



Querying NoSQL with SQL

HAVING Your JSON Cake and SELECTing it too

[YOU MAY WANT TO SIT CLOSE]

Matthew D. Groves, @mgroves

2017 Cincinnati Day of Agile & Cincy.Develop(); Sponsors

Diamond



Gold



Silver





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

- Cincy.Develop(); / Day of Agile
- <http://dayofagile.com/>





- 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



Foot Locker

Communications



at&t



Globe



E-Commerce & Digital Advertising



Finance & Business Services



Travel & Hospitality



Games & Gaming



Media & Entertainment





Why NoSQL?



Key-Value

- Couchbase
- Riak
- BerkeleyDB
- Redis

Document

- Couchbase
- MongoDB
- DynamoDB
- DocumentDB

Wide Column

- Hbase
- Cassandra
- Hypertable

Graph

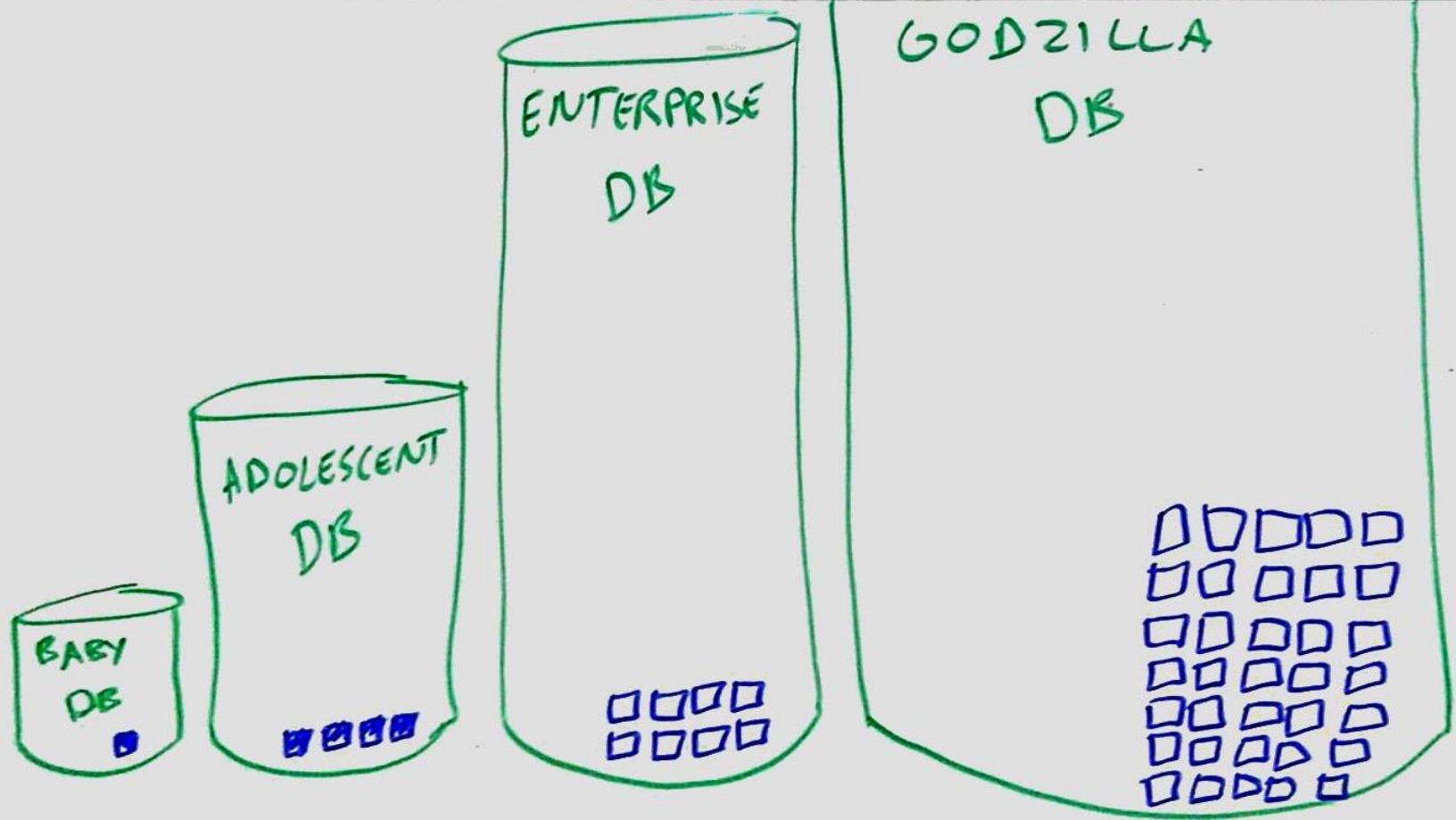
- OrientDB
- Neo4J
- DEX
- GraphBase



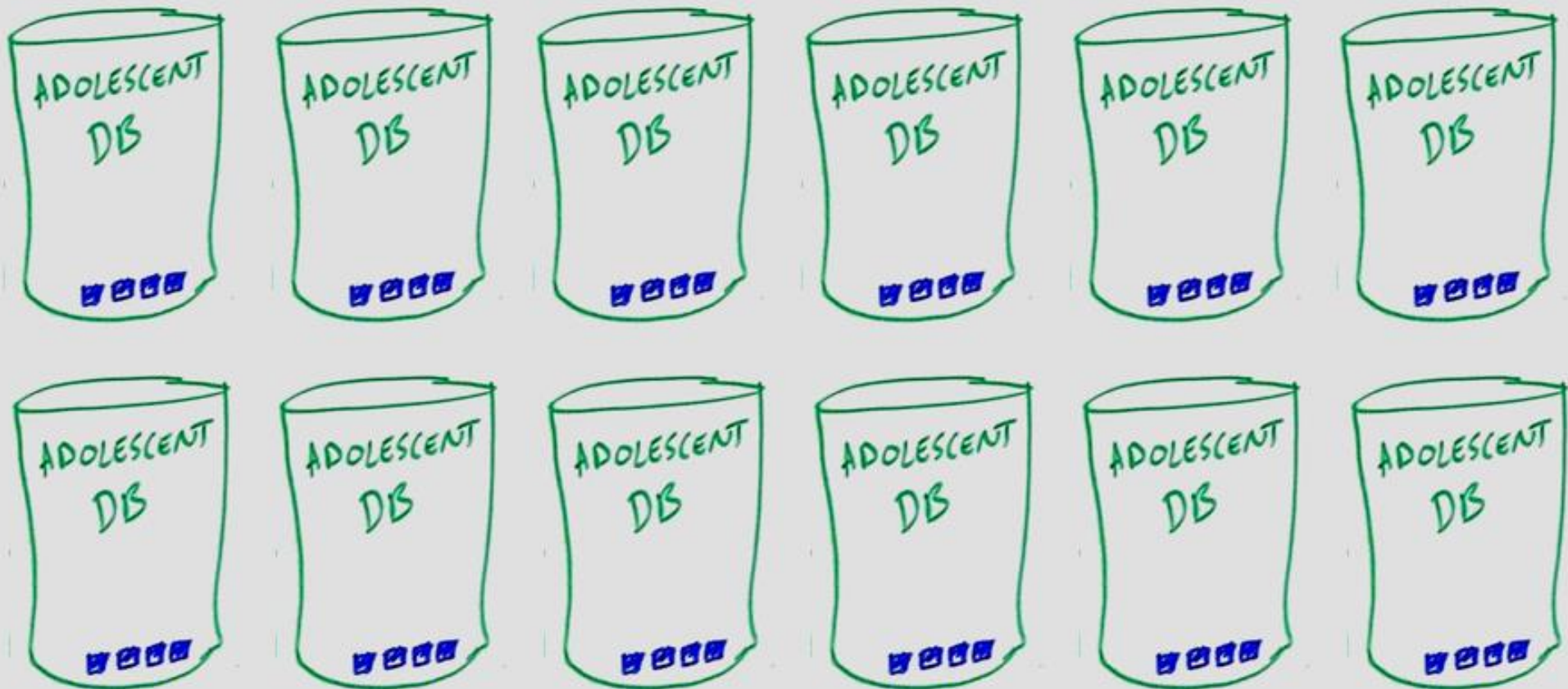
Document

- Couchbase
 - MongoDB
 - DynamoDB
 - DocumentDB
- Get by key(s)
 - Set by key(s)
 - Replace by key(s)
 - Delete by key(s)
 - Map/Reduce

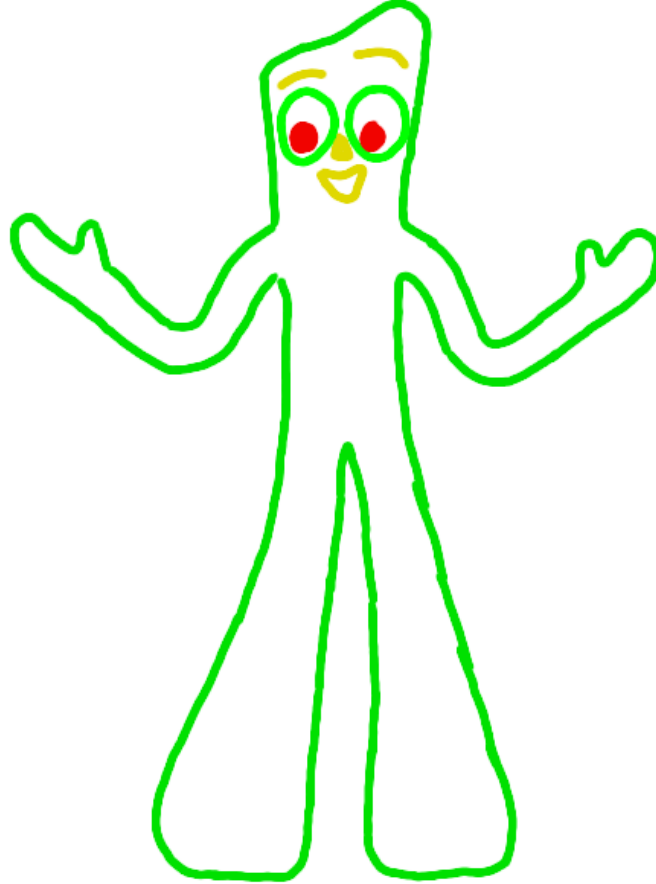
Why NoSQL? Scalability



Why NoSQL? Scalability



Why NoSQL? Flexibility



Why NoSQL? Performance



Why NoSQL? Availability





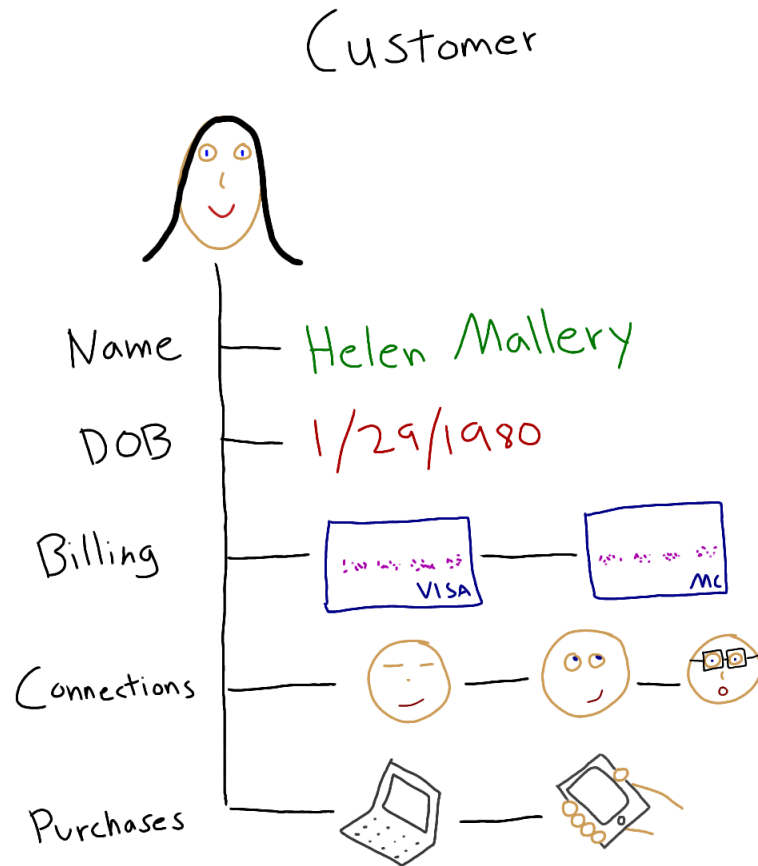
JSON Data Modeling

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

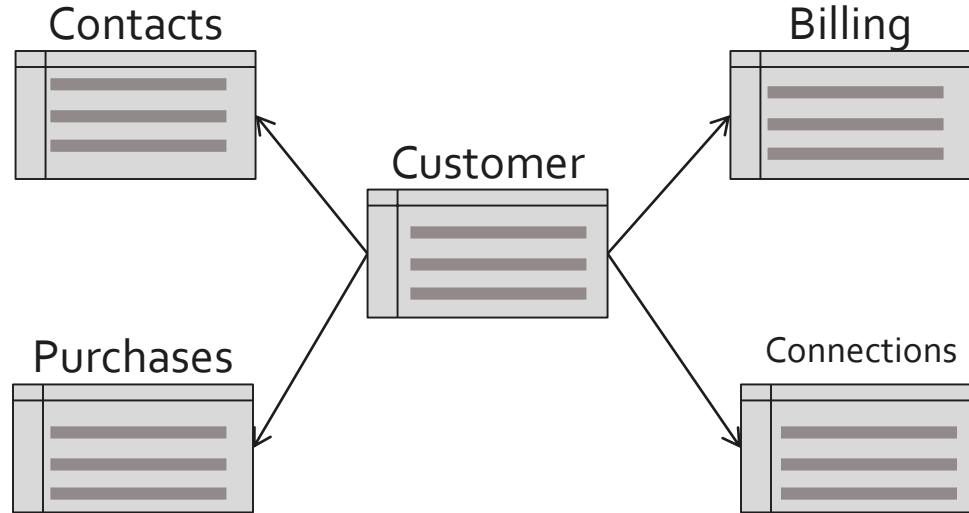


Table: Customer

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

```
{
  "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 }
  ]
}
```


DocumentKey: **CBL2015**

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

```
{
  "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"
    }
  ],
  "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		

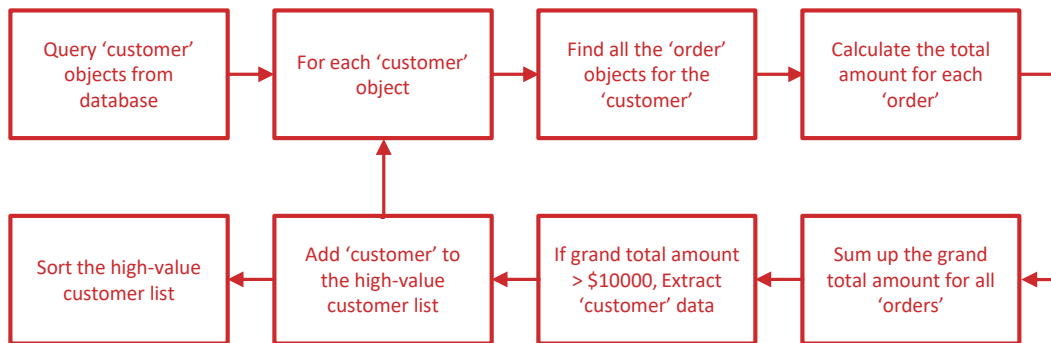


SQL for JSON

Why SQL for JSON?



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





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

airline documents

key: airline_24

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

route documents

key: route_5966

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

Foreign Key ID

airport documents

key: airport_3577

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

landmark documents

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



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

```
1  [
2    {
3      "destinationairport": "MRS",
4      "name": "Air France",
5      "sourceairport": "TLV"
6    }
7  ]
```

SELECT: Aggregation



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

```
1 - [
2 - {
3     "airlineid": "airline_4547",
4     "numRoutes": 12
5 },
6 - {
7     "airlineid": "airline_2009",
8     "numRoutes": 11
9 },
10 - ]
```



MIN

MAX

SUM

COUNT

AVG

ARRAY_AGG [DISTINCT]

SELECT: UNION, UNION ALL, INTERSECT, EXCEPT



```
1  SELECT r.airlineid
2  FROM `travel-sample` r
3  WHERE r.destinationairport = 'CMH'
4
5  UNION
6
7  SELECT r.airlineid
8  FROM `travel-sample` r
9  WHERE r.sourceairport = 'CMH';
```

airlineid
airline_321
airline_439
airline_596
airline_2009
airline_1316
airline_3000

```
1  SELECT a.name, a.callsign, META(a).id
2  FROM `travel-sample` a
3  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

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



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




```
1  SELECT r.sourceairport, r.equipment, s.*
2  FROM `travel-sample` r
3  UNNEST r.schedule s
4  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

matt_groves

```
1 {  
2   "name": "Matt Groves",  
3   "preferredAirlines": [  
4     "airline_10",  
5     "airline_10123",  
6     "airline_10226"  
7   ]  
8 }
```



JOIN, NEST, and UNNEST

can be chained in any combination



SELECT Statement

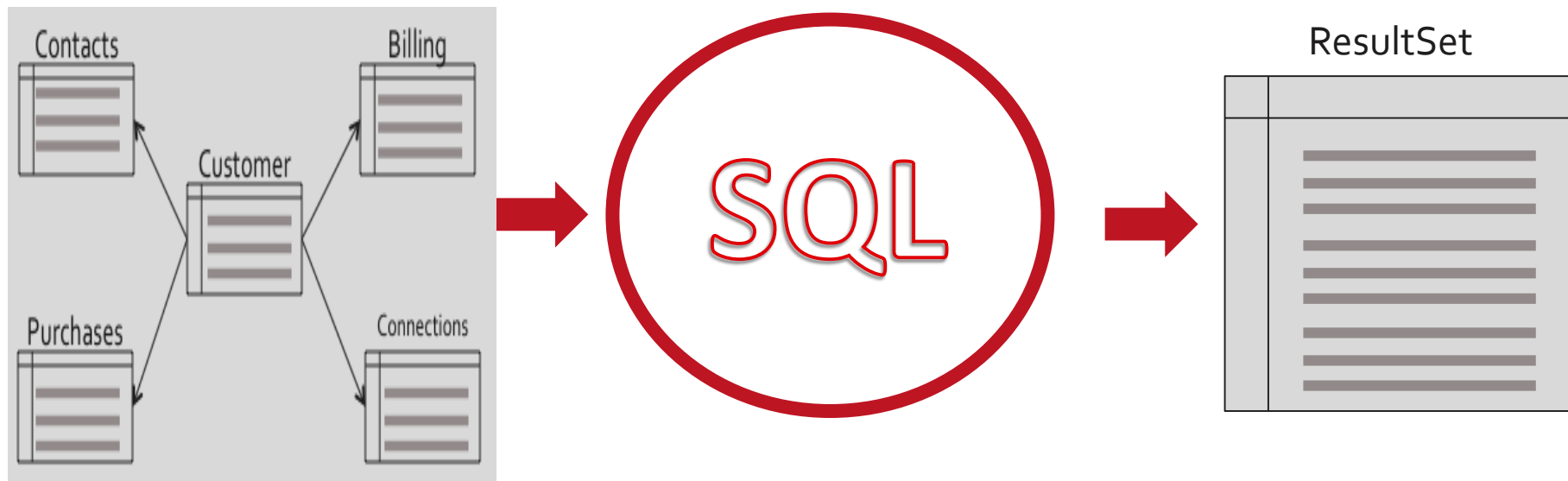


```
SELECT      customers.id,  
            customers.NAME.lastname,  
            customers.NAME.firstname  
            Sum(orderline.amount)  
FROM        `orders` UNNEST orders.lineitems AS orderline  
            JOIN `customers` ON KEYS orders.custid  
WHERE       customers.state = 'NY'  
GROUP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



LoyaltyInfo

```
{
  "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 }
  ]
}
```

Orders

```
{
  "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 }
  ]
}
```

CUSTOMER

```
{
  "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 }
  ]
}
```

CosmosDb SQL

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 }
  ]
}
```

Matt's Cluster > Query

Query Workbench ▾

Dashboard

Servers

Buckets

Indexes

Search

Query

XDCR

Security

Settings

Logs

Query Editor

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

Execute

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

Query Results

```
1 {
2   {
3     "destinationairport": "MRS",
4     "name": "Air France",
5     "sourceairport": "TLV"
6   }
7 }
```


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

Query

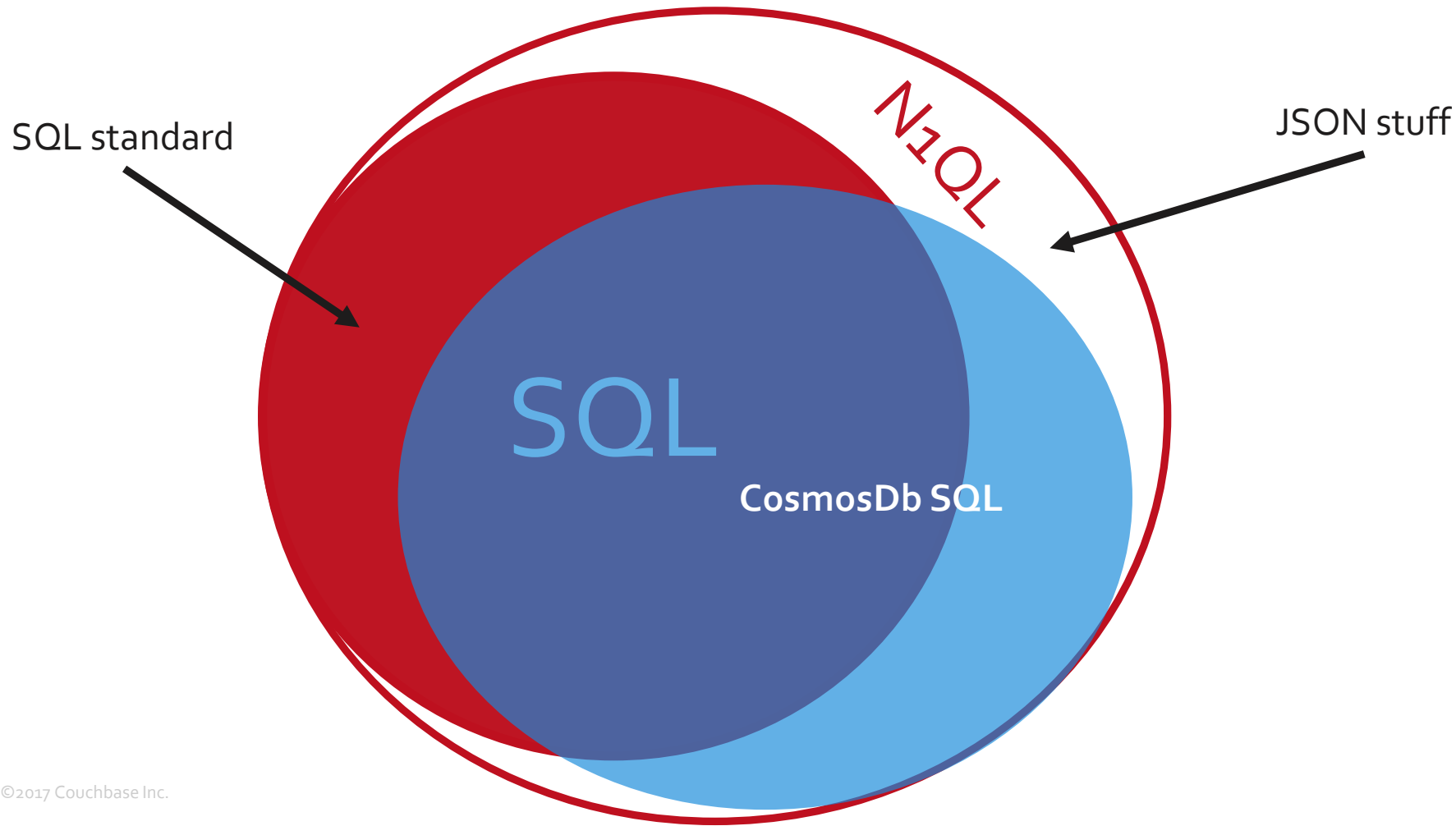
```
1 SELECT food.id,  
2     food.description,  
3     food.tags,  
4     food.foodGroup  
5 FROM food  
6 WHERE food.foodGroup = "Snacks" and food.id = "19015"
```

Run it!



Results < 1 / 1 > | Sample Docs

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





Demo: N1QL

Ranging over collections	<ul style="list-style-type: none">• WHERE ANY c IN children SATISFIES c.age > 10 END• WHERE EVERY r IN ratings SATISFIES r > 3 END
Mapping with filtering	<ul style="list-style-type: none">• ARRAY c.name FOR c IN children WHEN c.age > 10 END
Deep traversal, SET, and UNSET	<ul style="list-style-type: none">• WHERE ANY node WITHIN request SATISFIES node.type = "xyz" END• UPDATE doc UNSET c.field1 FOR c WITHIN doc END
Dynamic Construction	<ul style="list-style-type: none">• SELECT { "a": expr1, "b": expr2 } AS obj1, name FROM ... // Dynamic object• SELECT [a, b] FROM ... // Dynamic array
Nested traversal	<ul style="list-style-type: none">• SELECT x.y.z, a[o] FROM a.b.c ...
IS [NOT] MISSING	<ul style="list-style-type: none">• WHERE name IS MISSING



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 ("1.ABC.X382", {"O_ID":482, "O_D_ID":3, "O_W_ID":4});
```

```
UPDATE `ORDERS`
    SET O_CARRIER_ID = "ABC987"
WHERE O_ID = 482 AND O_D_ID = 3 AND O_W_ID = 4
```

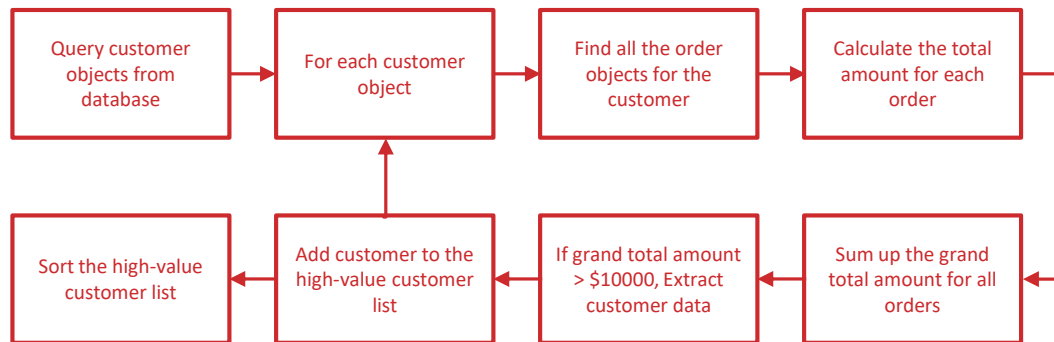
```
DELETE FROM `NEW_ORDER`
WHERE NO_D_ID = 291 AND
      NO_W_ID = 3482 AND
      NO_O_ID = 2483
```



- **CREATE INDEX ON ...**
- **DROP INDEX ...**
- **EXPLAIN ...**

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

Find High-Value Customers with Orders > \$10000



```
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	<ul style="list-style-type: none">▪ SELECT, INSERT, UPDATE, DELETE, MERGE	<ul style="list-style-type: none">▪ SELECT, INSERT, UPDATE, DELETE, MERGE
Query Operations	<ul style="list-style-type: none">▪ Select, Join, Project, Subqueries▪ Strict Schema▪ Strict Type checking	<ul style="list-style-type: none">▪ Select, Join, Project, Subqueries✓ Nest & Unnest✓ Look Ma! No Type Mismatch Errors!▪ JSON keys act as columns
Schema	<ul style="list-style-type: none">▪ Predetermined Columns	<ul style="list-style-type: none">✓ Fully addressable JSON✓ Flexible document structure
Data Types	<ul style="list-style-type: none">▪ SQL Data types▪ Conversion Functions	<ul style="list-style-type: none">▪ JSON Data types▪ Conversion Functions
Query Processing	<ul style="list-style-type: none">▪ INPUT: Sets of Tuples▪ OUTPUT: Set of Tuples	<ul style="list-style-type: none">▪ 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)



UC San Diego

SQL++



Couchbase

N1QL

<http://tinyurl.com/UCSDsql>

Couchbase Analytics (CBAS) and SQL++



Couchbase - Analytics Workbench

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

Execute

← History (1/1) →

Load Query

Save Query

1 SELECT "Hello, beer!" AS greeting;>

JSON Table Tree

Results

Save JSON

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

1- {
2 "requestID": "d93e20fd-8a92-46a6-9798-328d7c96bb56",
3 "clientContextID": "a37f960a-f84f-4544-8702-65d48595274d",
4 "signature": "*",
5- "results": [{
6 "greeting": "Hello, beer!"
7 }]
8 ,
9 "status": "success",
10- "metrics": {
11 "elapsedTime": "1.253861973s",
12 "executionTime": "1.164601686s",
13 "resultCount": "1",
14 "resultSize": "35"
15 }
16 }
17 }

SQL EMPIRE STRIKES BACK


Industry is Choosing SQL to Query JSON




PostgreSQL



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

	Column Name	Data Type	Allow Nulls
	Id	uniqueidentifier	<input type="checkbox"/>
	SpeakingInfo	nvarchar(MAX)	<input type="checkbox"/>




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

100 %

Results Messages

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





Ecosystem



Couchbase

+



looker



pentaho

informatica

Metanautix

Qlik

tableau[®]
S O F T W A R E

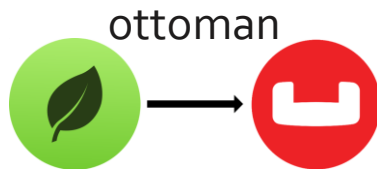
PROGRESS DataDirect



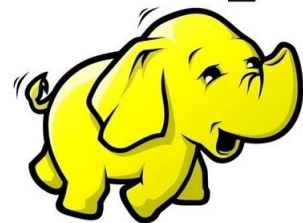
Couchbase



elastic



hadoop



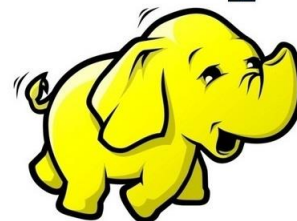


Azure Cosmos DB



Gremlin
 $G = (V, E)$

hadoop





<http://query.pub.couchbase.com/tutorial/>

<https://www.documentdb.com/sql/demo>



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

Couchbase, everybody!



Couchbase



Where do you find us?

- blog.couchbase.com
- [@couchbasedev](https://twitter.com/couchbasedev)
- [@mgroves](https://twitter.com/mgroves)



Couchbase
DEVELOPER COMMUNITY



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?](#)



 **Couchbase**





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



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



No.