Querying NoSQL with SQL

HAVING Your JSON Cake and SELECTing it too

Matthew D. Groves, @mgroves

TITANIUM SPONSORS

epiq

PAIGE TECHNOLOGIES
INTELLIGENT PAIRING. PERPETUAL SUCCESS.

ADAPTIVE™ SOLUTIONS GROUP

okta

Platinum Sponsors

VinSolutions
Make every connection count.

jack henry & ASSOCIATES INC.®

VALOREM

NAIC
National Association of Insurance Commissioners

NIPR
NATIONAL INSURANCE PRODUCER REGISTRY

TEKsystems
Our people make IT possible.

KEYHOLE SOFTWARE

smartsheet
Work Better™

Couchbase

EquipmentShare

KOCH™

PLURALSIGHT

Google Cloud

VML

mylo™ POWERED BY LOCKTON

Gold Sponsors

Advantage Tech IT Staffing & Recruiting Services

PeopleAdmin

CENTRIQ TRAINING

Cerner

KU EDWARDS CAMPUS
The University of Kansas

twilio

OBJECT PARTNERS

{Track:js}
JAVASCRIPT ERROR MONITORING

service management group®

stackify

Balance Innovations

Allied Global SERVICES

GARMIN.

dsi.

LRS™ CONSULTING SERVICES

Aviture
ingenuity. engineered.

TouchNet + Heartland

Mediware®

An affiliate of C&A Industries, Inc.
AUREUS GROUP
Finance | Systems | Executive

Children's Mercy KANSAS CITY

Commerce Bank

SECURITY PS
STRATEGIC INFORMATION SECURITY

ProKarma

Agenda

- Why NoSQL?
- JSON Data Modeling
- SQL for JSON
- Ecosystem and convergence

Where am I?

- KCDC (Kansas City Developer Conference)
- http://kcdc.info

Who am I?

- Matthew D. Groves
- Developer Advocate for Couchbase
- @mgroves on Twitter
- Podcast and blog: http://crosscuttingconcerns.com
- “I am not an expert, but I am an enthusiast.”
  – Alan Stevens

Couchbase
DEVELOPER COMMUNITY

Querying NoSQL with SQL

HAVING Your JSON Cake and SELECTing it too

Matthew D. Groves, @mgroves

Major Enterprises Across Industries are Adopting NoSQL

Technology	Retail & Apparel	Communications	E-Commerce & Digital Advertising
cisco salesforce	Walmart TESCO	at&t verizon	ebay LinkedIn
Finance & Business Services	**Travel & Hospitality**	**Games & Gaming**	**Media & Entertainment**
VISA PayPal	Marriott ORBITZ	EA BLIZZARD ENTERTAINMENT	THOMSON REUTERS Mc Graw Hill

Why NoSQL?

NoSQL Landscape

Document
- Couchbase
- MongoDB
- DynamoDB
- DocumentDB

Key-Value
- Couchbase
- Riak
- BerkeleyDB
- Redis

Graph
- OrientDB
- Neo4J
- DEX
- GraphBase

Wide Column
- Hbase
- Cassandra
- Hypertable

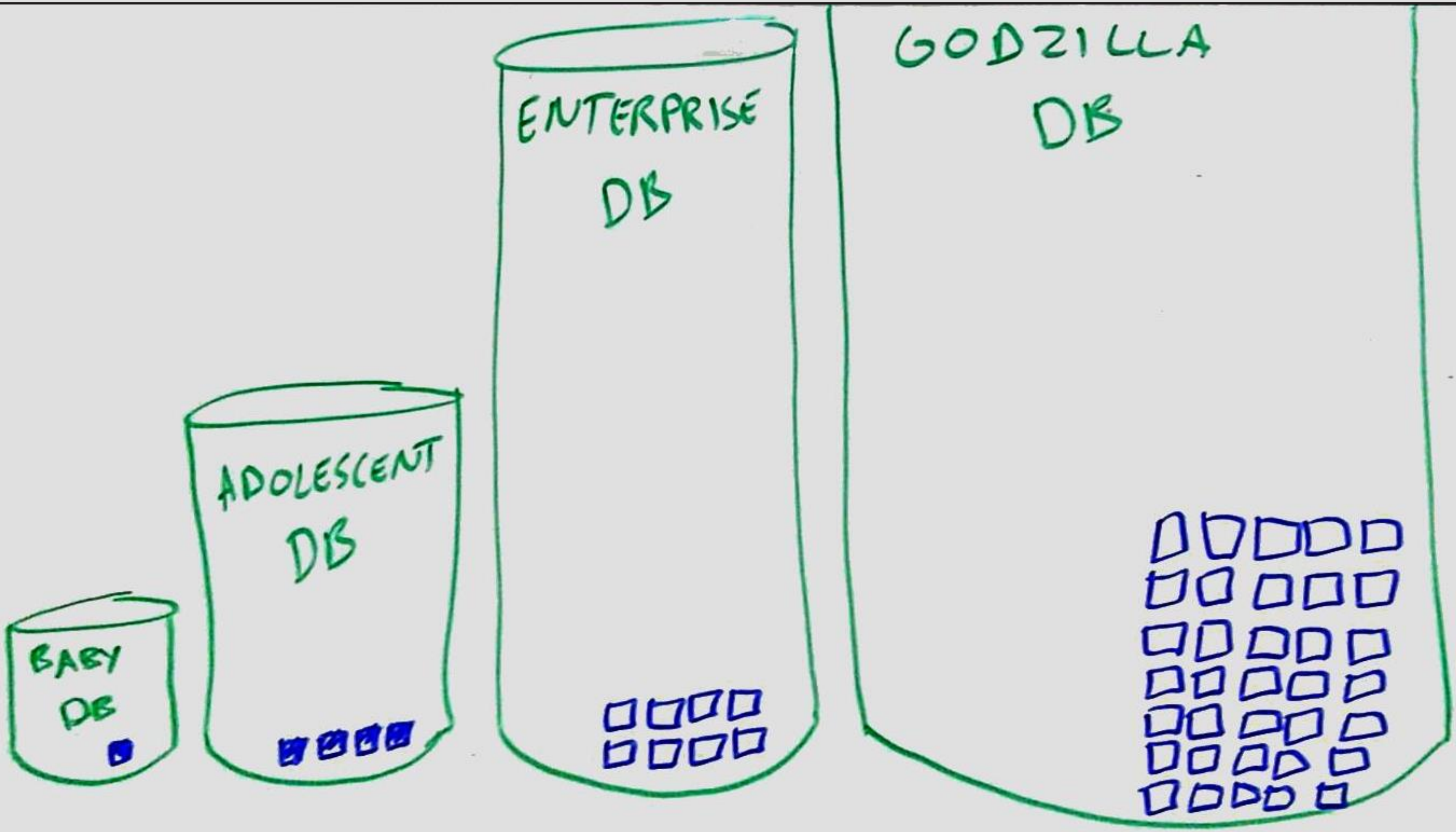
NoSQL Landscape

Document

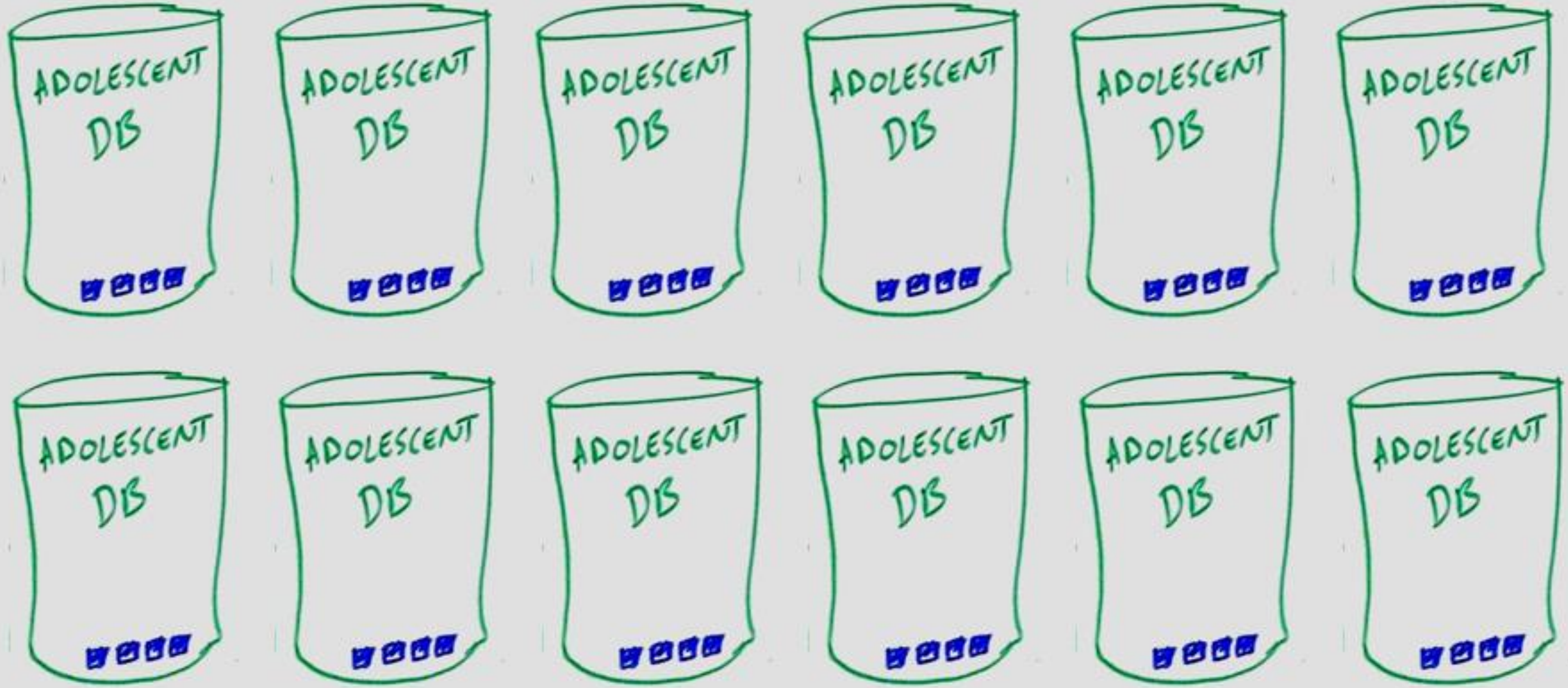
- Couchbase
- MongoDB
- DynamoDB
- DocumentDB

- Get by key(s)
- Set by key(s)
- Replace by key(s)
- Delete by key(s)
- Map/Reduce

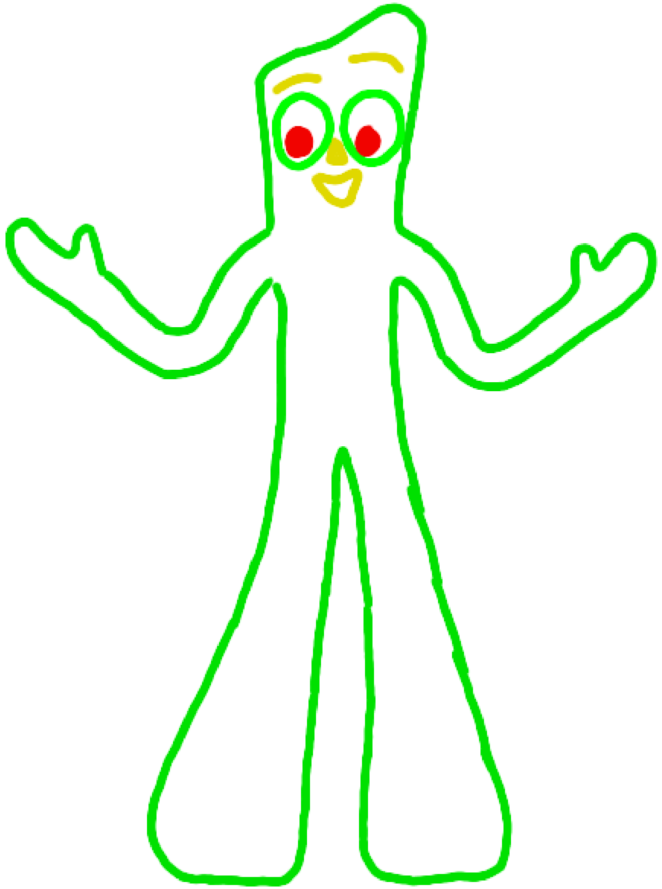
Why NoSQL? Scalability



Why NoSQL? Scalability



Why NoSQL? Flexibility



Why NoSQL? Performance

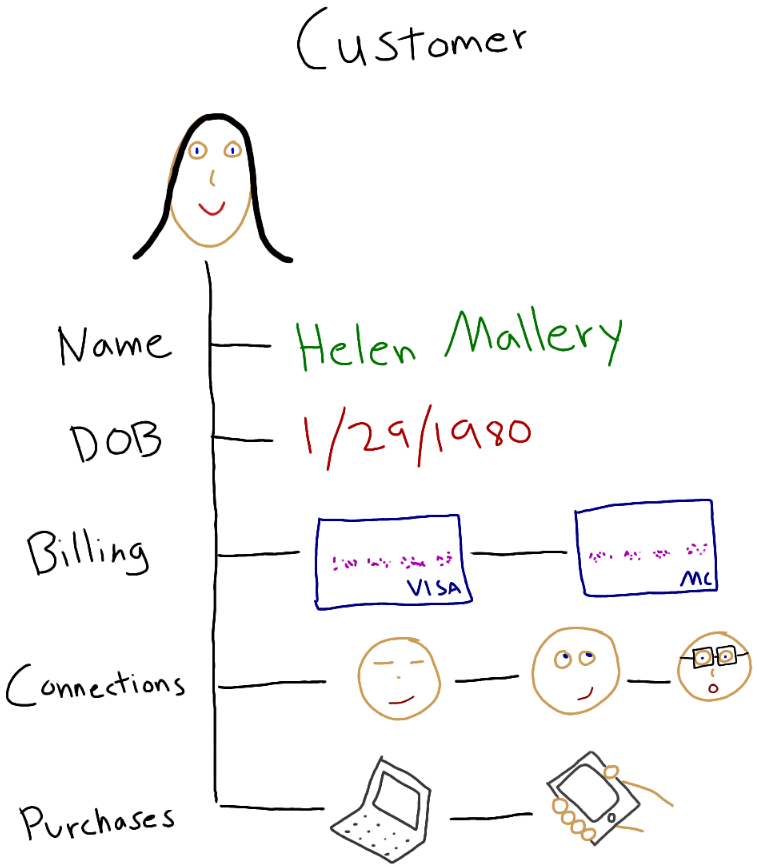
Why NoSQL? Availability

JSON Data Modeling

Models for Representing Data

Data Concern	Relational Model	JSON Document Model
Rich Structure		
Relationships		
Value Evolution		
Structure Evolution		

Properties of Real-World Data



Modeling Data in Relational World

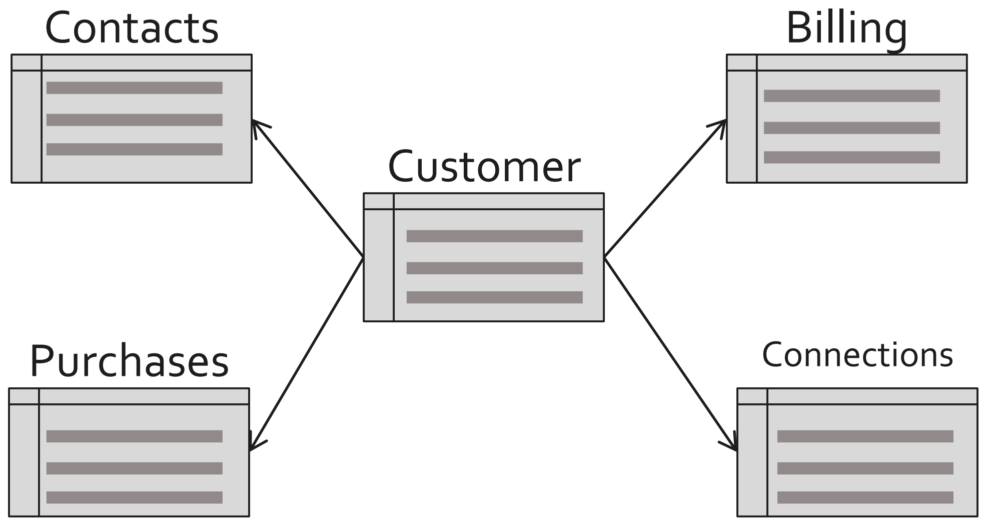


Table: Customer

CustomerID	Name	DOB
CBL2015	Jane Smith	1990-01-30

Customer DocumentKey: CBL2015

```
{
   "Name" : "Jane Smith",
   "DOB"  : "1990-01-30”
}
```

Table: Customer

CustomerID	Name	DOB
CBL2015	Jane Smith	1990-01-30

Table: Purchases

CustomerID	Item	Amount	Date
CBL2015	laptop	1499.99	2019-03

Customer DocumentKey: CBL2015

```
{
  "Name" : "Jane Smith",
  "DOB"  : "1990-01-30",
  "Purchases" : [
    {
      "item"   : "laptop",
      "amount" : 1499.99,
      "date"  : "2019-03",
    }
  ]
}
```

Table: Customer

CustomerID	Name	DOB
CBL2015	Jane Smith	1990-01-30

Table: Purchases

CustomerID	Item	Amount	Date
CBL2015	laptop	1499.99	2019-03
CBL2015	phone	99.99	2018-12

Customer DocumentKey: CBL2015

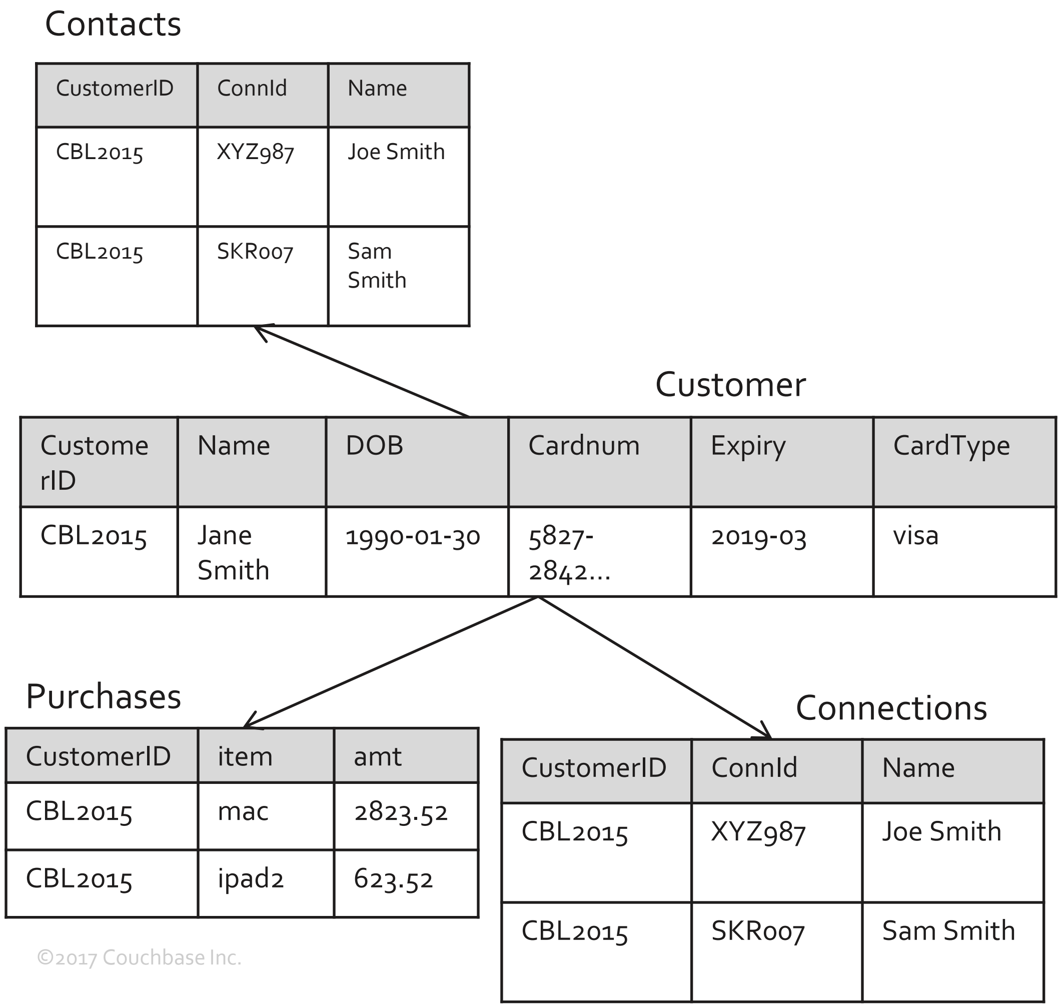
```
{
   "Name" : "Jane Smith",
   "DOB"  : "1990-01-30",
   "Purchases" : [
      {
         "item"    : "laptop",
         "amount" : 1499.99,
         "date"  : "2019-03",
      },
      {
         "item"    : "phone",
         "amount" : 99.99,
         "date"  : "2018-12"
      }
   ]
}
```

Table: Connections

CustomerID	ConnId	Name
CBL2015	XYZ987	Joe Smith
CBL2015	SKR007	Sam Smith

Customer DocumentKey: CBL2015

```
{
   "Name" : "Jane Smith",
   "DOB"  : "1990-01-30",
   "Billing" : [
      {
         "type"    : "visa",
         "cardnum" : "5827-2842-...",
         "expiry"  : "2019-03"
      }, ...
   ],
   "Connections" : [
      {
         "ConnId"   : "XYZ987",
         "Name"     : "Joe Smith"
      },
      {
         "ConnId"   : ”SKR007",
         "Name"     : ”Sam Smith"
      }
}
```



Contacts

CustomerID	ConnId	Name
CBL2015	XYZ987	Joe Smith
CBL2015	SKR007	Sam Smith

Customer

CustomerID	Name	DOB	Cardnum	Expiry	CardType
CBL2015	Jane Smith	1990-01-30	5827-2842…	2019-03	visa

Purchases

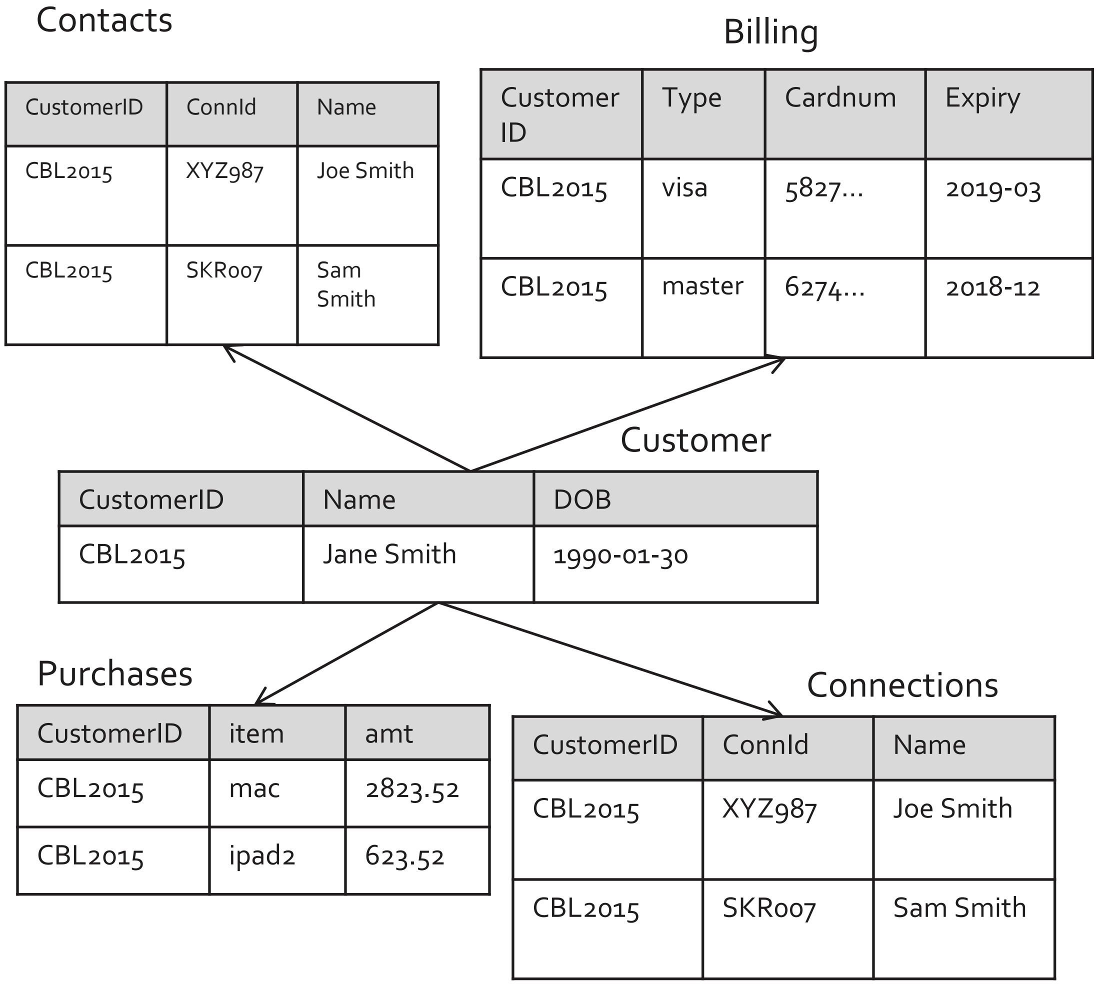
CustomerID	item	amt
CBL2015	mac	2823.52
CBL2015	ipad2	623.52

Connections

CustomerID	ConnId	Name
CBL2015	XYZ987	Joe Smith
CBL2015	SKR007	Sam Smith

DocumentKey: **CBL2015**

```
{
  "Name" : "Jane Smith",
  "DOB"  : "1990-01-30",
  "cardnum" : "5827-2842…",
  "expiry" : "2019-03",
  "cardType" : "visa",
  "Connections" : [
     {
        "CustId"   : "XYZ987",
        "Name"     : "Joe Smith"
     },
     {
        "CustId"   : "PQR823",
        "Name"     : "Dylan Smith"
     }
     {
        "CustId"   : "PQR823",
        "Name"     : "Dylan Smith"
     }
  ],
  "Purchases" : [
     { "id":12, item: "mac",   "amt": 2823.52 }
     { "id":19, item: "ipad2", "amt": 623.52 }
  ]
}
```



Contacts

CustomerID	ConnId	Name
CBL2015	XYZ987	Joe Smith
CBL2015	SKR007	Sam Smith

Billing

Customer ID	Type	Cardnum	Expiry
CBL2015	visa	5827...	2019-03
CBL2015	master	6274...	2018-12

Customer

CustomerID	Name	DOB
CBL2015	Jane Smith	1990-01-30

Purchases

CustomerID	item	amt
CBL2015	mac	2823.52
CBL2015	ipad2	623.52

Connections

CustomerID	ConnId	Name
CBL2015	XYZ987	Joe Smith
CBL2015	SKR007	Sam Smith

DocumentKey: **CBL2015**

```
{
   "Name" : "Jane Smith",
   "DOB"  : "1990-01-30",
   "Billing" : [
      {
         "type"   : "visa",
         "cardnum" : "5827-2842-2847-3909",
         "expiry"  : "2019-03"
      },
      {
         "type"   : "master",
         "cardnum" : "6274-2842-2847-3909",
         "expiry"  : "2019-03"
      }
   ],
   "Connections" : [
      {
         "CustId"   : "XYZ987",
         "Name"     : "Joe Smith"
      },
      {
         "CustId"   : "PQR823",
         "Name"     : "Dylan Smith"
      }
      {
         "CustId"   : "PQR823",
         "Name"     : "Dylan Smith"
      }
   ],
   "Purchases" : [
      { "id":12, item: "mac",   "amt": 2823.52 }
      { "id":19, item: "ipad2", "amt": 623.52 }
   ]
}
```

Models for Representing Data

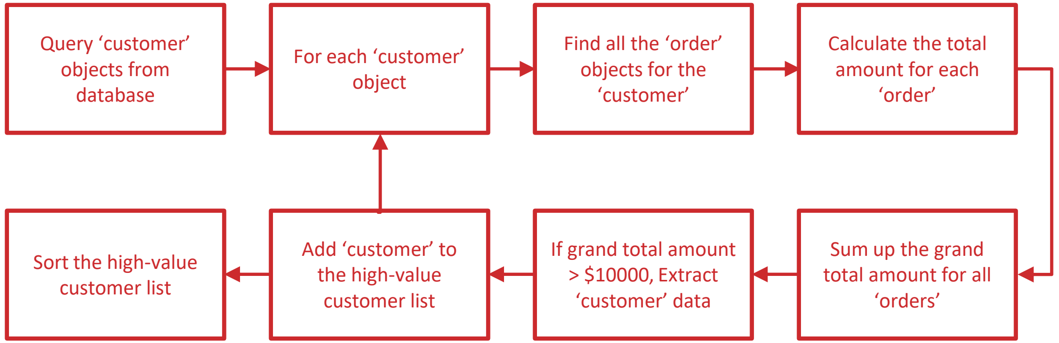
Data Concern	Relational Model	JSON Document Model (NoSQL)
Rich Structure		
Relationships		
Value Evolution		
Structure Evolution		

SQL for JSON

Why SQL for JSON?

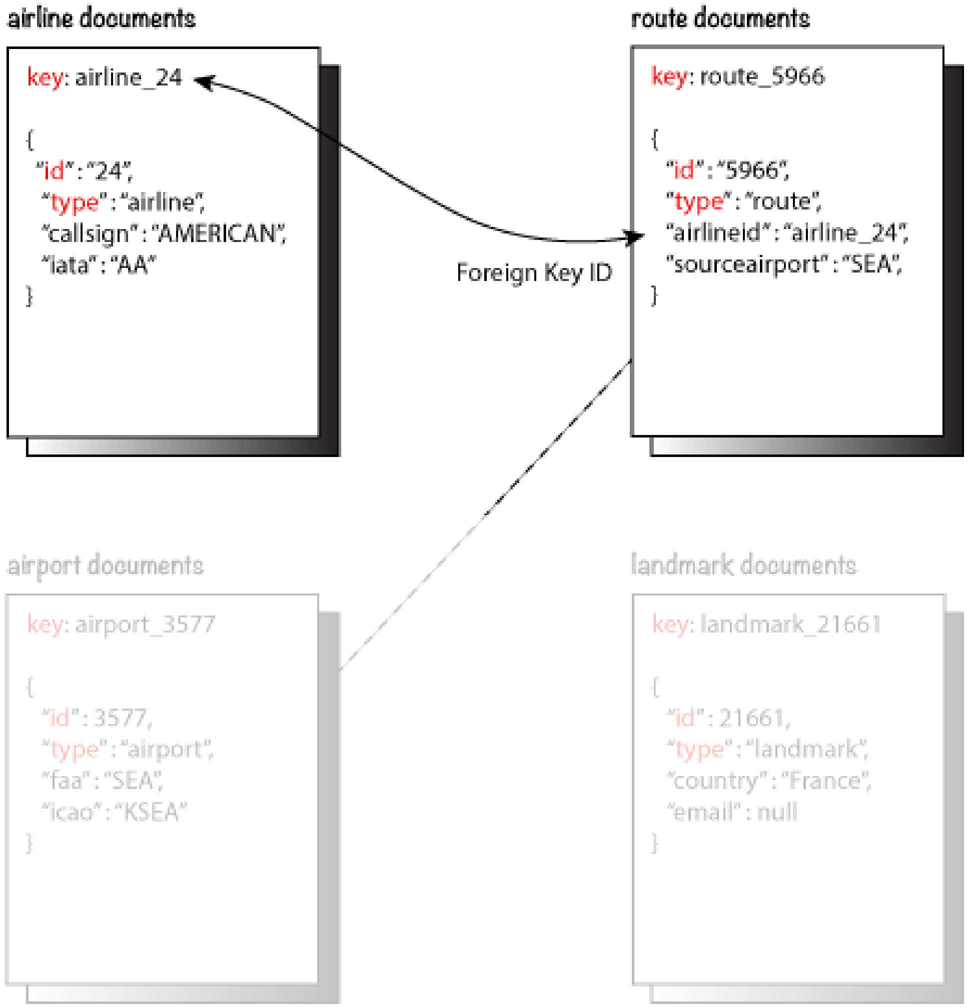
Client Side Querying is Inadequate

Example: Find High-Value Customers with Orders > $10000


Query 'customer' objects from database
For each 'customer' object
Find all the 'order' objects for the 'customer'
Calculate the total amount for each 'order'
Sum up the grand total amount for all 'orders'
If grand total amount > $10000, Extract 'customer' data
Add 'customer' to the high-value customer list
Sort the high-value customer list

Goal of Query for JSON

Give developers and enterprises an expressive, powerful, and complete language for querying, transforming, and manipulating JSON data.



key: airline_24

```
{
 "id":"24",
  "type":"airline",
  "callsign":"AMERICAN",
  "iata":"AA"
}
```

key: route_5966

```
{
  "id":"5966",
  "type":"route",
  "airlineid":"airline_24",
  "sourceairport":"SEA",
}
```

key: airport_3577

```
{
  "id": 3577,
  "type":"airport",
  "faa":"SEA",
  "icao":"KSEA"
}
```

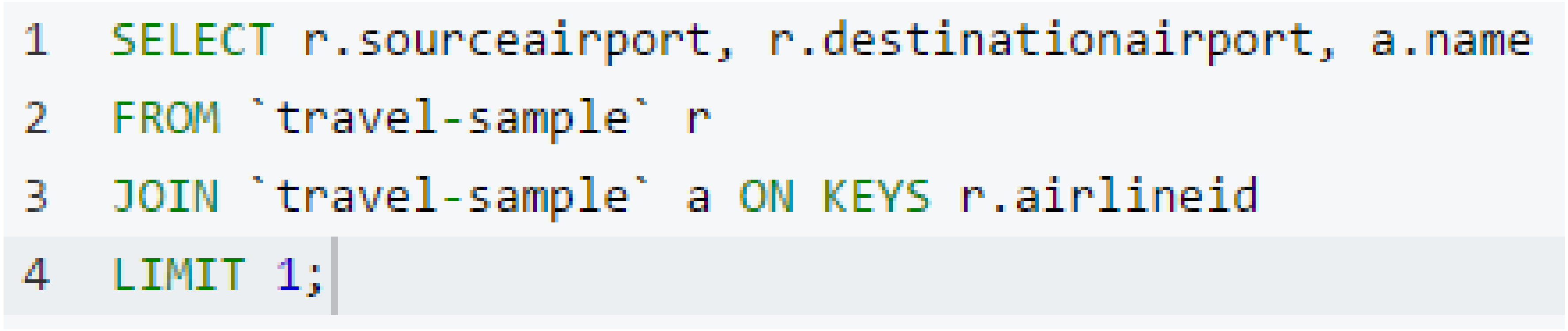
key: landmark_21661

```
{
  "id": 21661,
  "type":"landmark",
  "country":"France",
  "email": null
}
```

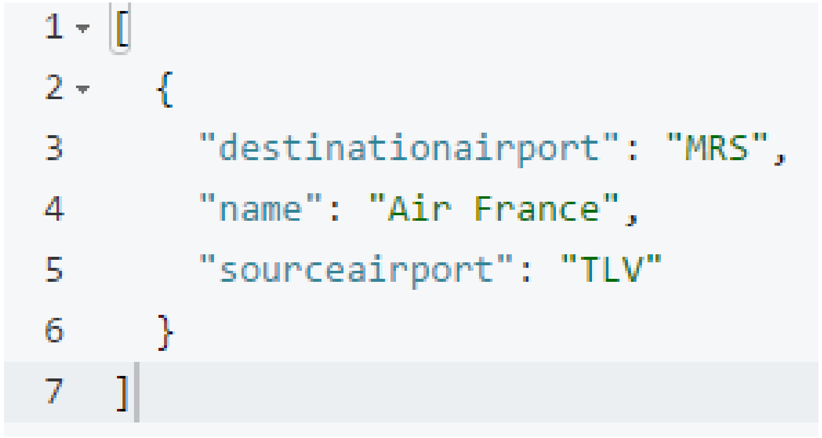
```
route_5966                    ← key
{
  "type": "route",                    ← document type identifier
  "airlineid": "airline_24",                   ← foreign key
  "sourceairport": "MCO",
  "destinationairport": "SEA",
  "equipment": "737",
  "schedule": [
    {"day": 1, "utc": "13:25:00", "flight": "AA788"},
    {"day": 4, "utc": "13:25:00", "flight": "AA419"},
    {"day": 5, "utc": "13:25:00", "flight": "AA519"}
  ]
}
```

```
airline_24                     ← key
{
    "active": "Y",
    "callsign": "AMERICAN",
    "country": "United States",
    "iata": "AA",
    "icao": "AAL",
    "name": "American Airlines",
    "type": "airline"         ← document type identifier
}
```

SELECT: JOIN

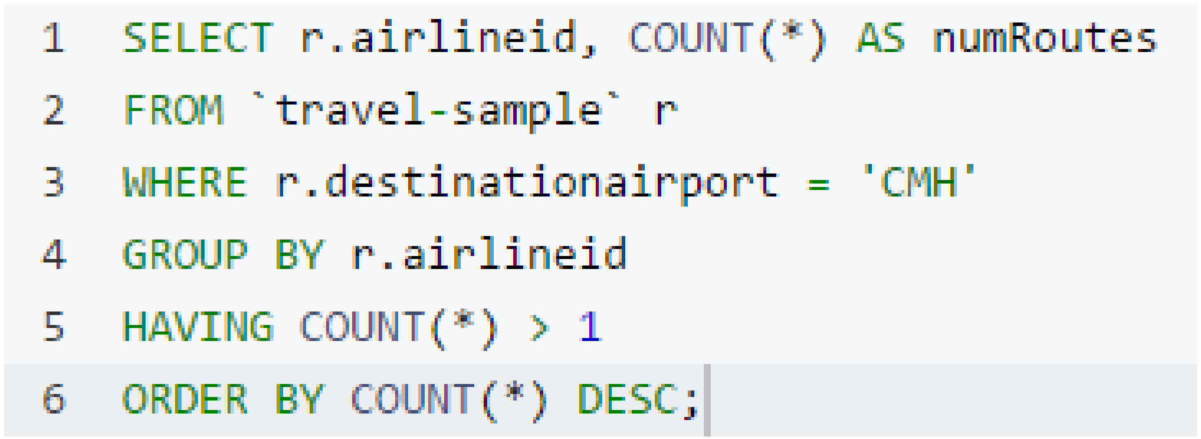


```
SELECT r.sourceairport, r.destinationairport, a.name
FROM `travel-sample` r
JOIN `travel-sample` a ON KEYS r.airlineid
LIMIT 1;
```

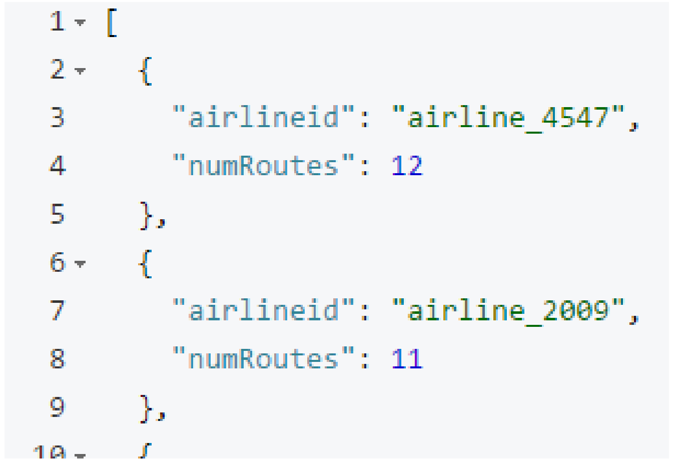


```
[
  {
    "destinationairport": "MRS",
    "name": "Air France",
    "sourceairport": "TLV"
  }
]
```

SELECT: Aggregation



```
SELECT r.airlineid, COUNT(*) AS numRoutes
FROM `travel-sample` r
WHERE r.destinationairport = 'CMH'
GROUP BY r.airlineid
HAVING COUNT(*) > 1
ORDER BY COUNT(*) DESC;
```



```
[
  {
    "airlineid": "airline_4547",
    "numRoutes": 12
  },
  {
    "airlineid": "airline_2009",
    "numRoutes": 11
  },
  {
```

SELECT: Aggregation

MIN

MAX

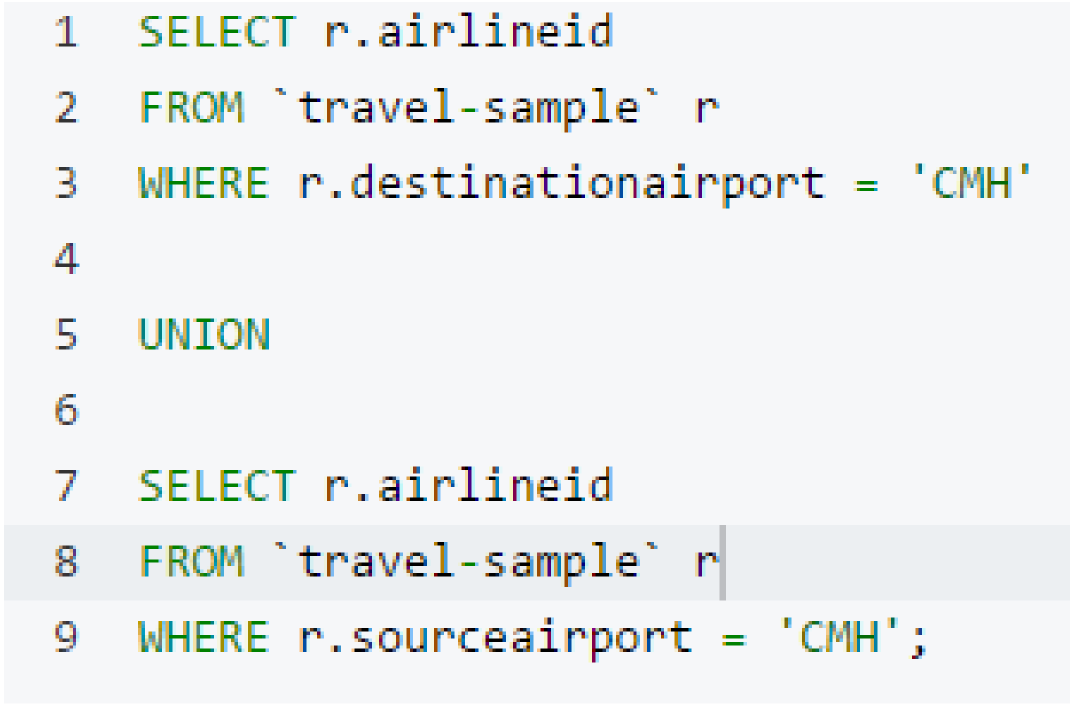
SUM

COUNT

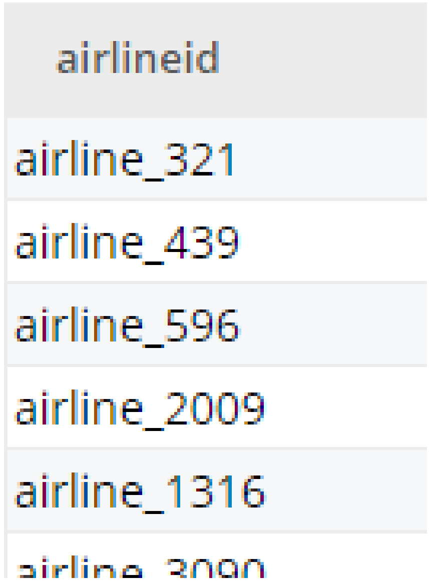
AVG

ARRAY_AGG   [DISTINCT]

SELECT: UNION, UNION ALL, INTERSECT, EXCEPT

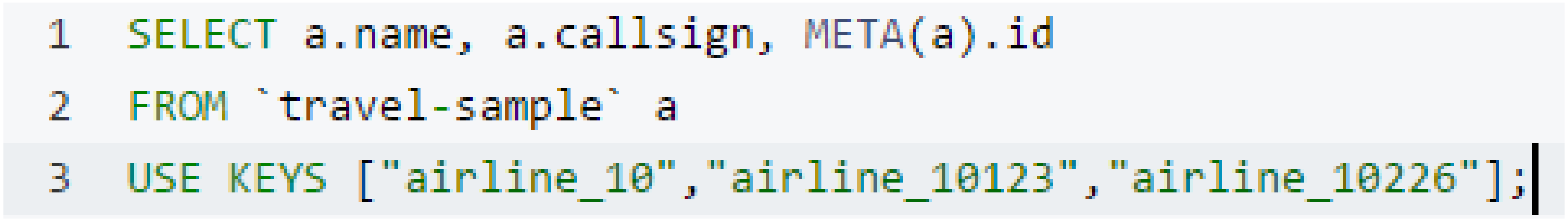


```
SELECT r.airlineid
FROM `travel-sample` r
WHERE r.destinationairport = 'CMH'

UNION

SELECT r.airlineid
FROM `travel-sample` r
WHERE r.sourceairport = 'CMH';
```

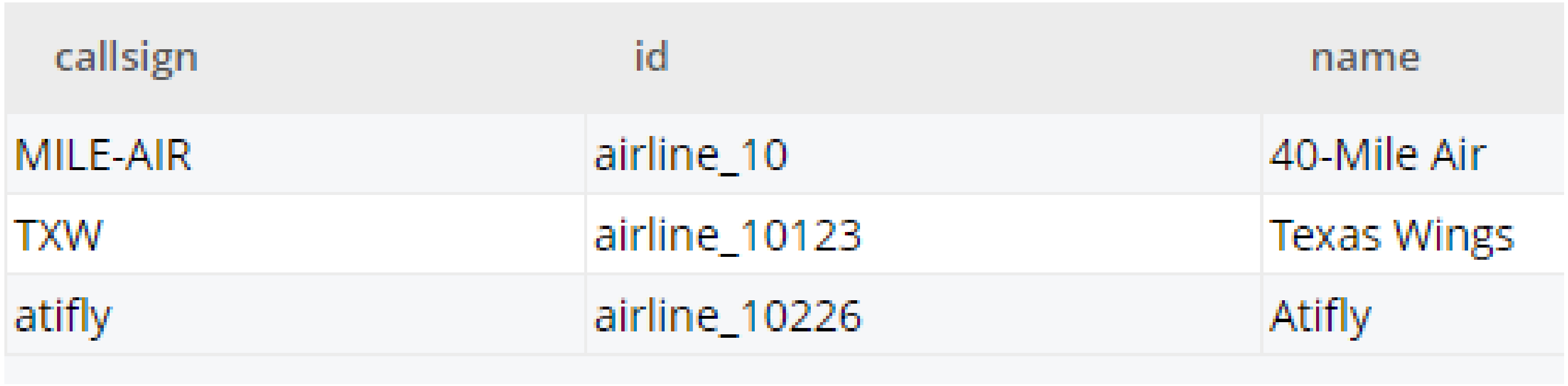


airlineid
airline_321
airline_439
airline_596
airline_2009
airline_1316
airline_3090

USE KEYS

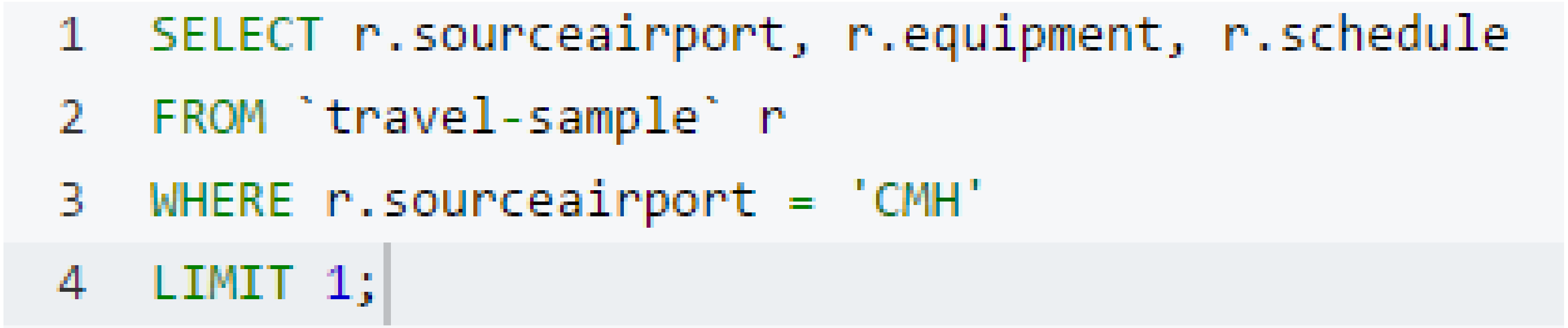


```
SELECT a.name, a.callsign, META(a).id
FROM `travel-sample` a
USE KEYS ["airline_10","airline_10123","airline_10226"];
```



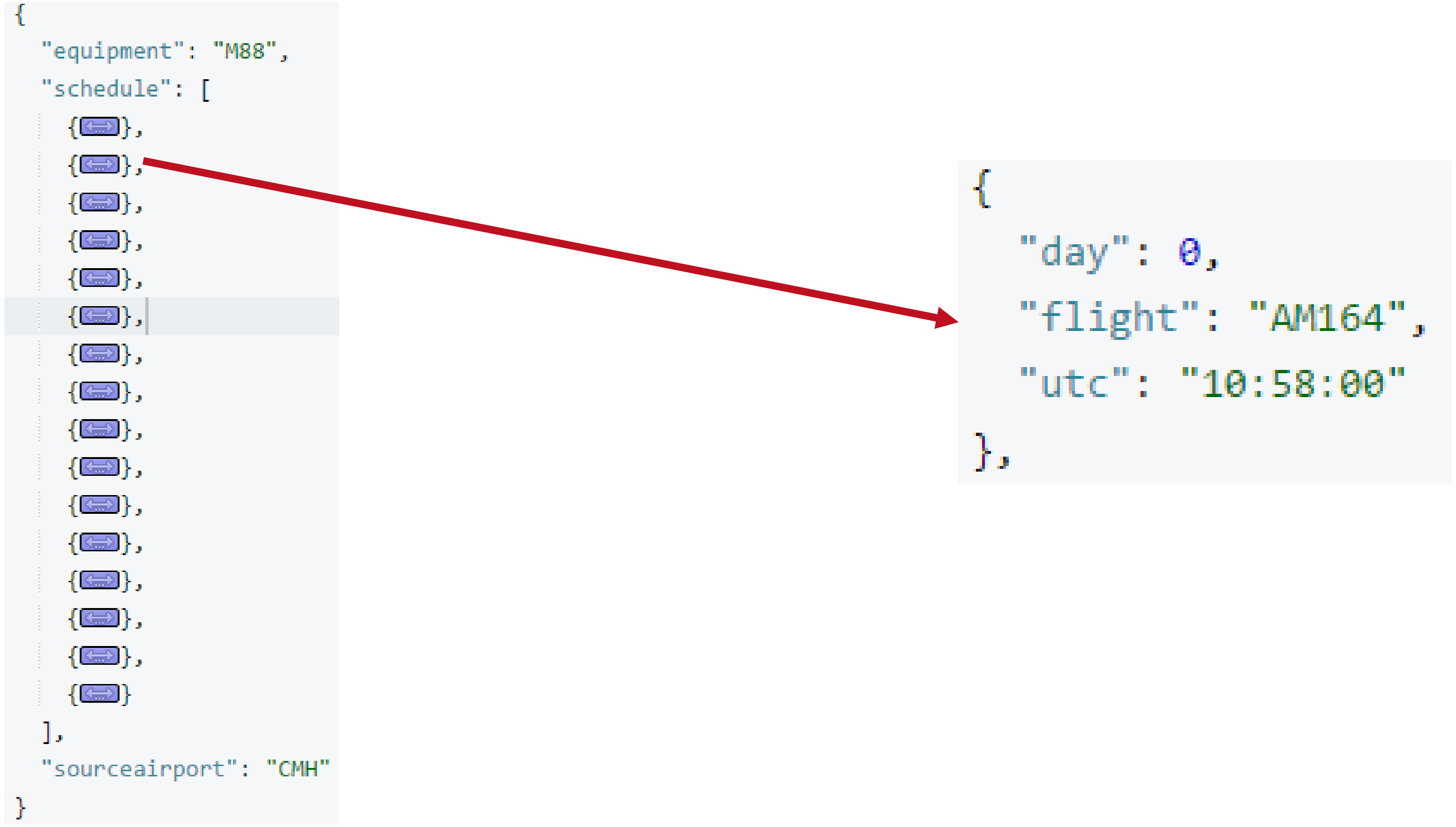
callsign	id	name
MILE-AIR	airline_10	40-Mile Air
TXW	airline_10123	Texas Wings
atifly	airline_10226	Atifly

UNNEST (aka JOIN)



```
SELECT r.sourceairport, r.equipment, r.schedule
FROM `travel-sample` r
WHERE r.sourceairport = 'CMH'
LIMIT 1;
```

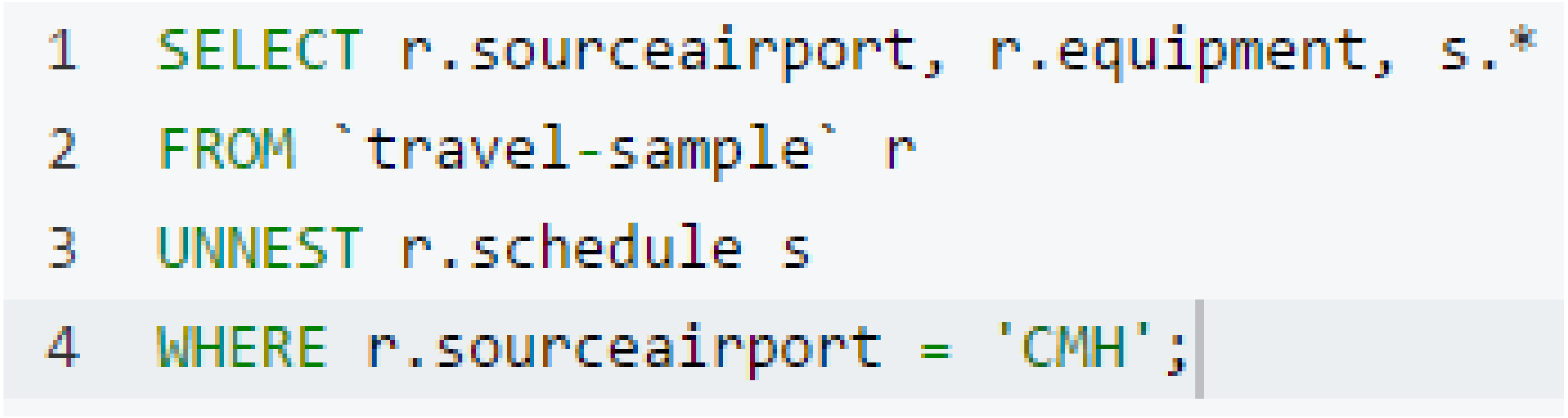
UNNEST (aka JOIN)



```
{
  "equipment": "M88",
  "schedule": [
    {↔},
    {↔},
    {↔},
    {↔},
    {↔},
    {↔},
    {↔},
    {↔},
    {↔},
    {↔},
    {↔},
    {↔},
    {↔},
    {↔},
    {↔},
    {↔}
  ],
  "sourceairport": "CMH"
}
```

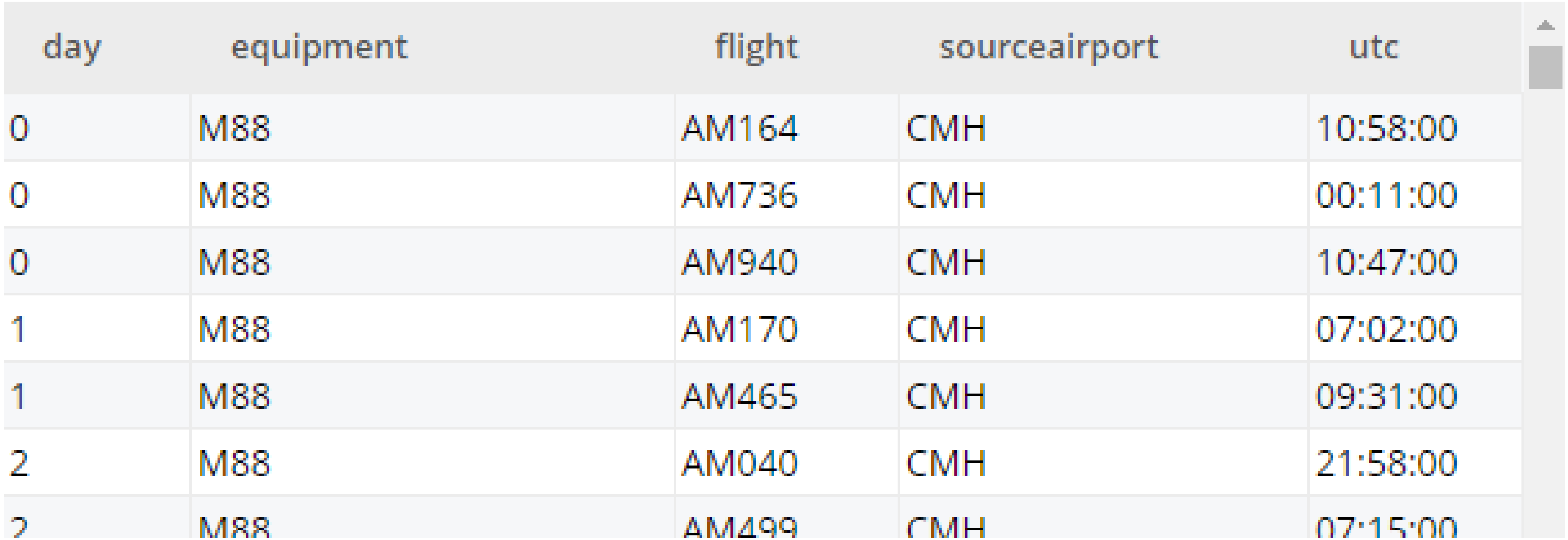
```
{
  "day": 0,
  "flight": "AM164",
  "utc": "10:58:00"
},
```

UNNEST (aka JOIN)



```
SELECT r.sourceairport, r.equipment, s.*
FROM `travel-sample` r
UNNEST r.schedule s
WHERE r.sourceairport = 'CMH';
```

UNNEST (aka JOIN)



day	equipment	flight	sourceairport	utc
0	M88	AM164	CMH	10:58:00
0	M88	AM736	CMH	00:11:00
0	M88	AM940	CMH	10:47:00
1	M88	AM170	CMH	07:02:00
1	M88	AM465	CMH	09:31:00
2	M88	AM040	CMH	21:58:00
2	M88	AM499	CMH	07:15:00

Composability

JOIN, NEST, and UNNEST

can be chained in any combination

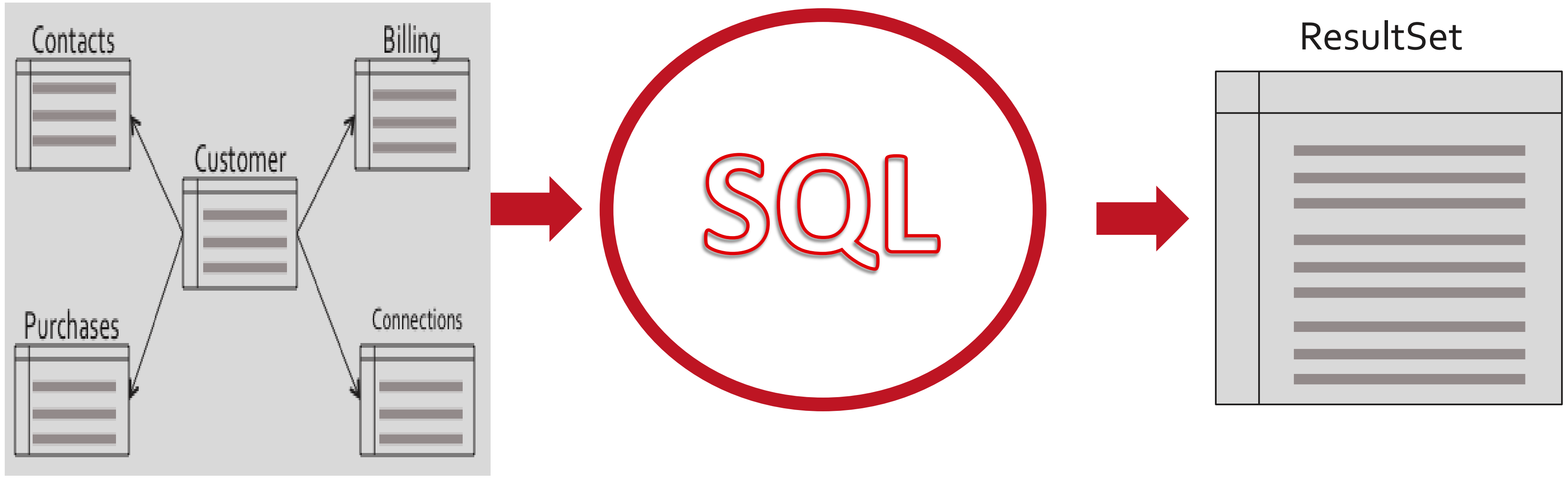
SELECT Statement

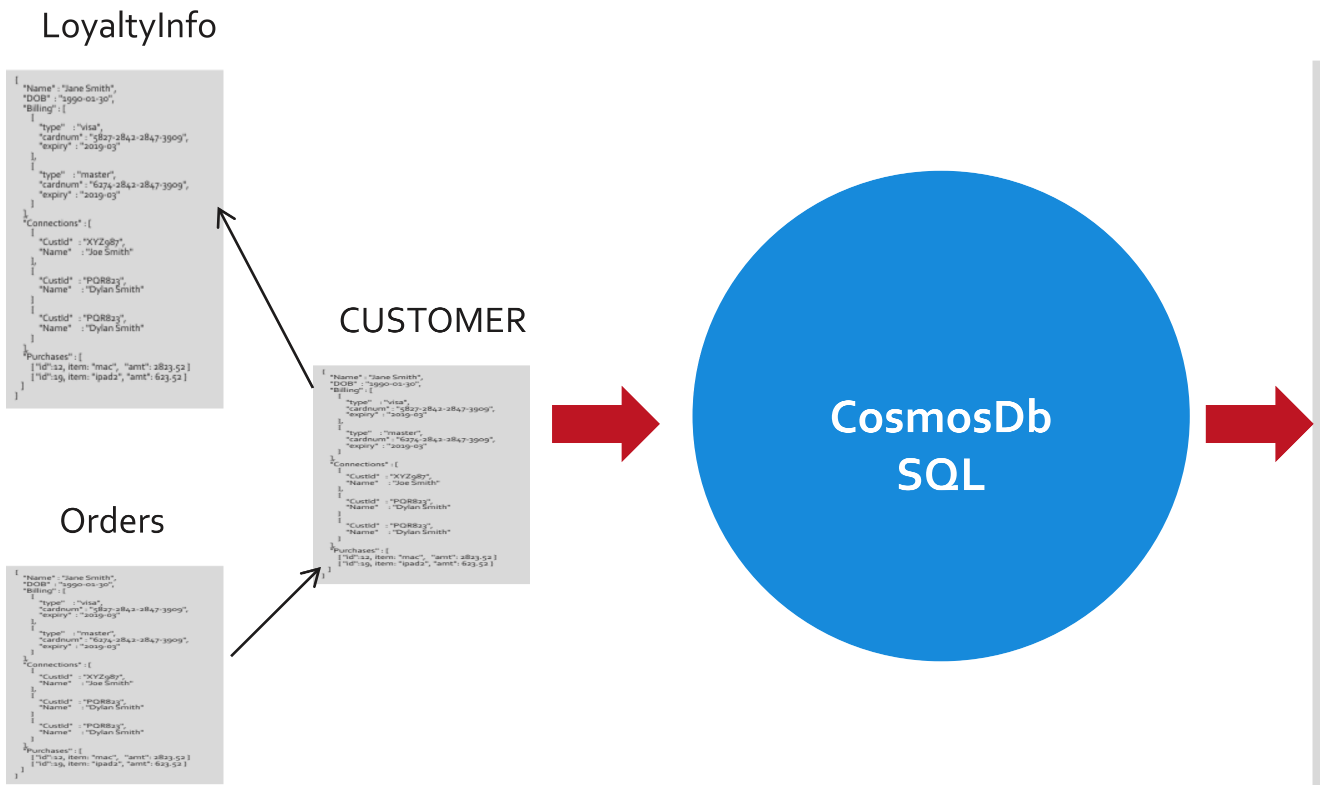
```
SELECT    customers.id,
          customers.NAME.lastname,
          customers.NAME.firstname
          Sum(orderline.amount)
FROM      `orders` UNNEST orders.lineitems AS orderline
          JOIN `customers` ON KEYS orders.custid
WHERE customers.state = 'NY'
GROUP BY customers.id,
          customers.NAME.lastname
HAVING    sum(orderline.amount) > 10000
ORDER BY sum(orderline.amount) DESC
```

- **Dotted sub-document reference**
- **Names are CASE-SENSITIVE**

- **UNNEST to flatten the arrays**

JOINS with Document KEY of customers

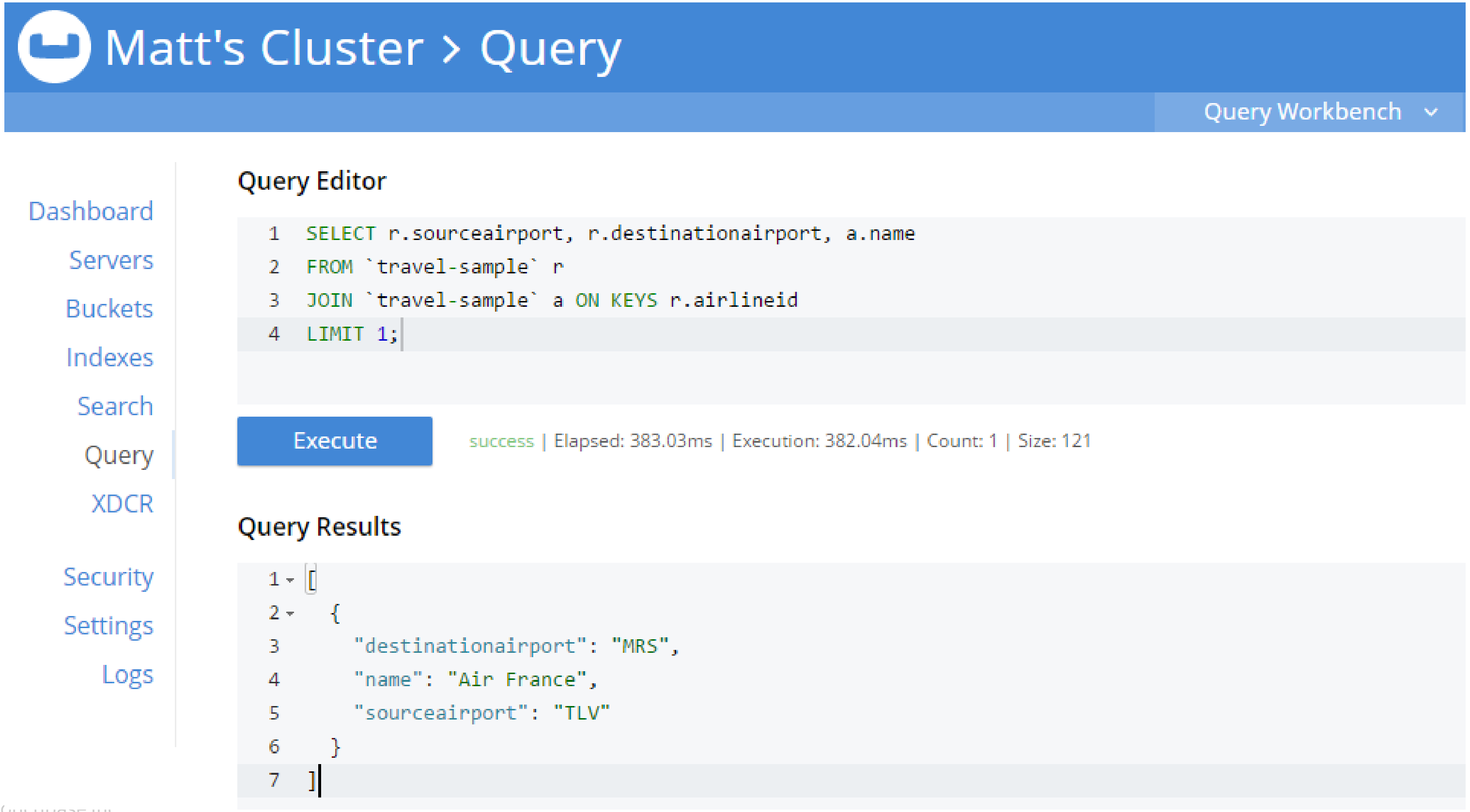

Contacts
Billing
Customer
Purchases
Connections
SQL
ResultSet



ResultDocuments

```
{
  "Name" : "Jane Smith",
  "DOB"  : "1990-01-30",
  "Billing" : [
    {
      "type"    : "visa",
      "cardnum" : "5827-2842-2847-3909",
      "expiry"  : "2019-03"
    },
    {
      "type"    : "master",
      "cardnum" : "6274-2842-2847-3909",
      "expiry"  : "2019-03"
    }
  ],
  "Connections" : [
    {
      "CustId"   : "XYZ987",
      "Name"     : "Joe Smith"
    },
    {
      "CustId"   : "PQR823",
      "Name"     : "Dylan Smith"
    }
    {
      "CustId"   : "PQR823",
      "Name"     : "Dylan Smith"
    }
  ],
  "Purchases" : [
    { "id":12, item: "mac",  "amt": 2823.52 }
    { "id":19, item: "ipad2", "amt": 623.52 }
  ]
}
```

N1QL (Couchbase)



Query Editor

```
SELECT r.sourceairport, r.destinationairport, a.name
FROM `travel-sample` r
JOIN `travel-sample` a ON KEYS r.airlineid
LIMIT 1;
```

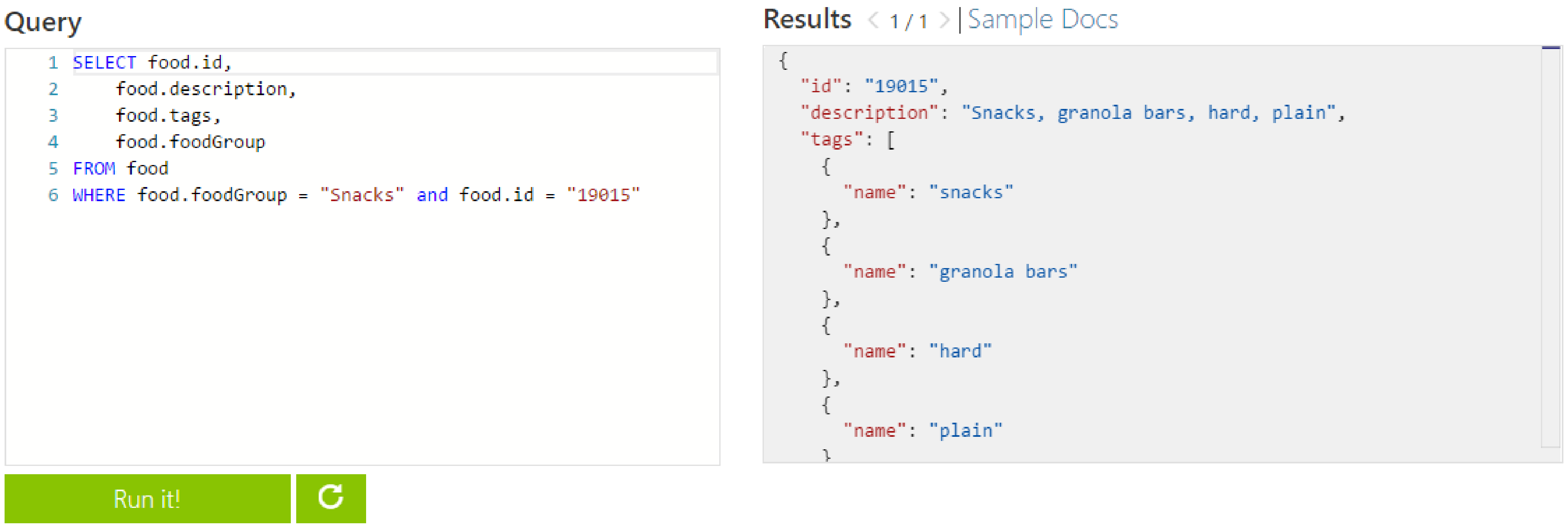
success | Elapsed: 383.03ms | Execution: 382.04ms | Count: 1 | Size: 121

Query Results

```
[
  {
    "destinationairport": "MRS",
    "name": "Air France",
    "sourceairport": "TLV"
  }
]
```

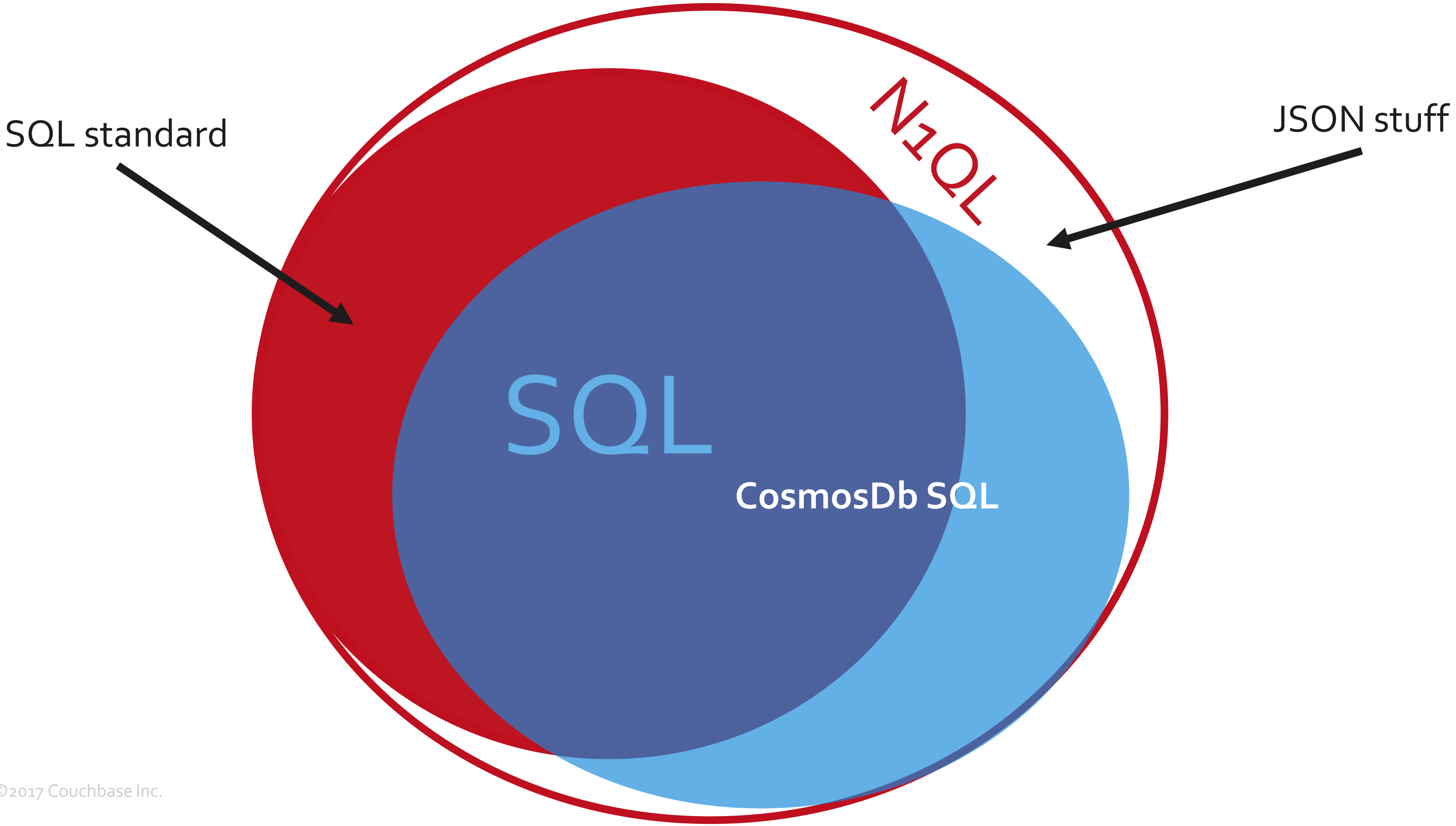
SQL (~~DocumentDb~~ CosmosDb)

- https://www.documentdb.com/sql/demo



```
SELECT food.id,
    food.description,
    food.tags,
    food.foodGroup
FROM food
WHERE food.foodGroup = "Snacks" and food.id = "19015"
```

```
{
  "id": "19015",
  "description": "Snacks, granola bars, hard, plain",
  "tags": [
    {
      "name": "snacks"
    },
    {
      "name": "granola bars"
    },
    {
      "name": "hard"
    },
    {
      "name": "plain"
    }
```


SQL standard
N1QL
JSON stuff
SQL
CosmosDb SQL

Demo: N1QL

Other JSON "Stuff"

Ranging over collections	• WHERE **ANY** c IN children SATISFIES c.age > 10 END • WHERE **EVERY** r IN ratings SATISFIES r > 3 END
Mapping with filtering	• **ARRAY** c.name FOR c IN children **WHEN** c.age > 10 END
Deep traversal, SET, and UNSET	• WHERE ANY node **WITHIN** request SATISFIES node.type = “xyz” END • UPDATE doc **UNSET** c.field1 FOR c **WITHIN** doc END
Dynamic Construction	• SELECT **{ “a”: expr1, “b”: expr2 }** AS obj1, name FROM … // Dynamic object • SELECT **[a, b]** FROM … // Dynamic array
Nested traversal	• SELECT **x.y.z, a[0]** FROM **a.b.c** …
IS [NOT] MISSING	• WHERE name **IS MISSING**

55

Data Types in JSON

N1QL supports all JSON data types

- Numbers
- Strings
- Booleans
- Null
- Arrays
- Objects

Data Modification Statements

JSON literals can be used in any expression

```
INSERT INTO `orders` (KEY, VALUE)
VALUES ("mykey", {"field1":482, "field2":3, "field3":4});
```

```
UPDATE `order`
    SET field4 = "ABC987"
WHERE field1 = 482 AND field2 = 3 AND field3 = 4
```

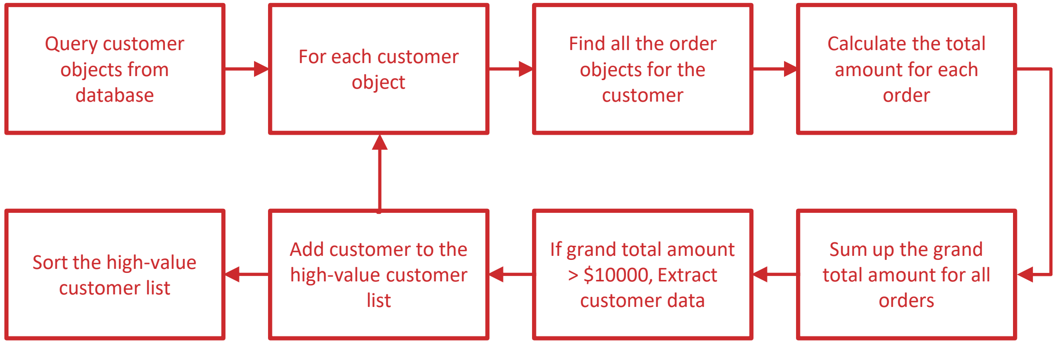
```
DELETE FROM `neworder`
WHERE fieldX = 291 AND
      fieldY = 3482 AND
      fieldZ = 2483
```

Index Statements

- CREATE INDEX ON …
- DROP INDEX …
- EXPLAIN …

https://dzone.com/articles/index-first-and-query-faster

Find High-Value Customers with Orders > $10000


Query customer objects from database
For each customer object
Find all the order objects for the customer
Calculate the total amount for each order
Sum up the grand total amount for all orders
If grand total amount > $10000, Extract customer data
Add customer to the high-value customer list
Sort the high-value customer list

```
SELECT     Customers.ID, Customers.Name, SUM(OrderLine.Amount)
FROM       `Orders` UNNEST Orders.LineItems AS OrderLine
           JOIN `Customers` ON KEYS Orders.CustID
GROUP BY            Customers.ID, Customers.Name
HAVING     SUM(OrderLine.Amount) > 10000
ORDER BY   SUM(OrderLine.Amount) DESC
```

Summary: SQL & SQL for JSON

Query Features	SQL	SQL for JSON
Statements	▪ SELECT, INSERT, UPDATE, DELETE, MERGE	▪ SELECT, INSERT, UPDATE, DELETE, MERGE
Query Operations	▪ Select, Join, Project, Subqueries ▪ Strict Schema ▪ Strict Type checking	▪ Select, Join, Project, Subqueries ✓ Nest & Unnest ✓ Look Ma! No Type Mismatch Errors! ▪ JSON keys act as columns
Schema	▪ Predetermined Columns	✓ Fully addressable JSON ✓ Flexible document structure
Data Types	▪ SQL Data types ▪ Conversion Functions	▪ JSON Data types ▪ Conversion Functions
Query Processing	▪ INPUT: Sets of Tuples ▪ OUPUT: Set of Tuples	▪ INPUT: Sets of JSON ▪ OUTPUT: Set of JSON

Summary: N1QL and CosmosDB SQL

Query Features	N1QL	CosmosDB SQL
SELECT	Yes	Yes
INSERT, UPDATE, DELETE, MERGE	Yes	No
Intra-document join	Yes: UNNEST	Yes: JOIN
Inter-document join	Yes: JOIN, NEST	No
Aggregation (GROUP BY, SUM,MAX,MIN)	Yes	No GROUP BY
Stored Procedures, Triggers, UDF	No	Kinda (JavaScript)

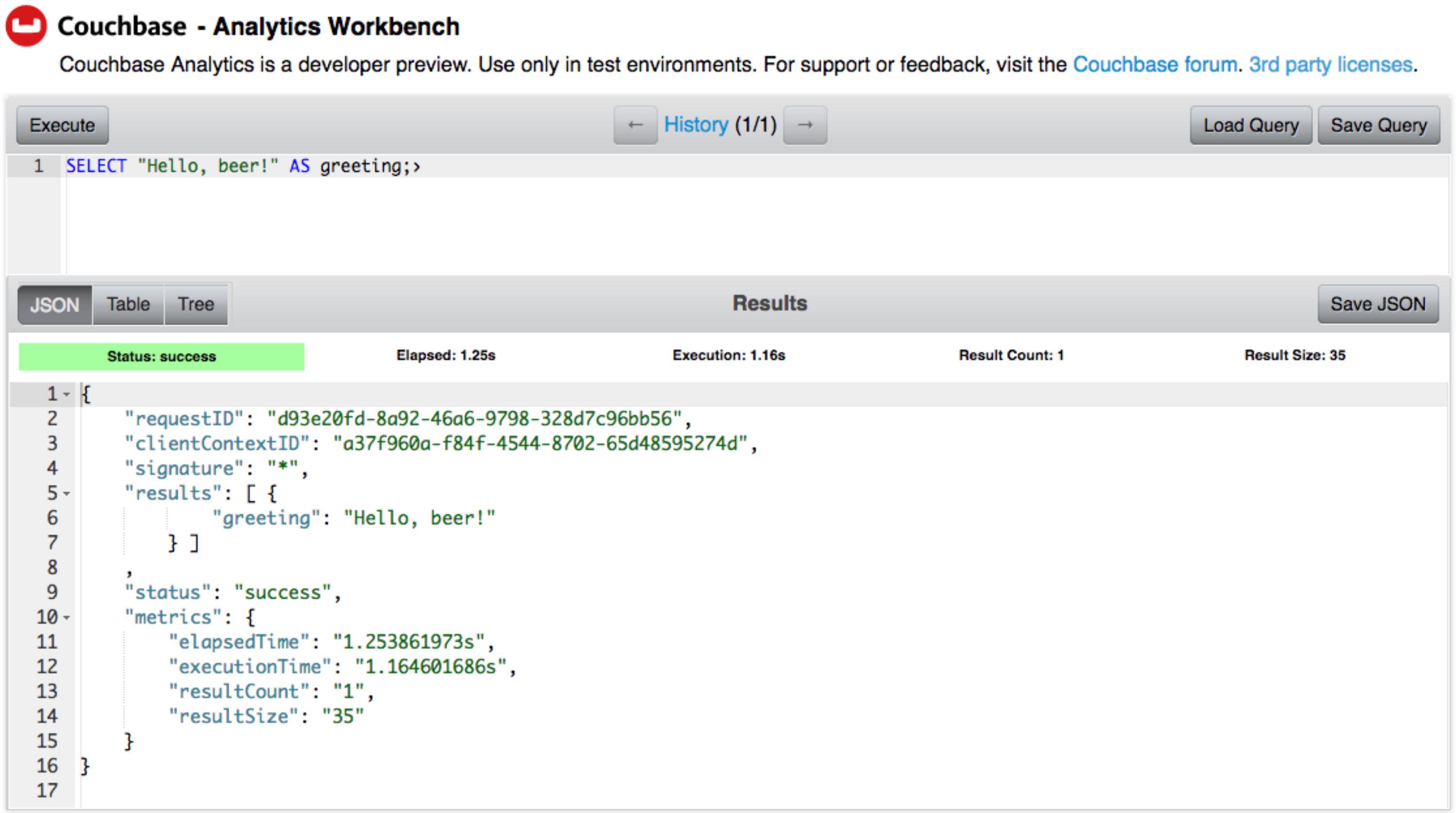
SQL for JSON

UC San Diego SQL++

Couchbase N1QL

http://tinyurl.com/UCSDsql

Couchbase Analytics (CBAS) and SQL++



Couchbase Analytics is a developer preview. Use only in test environments. For support or feedback, visit the Couchbase forum. 3rd party licenses.

```
SELECT "Hello, beer!" AS greeting;
```

Status: success   Elapsed: 1.25s   Execution: 1.16s   Result Count: 1   Result Size: 35

```
{
    "requestID": "d93e20fd-8a92-46a6-9798-328d7c96bb56",
    "clientContextID": "a37f960a-f84f-4544-8702-65d48595274d",
    "signature": "*",
    "results": [ {
            "greeting": "Hello, beer!"
        } ]
    ,
    "status": "success",
    "metrics": {
        "elapsedTime": "1.253861973s",
        "executionTime": "1.164601686s",
        "resultCount": "1",
        "resultSize": "35"
    }
}
```

SQL EMPIRE STRIKES BACK

Industry is Choosing SQL to Query JSON

IBM

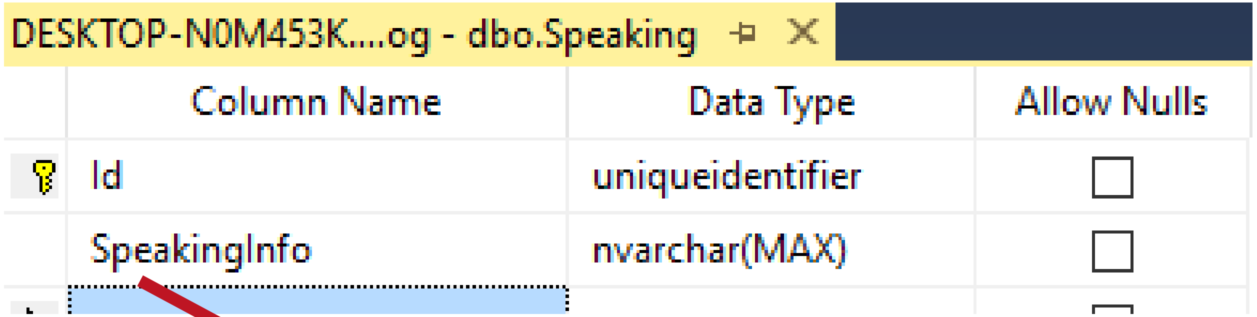
ORACLE®

PostgreSQL

Microsoft® SQL Server™®

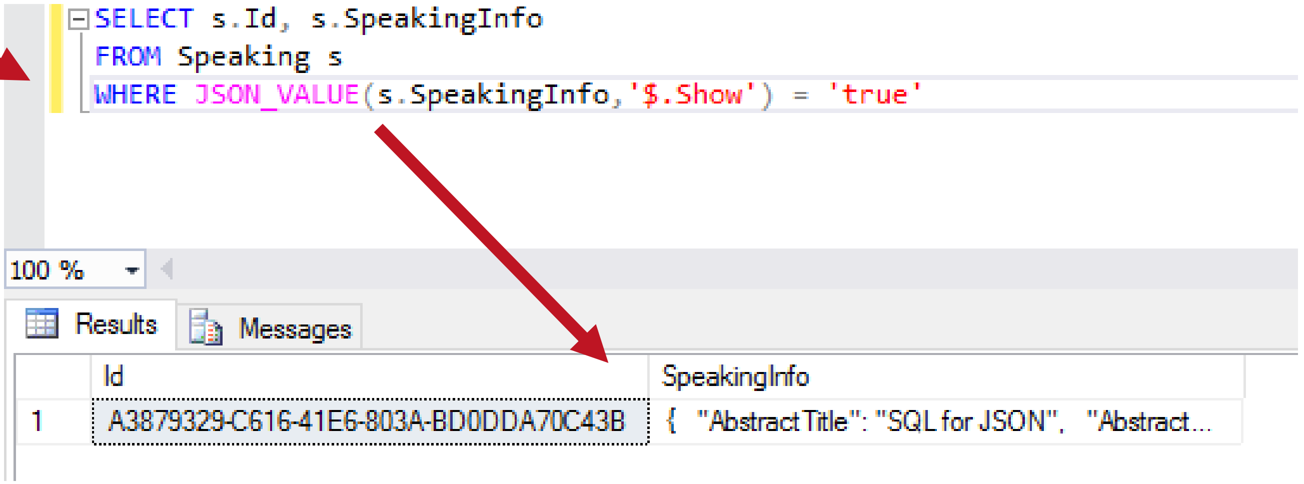
MySQL®

JSON in SQL Server



DESKTOP-N0M453K....og - dbo.Speaking

	Column Name	Data Type	Allow Nulls
🔑	Id	uniqueidentifier	☐
	SpeakingInfo	nvarchar(MAX)	☐



```
SELECT s.Id, s.SpeakingInfo
FROM Speaking s
WHERE JSON_VALUE(s.SpeakingInfo,'$.Show') = 'true'
```

	Id	SpeakingInfo
1	A3879329-C616-41E6-803A-BD0DDA70C43B	{ "AbstractTitle": "SQL for JSON", "Abstract...

Try SQL for JSON

http://tiny.cc/n1ql

http://tiny.cc/cosmosdb

Podcast

https://blog.couchbase.com/tag/podcast/

Free event:

Couchbase Day Kansas City

August 15th

http://tiny.cc/cbkc

Couchbase, everybody!

Where do you find us?

- blog.couchbase.com
- @couchbasedev
- @mgroves
- http://tiny.cc/cbkc

Couchbase
DEVELOPER COMMUNITY

Frequently Asked Questions

1. How is Couchbase different than Mongo?
2. Is Couchbase the same thing as CouchDb?
3. How did you get to be both incredibly handsome *and* tremendously intelligent?
4. What is the Couchbase licensing situation?
5. Is Couchbase a Managed Cloud Service?

CouchDB and Couchbase

CouchDB relax

CouchOne

Couchbase

membase

memcached

< Back

MongoDB vs Couchbase

mongoDB®

Couchbase

- Memory first architecture
- Master-master architecture
- Auto-sharding
- N1QL
- Full Text Search
- Mobile & Sync

Licensing

Couchbase Server Community

- Open source (Apache 2)
- Binary release is one release behind Enterprise
- Free to use in dev/test/qa/prod
- Forum support only

Couchbase Server Enterprise

- Mostly open source (Apache 2)
- Some features not available on Community (XDCR TLS, MDS, Rack Zone, etc)
- Free to use in dev/test/qa
- Need commercial license for prod
- Paid support provided

< Back

Managed Cloud Service (DBaaS)?

No.

< Back