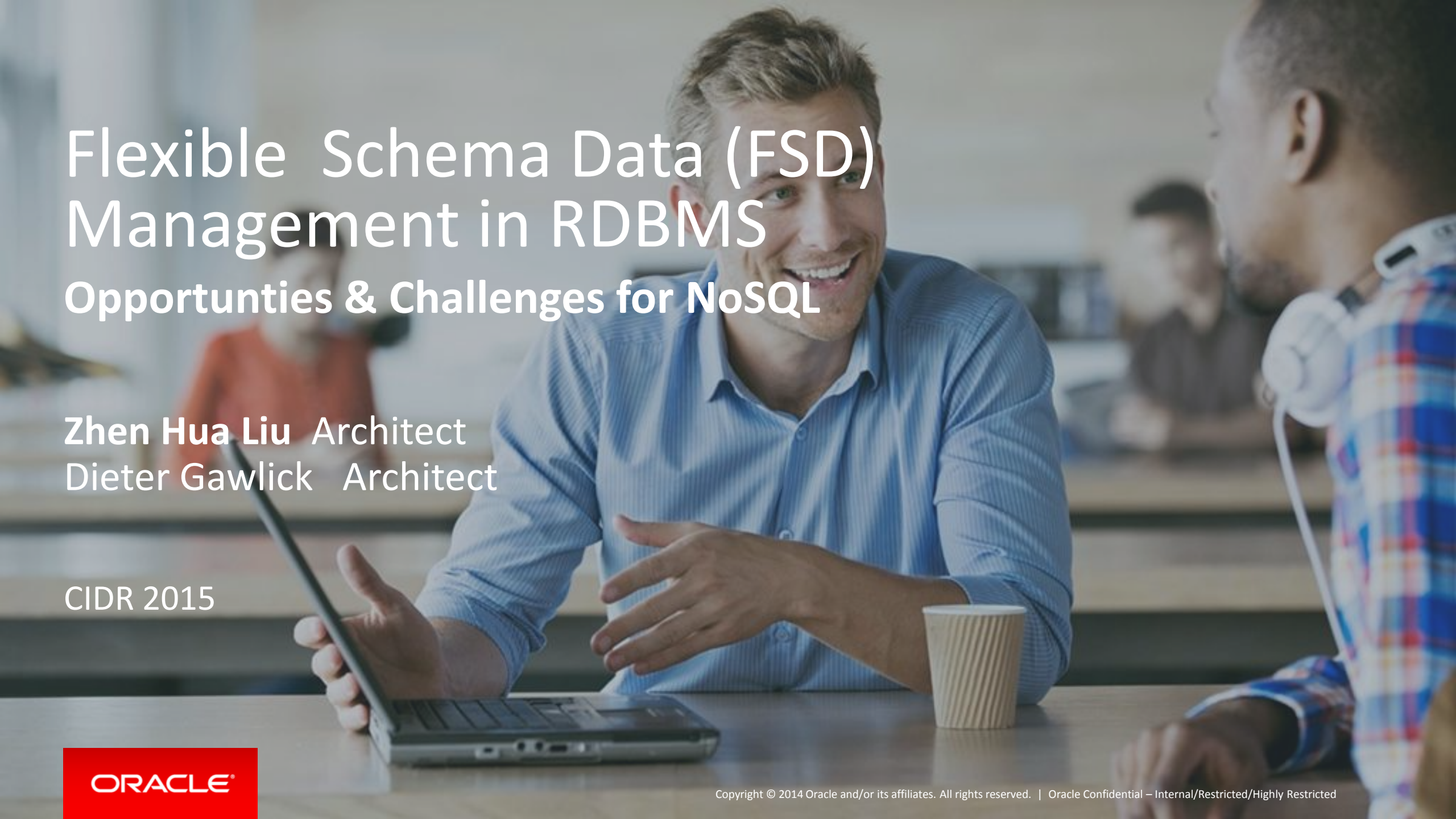

ORACLE®



Flexible Schema Data (FSD) Management in RDBMS

Opportunties & Challenges for NoSQL

Zhen Hua Liu Architect
Dieter Gawlick Architect

CIDR 2015

ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

1. **DBMS Generational Gap: Need Support of Flexible Schema Data**
2. Requirements & Challenges of FSD management
3. Engineering Practices&Principles of FSD management
4. Each FSD Principle with Opportunities & Challenges Analysis
5. Related Work & Conclusion
6. Q & A

DBMS Generational Gap

Upfront Schema Design becomes bottleneck!

- My Father's RDBMS: Classical Bank Record Keep Application
  - Process Statically Shaped Data
  - DBA Style: Schema First, Data Later
  - Every New Shaped Data demands Schema Evolution
- My Generation's DBMS: Flexible Schema Data Management
  - Process Semi-Structured/Unstructured Dynamically Shaped Data: Web Data, Diversity Data
  - Agile Style: Data First, Schema Later/Never
  - **On Write: No Schema for Store; On Read: Soft Schema for Query**

Simple & Popular FSD Example - JSON

```
{
    "firstName": "John“,
    "lastName" : "Smith“,
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street“,
        "city": "New York“,
        "state": "NY“,
        "postalCode": "10021“,
        "isBusiness" : false
    },
    "phoneNumbers": [
        {"type": "home“, "number": "212 555-1234“},
        {"type": "fax“, "number": "646 555-4567“ }
    ],
    "creditHistory": [
        {"year": 2011,"creditScore": 650}
    ],
    "bankruptcies " : null
}
```

- Each FSD instance is a set of key-value pairs organized hierarchically
- FSD collection: a set of FSD instances
- Schema is considered **unbounded** for the entire FSD collection
- Storing FSD using relational tables requires **Constant Schema Evolution**
- Storing FSD using vertical table requires many self-joins & suboptimal object retrieval time

FSD Management Requirements

Conventional Schema-based RDBMS Wisdoms are being Challenged!

- **Storage Requirement:** How to store data without upfront schema definition ?
- **Query Requirement:** How to query data without upfront schema definition ?
- **Indexing Requirement:** How to index data without upfront schema definition ?

NoSQL ? Does this imply SQL is dead ?

- **Consolidated Data Management Platform Requirement:** How to query both my data and my father's data together ?

Think Out of Box here means **THINK OUT OF SCHEMA !**

New Engineering Practises/Principles for managing FSD

My Data & My Father's Data are jointly queried together via NoSQL

- **Storage Principle**: Schema-less Storage
  - **Document Object Storage Model:** Not relying on schema to decompose & shred data.
  - **DataGuide**: dynamically computed SOFT Schema to support **schema on read** capability
- **Query Principle**: Declarative Query (Central Dogma of DBMS)
  - **NoSQL**: **N**aturally **o**pen SQL as **S**et **Q**uery **L**anguage with embedded FSD domain language
  - **Declarative FSD Domain Language**: for query both FSD schema & data together
- **Index Principle**: Schema-less Indexing
  - **Search Index** : Data First/Schema Never Indexing – Ad-hoc workload
  - **Table index**: Data First/Schema Later Indexing – known workload

My Principles are inspired from My Father's RDBMS Extensibility Ideas (UDT/UDF/UDI)

Program Agenda

1. DBMS Generational Gap : Need Support of Flexible Schema Data
2. Requirements & Challenges of FSD management in RDBMS
3. Engineering Practices & Principles of FSD management in RDBMS
4. **Each FSD Principle with Opportunities & Challenges Analysis**
5. Conclusion
6. Q & A

FSD Storage principle in RDBMS

Storage Principle – Document Object Store without shredding

- Each FSD instance is **self-contained** without relying on central schema definition
  - Schema & Data are stored together in each FSD instance
- Each domain specific FSD can be stored as varchar/varbinary/CLOB/BLOB **without a new SQL datatype**!
  - JSON as FSD Example: FSD Check Constraint/Soft Schema Validation
  - *CREATE TABLE PERSON_JTAB (jcol VARCHAR(32000) CONSTRAINT jcon CHECK (jcol IS JSON));*
- 100% operational completeness support for FSD
  - Transactions, Replication, Partition, Security, Temporial, Provenance, Export/Export, Fault Tolerance, Client APIs, …

FSD Query Principle in RDBMS

Query Principle: SQL: **S**et **Q**uery **L**anguage with FSD Domain Language

- Leveraging SQL as **Inter-Document S**et **Q**uery **L**anguage
- Leveraging FSD Domain Language as **Intra-Document** query language
- FSD Domain language for each FSD instance to do
  - path navigation, extracting scalar values & fragments, searching content, transforming and updating fragments. JSON as an example

```
SELECT JSON_VALUE(T.jcol, '$.person.name'),
       JSON_QUERY(T.jcol, '$.person.address')
FROM PERSON_JTAB T
WHERE JSON_EXISTS(T.jcol, '$.person.creditHistory?(score >=
700)') AND
JSON_TEXTCONTAINS(T.jcol, '$.person..experiences', 'semi-
structured data processing')
ORDER BY JSON_VALUE(T.jcol, '$.person.address.zip')
```

FSD Query Principle in RDBMS

JSON_TABLE() for Relational View Projection – UNNEST array

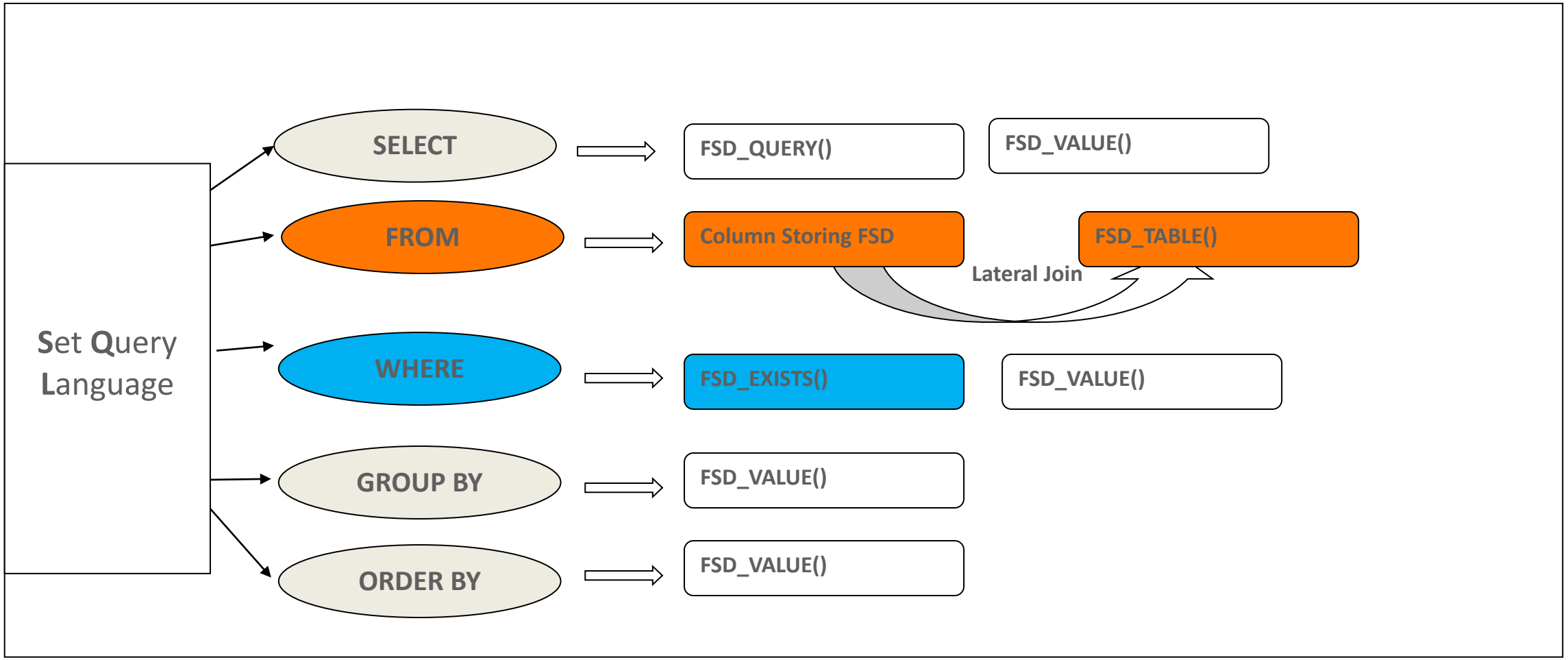
- Project FSD data as relational table
- Objects inside array become ROWS
- Values inside object become COLUMNS

PH_TYPE	PH_NUM
home	212 555-1234
fax	646 555-4567

```
SELECT jrv.*
FROM Person_Jtab, JSON_TABLE (jcol, '$.person.phoneNumbers[*]'
             COLUMNS
                    PH_TYPE   VARCHAR2(10) PATH '$.type',
                    PH_NUM    VARCHAR2(10) PATH '$.number') jrv;
```

FSD Query Principle in RDBMS

FSD UDFs to Set Query Language without a brand new query language!



FSD Index principle in RDBMS

Search Index: Data First/Schema Never Indexing

- Schema and data are indexed together for ad-hoc exploratory and search queries over FSD collection
- Generalized Inverted Index (FSD_EXISTS(), FSD_TEXTCONTAINS())
  - Classical SIGIR: Full Text Search for Document Content
  - **Beyond classical SIGIR**:
    - Range Value Search for leaf Scalar Value (auto detecting and indexing number, dates, timestamps)
    - Hierarchical Path Containment for both full text search & range value

```
SELECT JSON_VALUE(T.jcol, '$.person.name)
FROM PERSON_JTAB T
WHERE JSON_EXISTS(T.jcol, '$.person.creditHistory?(score >= 700)')
AND JSON_TEXTCONTAINS(T.jcol, '$.person..experiences', 'semi-
structured data processing')
```

FSD Index principle in RDBMS

Table Index: Data First/Schema Later as Indexing

- Table Index based on Dataguide & known query workload
- FSD_VALUE() Scalar Value Functional Index
- FSD_TABLE() Table Index/Materialized view
- Table Index brings relational model back into FSD as secondary view & indexing structures instead of primary storage structures
  - Provide maximum flexibility because table index is secondary structure that can be dropped and recreated without impacting primary storage structures
  - No schema evolution & management Issue
  - Enables Paradigm of **Data First, Schema Later as table Index**

FSD Data Model Challenges

What's wrong with a single tree model ?

- **Single Hierarchy issue** for document storage Model
  - Document Storage Model imposes single hierarchy restriction whereas relational Model provides flexible hierarchies access
- Your father can do multi-hierarchical access of relational data & you are re-inventing IMS !
- Need DataGuide to E/R Model EcoSystem ?
  - Given students taking courses hierarchy, it shall infer courses being taken by students hierarchy
- Need Declarative Hierarchy Transformation Language for FSD
  - Leverage Category Theory Providing Hierarchy Equivalency Transformation & Query Access ?

FSD Layout Challenges

Row & Columnar Dual FSD Data Format

- **FSD Instance layout** for both efficient path query and piece-wise update (Avro, BSON, Tree encoding)
- **FSD Set Columnar layout of FSD** for Efficient Vector based Set processing query over FSD collection (Parquet, Dremel)
- Which is Better Storage Model ?
- Leveraging idea of InSitu Query Processing for FSD
  - **Exploit instance/set query friendly Data Layout as secondary Just-In-Time In-Memory structures!**
  - **Never Getting Stale with a storage model**

FSD Layout Challenges

Keeping Dual Formats In-Synch in Real Time

- Tension between dual formats:
  - Ingestion/update friendly instance oriented format
    - Optimal for OLTP workload
    - Not so good for query
  - Search & Analytic query friendly set oriented format:
    - columnar & inverted index favors compression & Batch loading, good for OLAP query
    - Not optimal for OLTP workload
- Keeping dual formats transactionally consistent – LSF/LSM/MVCC

Program Agenda

1. DBMS Generational Gap : Need Flexible Schema Data
2. Requirements & Challenges of FSD management in RDBMS
3. Engineering Practises&Principles of FSD management in RDBMS
4. Each FSD Principle with Opportunities & Challenges Analysis
5. **Related Work & Conclusion**
6. Q & A

Related Work

Common Theme: Break strong dependency of using schema to store data

- Research:
  - Stanford LOREL Semi-Structured DB
  - XMLDB
  - Argo/SQL, NoBench benchmark from University of Wisconsin
  - SiNew/TeraData from Yale University Research
- Industry:
  - DB2, Oracle, Microsoft, TeraData SQL/JSON, SQL/XML Support
  - No-SQL DB Products: MongoDB, MarkLogic

Conclusion

Take Home Message for FSD: What would happen if I do not depend on schema to store data ?

- Engineering Practises & Principles for FSD
  - **Storage** – Native Store without shredding, Just-in-time secondary structures for both instance & set friendly access patterns
  - **Query** – SQL As Set Query Language with FSD domain Language embedded
  - **Index** – Search Index and Table index
- The underlying philosophy is very simple:
  - **Treat Schema as if it were data:** Store, Index and Query schema along with the data
- ***My Data & My Father's Data can be queried together by Naturally openning Set Query Language (NoSQL) !***

Questions

Hardware and Software
Engineered to Work Together