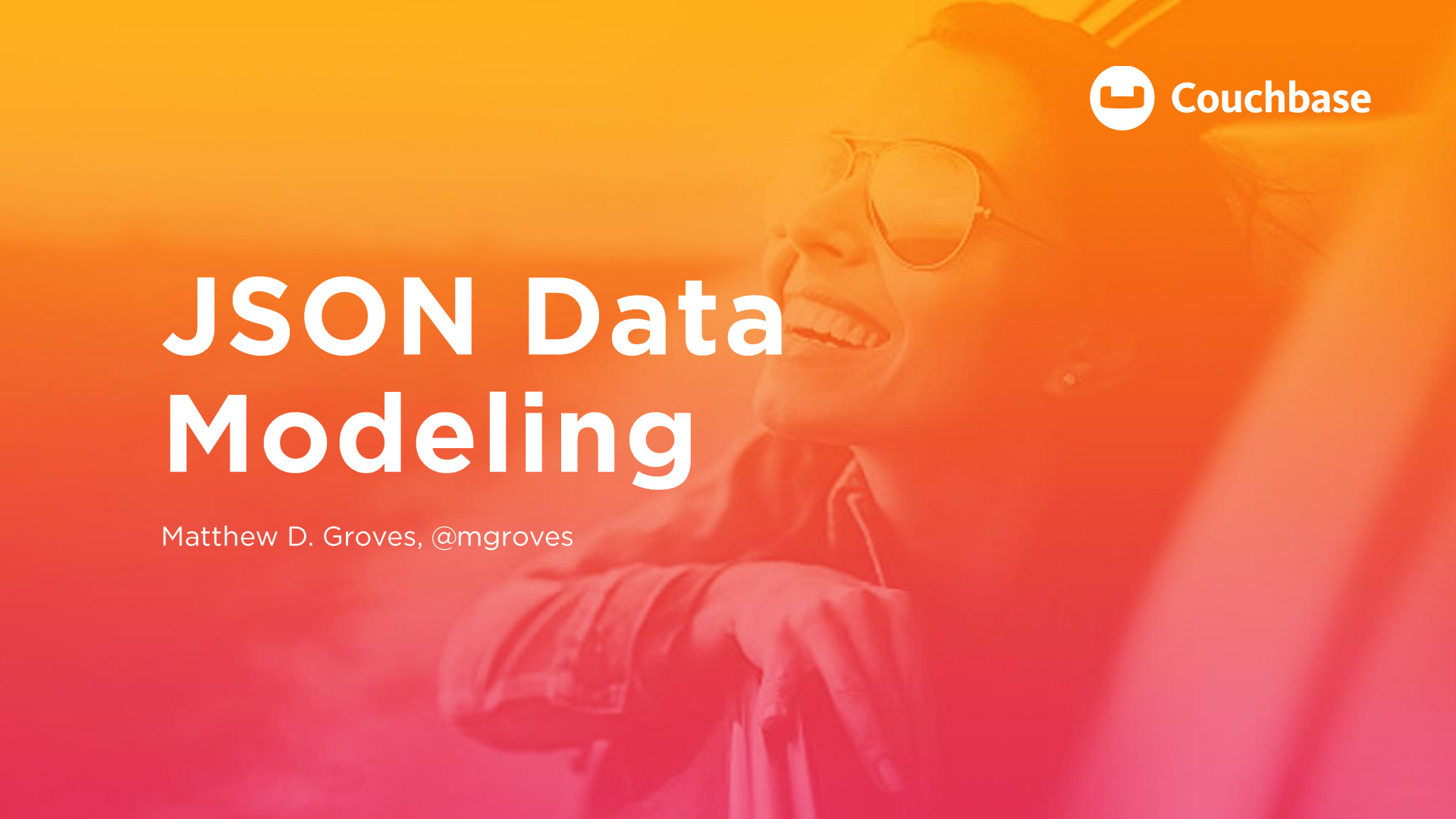


Couchbase

JSON Data Modeling

Matthew D. Groves, @mgroves

CincyDeliver 2019 Sponsors

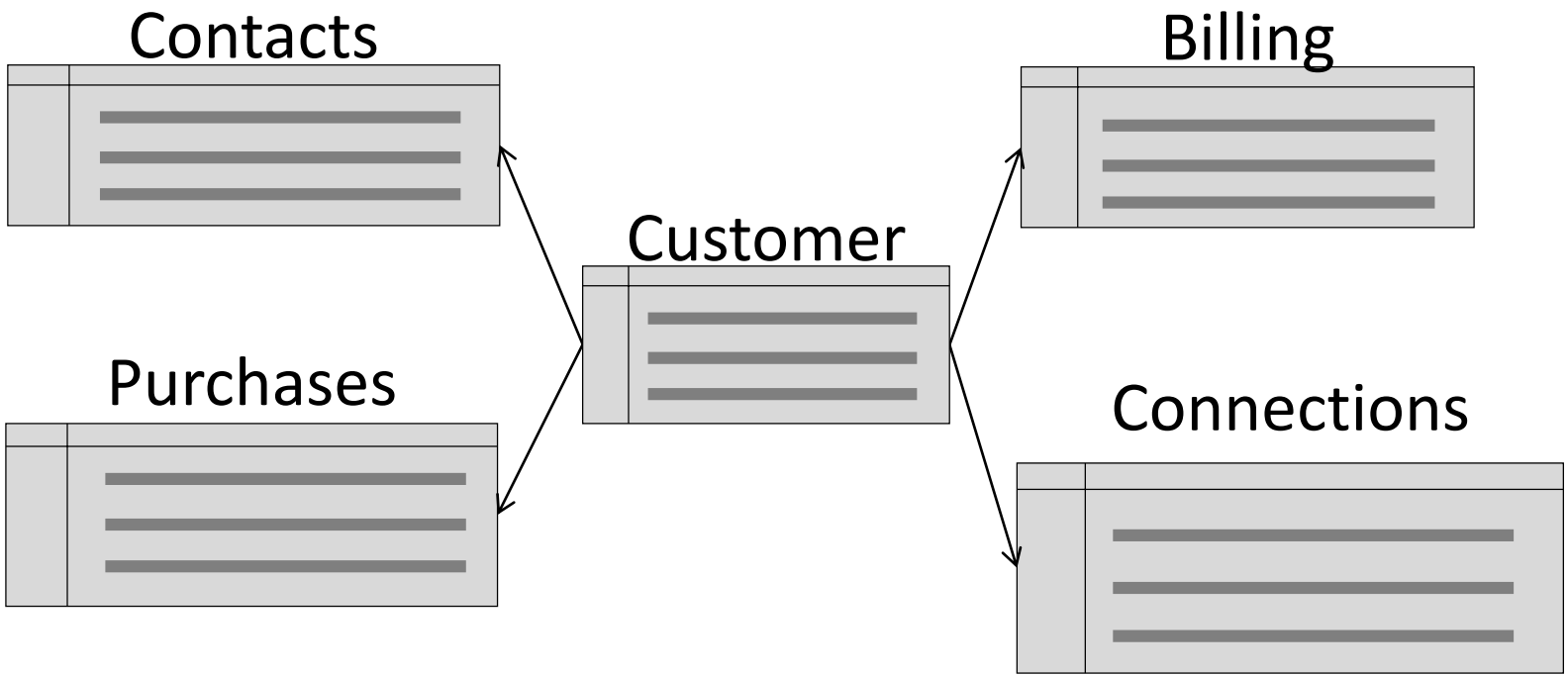
CINCYDELIVER.ORG

Diamond

Platinum

Gold

Modeling Data in a Relational World



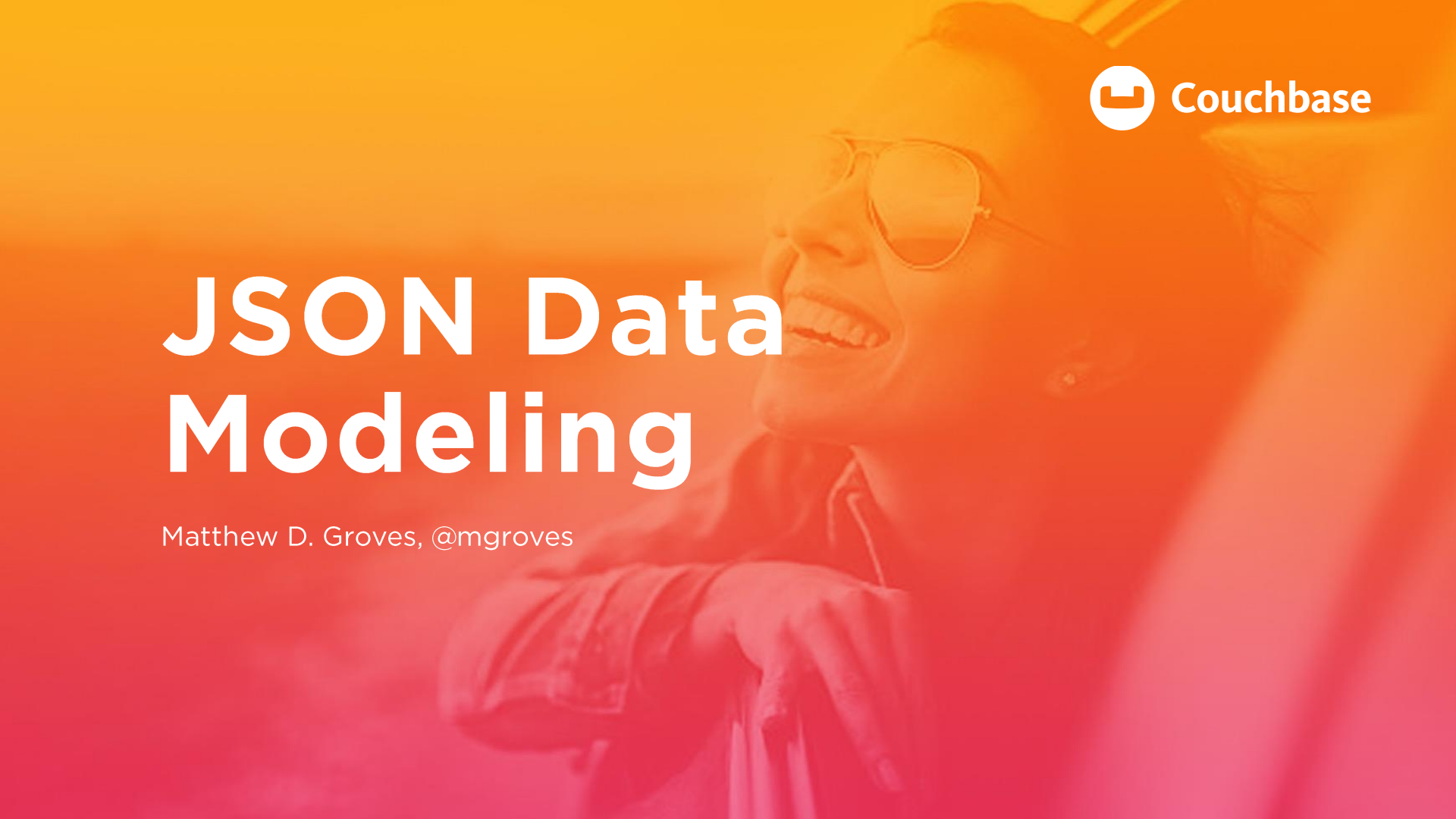
Where am I?

- Cincy Deliver
- https://www.cincydeliver.org/

Who am I?

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

by @natelovett



Couchbase

JSON Data Modeling

Matthew D. Groves, @mgroves

AGENDA

01/ Why NoSQL?

02/ JSON Data Modeling

03/ Accessing Data

04/ Migrating Data

05/ Summary / Q&A

1 | Why NoSQL?

NoSQL Landscape

Document
- Couchbase
- MongoDB
- DynamoDB
- CosmosDB

Key-Value
- Couchbase
- Riak
- BerkeleyDB
- Redis

Graph
- OrientDB
- Neo4J
- DEX
- GraphBase

Wide Column
- Hbase
- Cassandra
- Hypertable

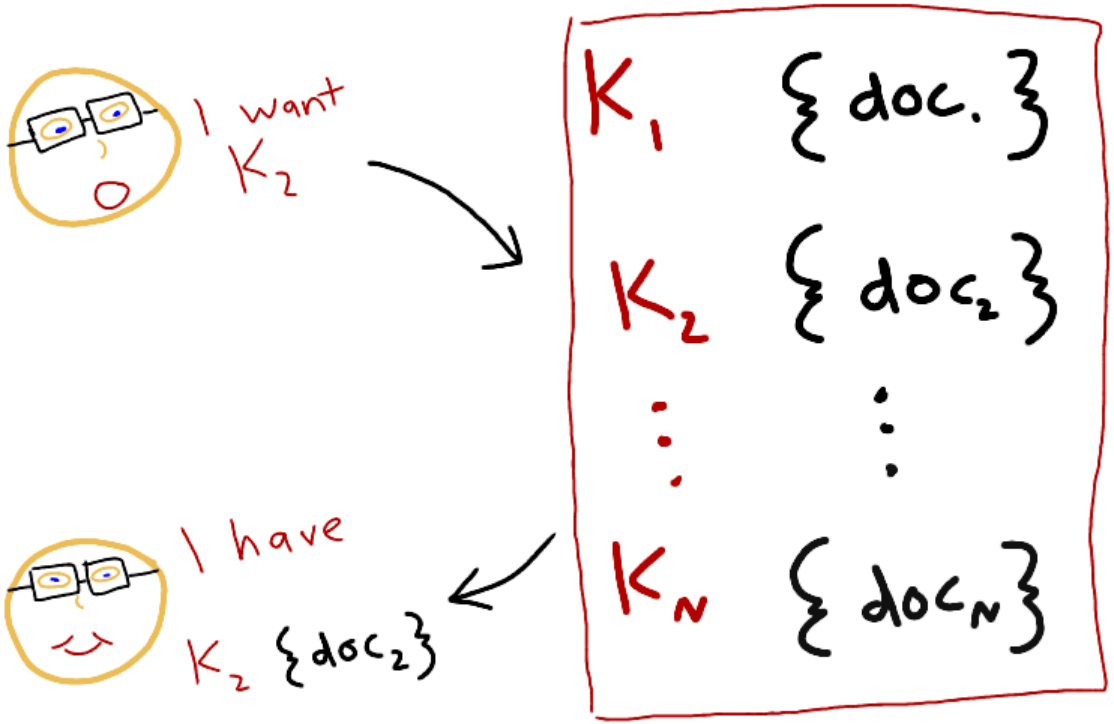
NoSQL Landscape

Document

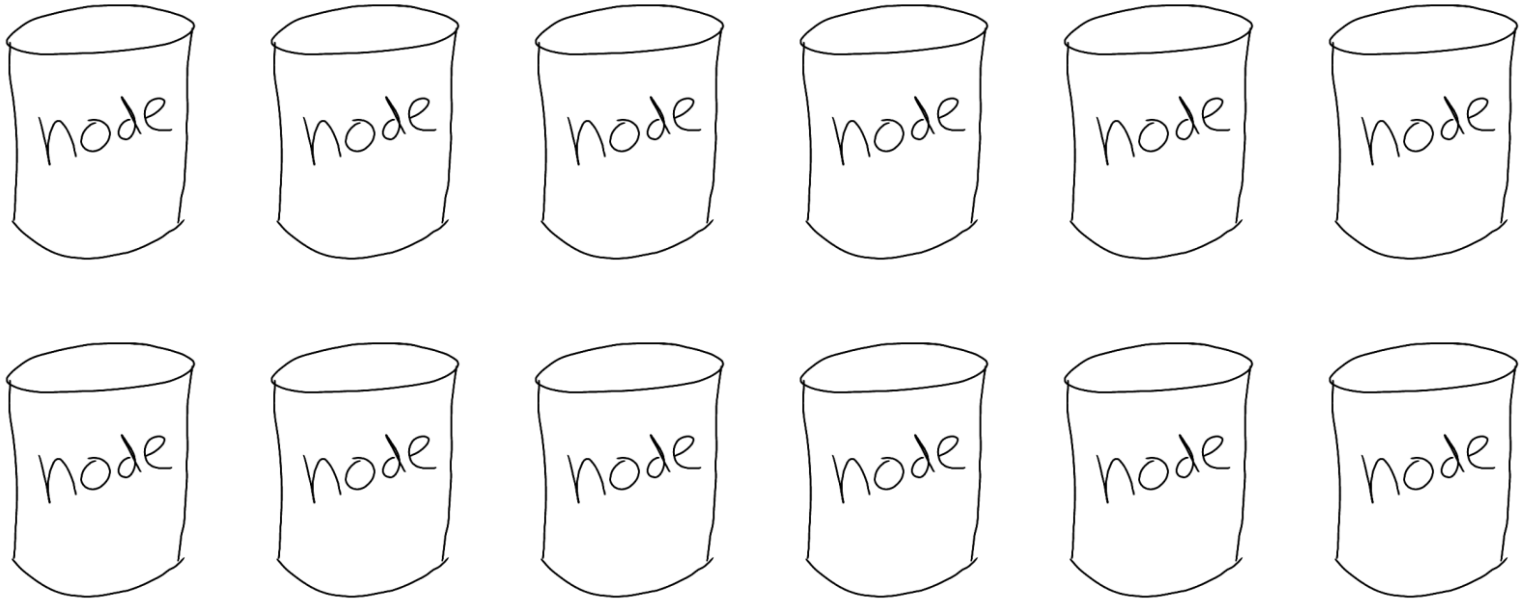
- Couchbase
- MongoDB
- DynamoDB
- CosmosDB

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

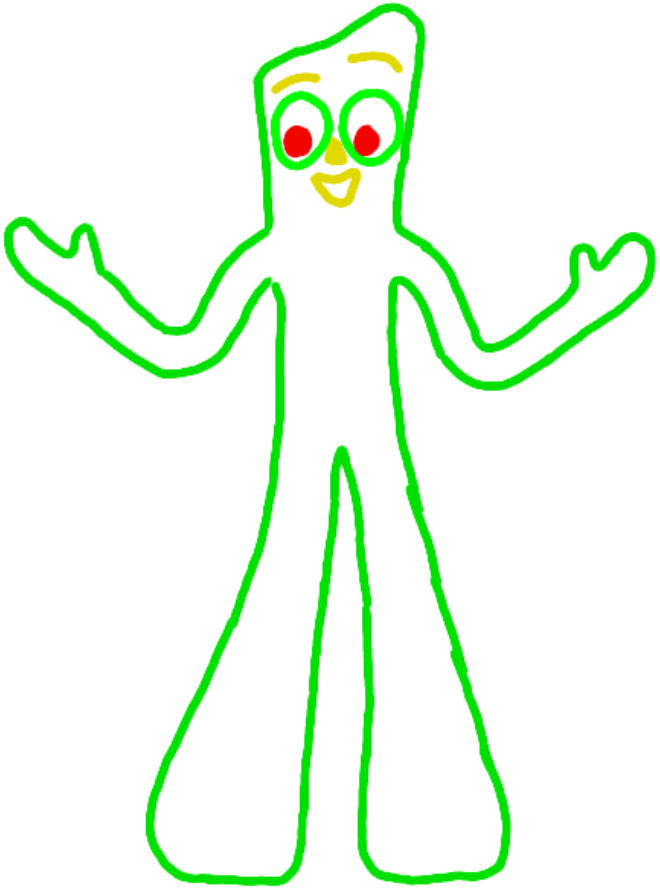
What's NoSQL?



Why NoSQL? Scalability



Why NoSQL? Flexibility



Why NoSQL? Availability

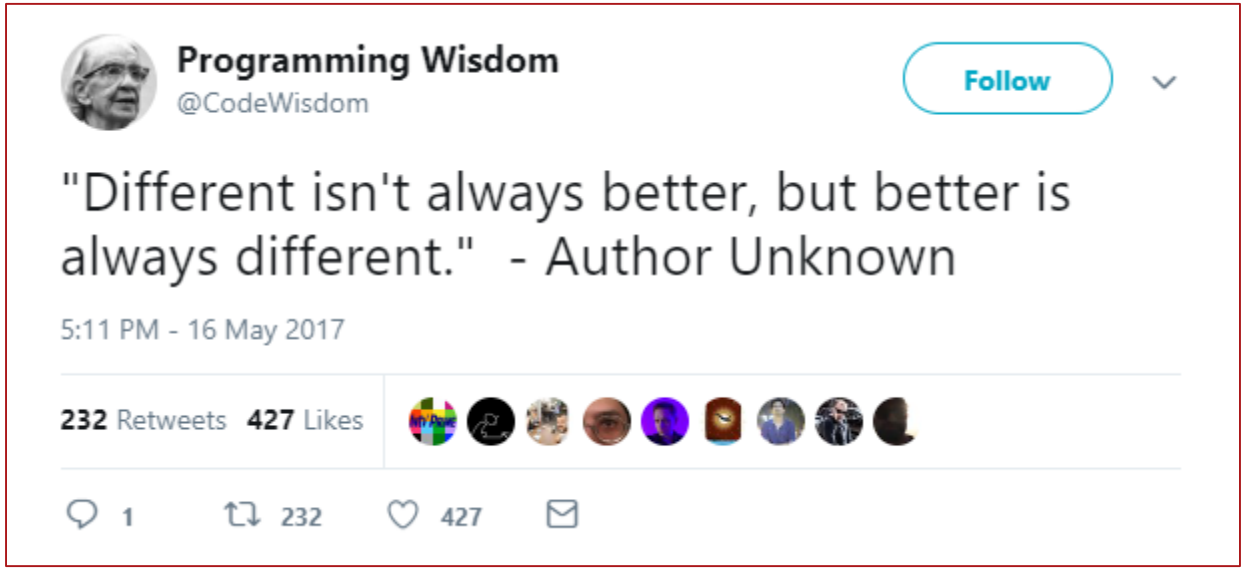
Why NoSQL? Performance

Use Cases for NoSQL

- Caching
- Session
- User profile
- Catalog
- Content management
- Personalization
- Customer 360
- IoT
- Communication
- Gaming
- Advertising
- Travel booking
- Loyalty programs
- Fraud monitoring
- Social media
- Finance

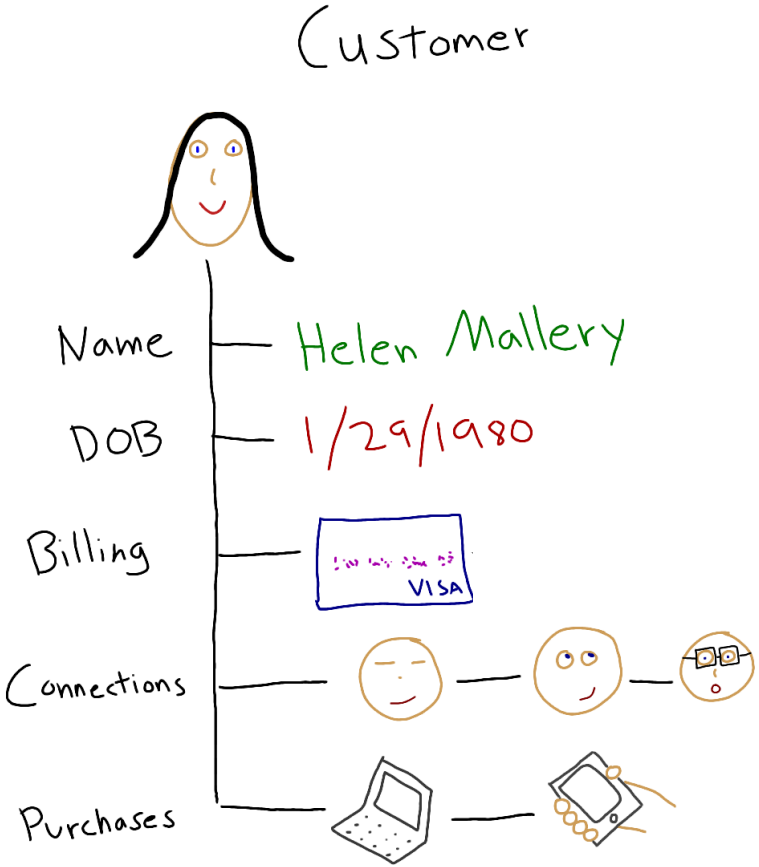
https://www.couchbase.com/customers

Use Cases


Programming Wisdom
@CodeWisdom
Follow
"Different isn't always better, but better is always different." - Author Unknown
5:11 PM - 16 May 2017
232 Retweets 427 Likes
1
232
427

2 | JSON Data Modeling

Properties of Real-World Data



Modeling Data in a Relational World

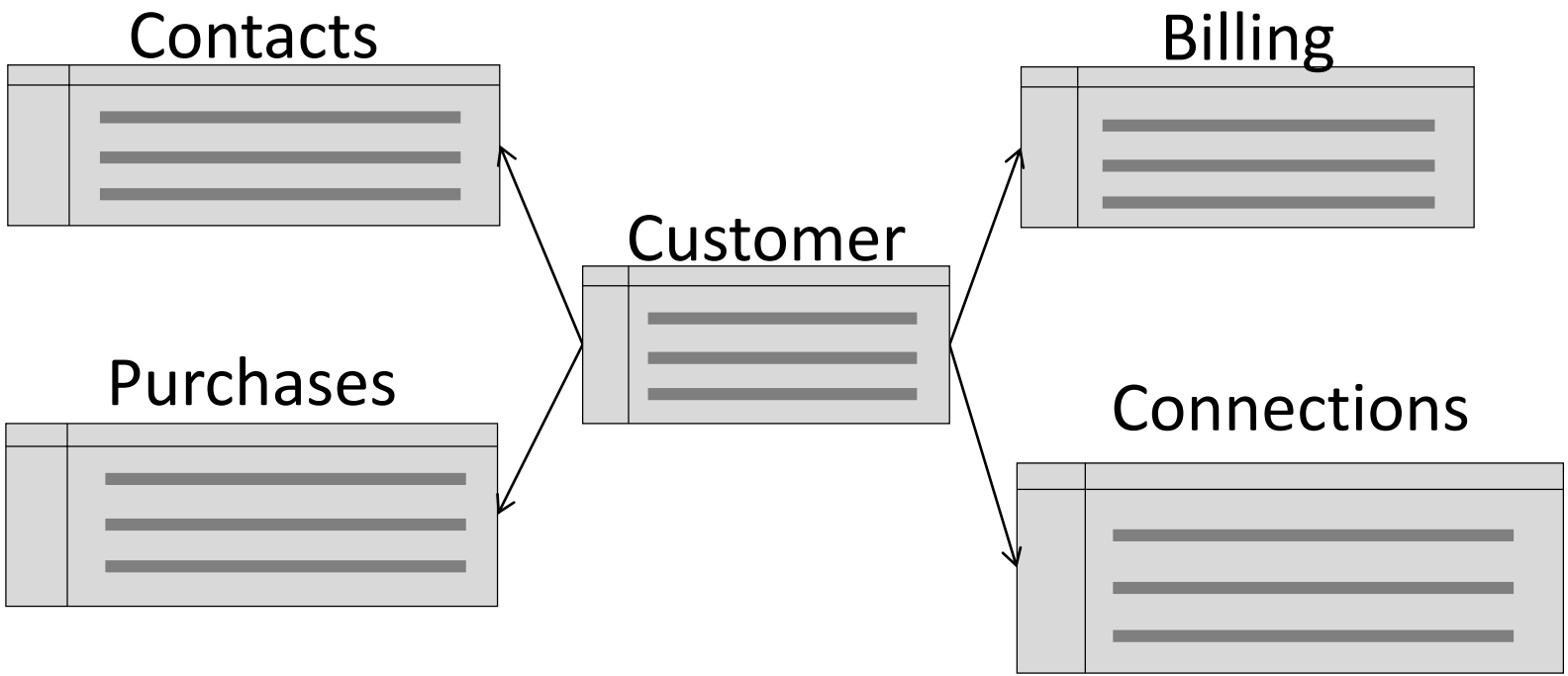


Table: Customer

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	Relation
CBL2015	XYZ987	Brother
CBL2015	SKR007	Father

Customer DocumentKey: CBL2015


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

Contacts

CustomerID	ConnId	Name
CBL2015	XYZ987	Joe Smith
CBL2015	SKR007	Sam Smith

Customer

Customer ID	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	Relation
CBL2015	XYZ987	Brother
CBL2015	SKR007	Father

DocumentKey: **CBL2015**

```
{
   "Name" : "Jane Smith",
   "DOB"  : "1990-01-30",
   "cardnum" : "5827-2842…",
   "expiry" : "2019-03",
   "cardType" : "visa",
   "Connections" : [
      {
         "CustId"   : "XYZ987",
         "Relation"    : "Brother"
      },
      {
         "CustId"   : "SKR007",
         " Relation "    : "Father"
      }
    ],
   "Purchases" : [
      { "id":12, item: "mac",   "amt": 2823.52 }
      { "id":19, item: "ipad2", "amt": 623.52 }
   ]
}
```

Contacts

CustomerID	ConnId	Name
CBL2016	XYZ987	Joe Smith
CBL2016	SKR007	Sam Smith

Billing

CustomerID	Type	Cardnum	Expiry
CBL2016	visa	5927…	2020-03
CBL2016	master	6273…	2019-11

Customer

CustomerID	Name	DOB
CBL2016	Bob Jones	1980-01-29

Purchases

CustomerID	item	amt
CBL2016	mac	2823.52
CBL2016	ipad2	623.52

Connections

CustomerID	ConnId	Relation
CBL2016	XYZ987	Brother
CBL2016	SKR007	Father

DocumentKey: **CBL2016**


```
{
   "Name" : "Bob Jones",
   "DOB"  : "1980-01-29",
   "Billing" : [
      {
         "type"    : "visa",
         "cardnum" : "5927-2842-2847-3909",
         "expiry"  : "2020-03"
      },
      {
         "type"    : "master",
         "cardnum" : "6273-2842-2847-3909",
         "expiry"  : "2019-11"
      }
   ],
   "Connections" : [
      {
         "CustId"   : "XYZ987",
         "Relation"    : "Brother"
      },
      {
         "CustId"   : "PQR823",
         "Relation"    : "Father"
      }
      ],
     "Purchases" : [
      { "id":12, item: "mac",   "amt": 2823.52 }
     { "id":19, item: "ipad2", "amt": 623.52 }
   ]
}
```

Modeling your data: Strategies / rules of thumb

Relationship is **one-to-one** or **one-to-many**

Store related data as **nested objects**

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

Modeling your data: Strategies / rules of thumb

Relationship is **many-to-one** or **many-to-many**

Store related data as **separate documents**

```
{
   "Name" : "Jane Smith",
   "DOB"  : "1990-01-30",
   "Connections" : [
         "XYZ987",
         "PQR823",
         "PQR828"
     ]
}
```

Modeling tools

- Hackolade
- Erwin DM NoSQL
- Idera ER/Studio

3 | Accessing Data

Modeling your data: Strategies / rules of thumb

Data reads are mostly **parent fields**

Store children as **separate documents**

```
{
   "Name" : "Jane Smith",
   "DOB"  : "1990-01-30",
   "Connections" : [
            "XYZ987",
            "PQR823",
            "PQR828"
      ]
}
```

Modeling your data: Strategies / rules of thumb

Data reads are mostly **parent + child fields**

Store children as **nested objects**

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

Modeling your data: Strategies / rules of thumb

Data writes are mostly **parent *or* child** (not both)

Store children as **separate documents**

```
{
   "Name" : "Jane Smith",
   "DOB"  : "1990-01-30",
   "Connections" : [
            "XYZ987",
            "PQR823",
            "PQR828"
      ]
}
```

Modeling your data: Strategies / rules of thumb

Data writes are mostly **parent and child** (both)

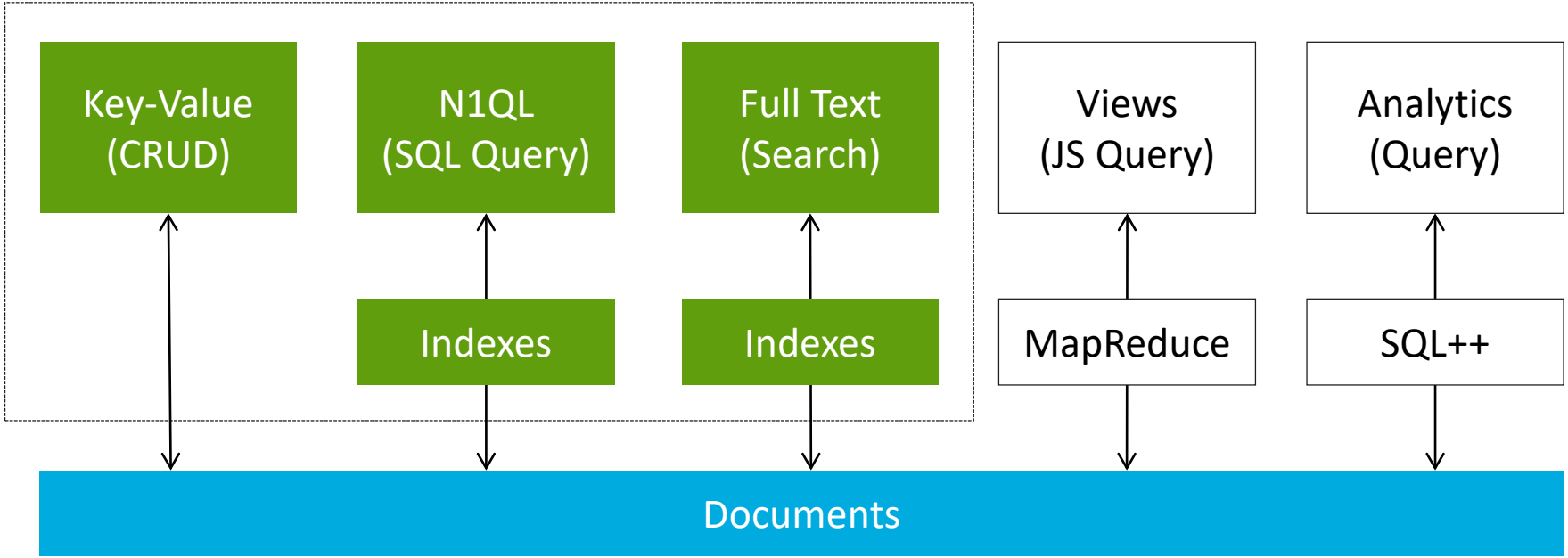
Store children as **nested objects**

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

Modeling your data: Strategies / rules of thumb

If …	Then …
Relationship is **one-to-one** or **one-to-many**	Store related data as **nested objects**
Relationship is **many-to-one** or **many-to-many**	Store related data as **separate documents**
Data reads are mostly **parent fields**	Store children as **separate documents**
Data reads are mostly **parent + child fields**	Store children as **nested objects**
Data writes are mostly **parent *or* child** (not both)	Store children as **separate documents**
Data writes are mostly **parent and child** (both)	Store children as **nested objects**

Accessing your data (Couchbase)



Key/Value

```
public ShoppingCart GetCartById(string id)
{
    return _bucket.Get<ShoppingCart>(id).Value;
}
```

```
public void CreateShoppingCart()
{
    _bucket.Insert(new Document<ShoppingCart>
    {
        Id = "shopping-cart-1",
        Content = new ShoppingCart { . . . }
    });
}
```

Subdocument access

```
{
   "username": "mgroves",
   "profile": {
      "phoneNumber": "123-456-7890",
      "address": {
         "street": "123 main st",
         "city": "Grove City",
         "state": "Ohio"
      }
   }
}
```

Key/Value: Recommendations for keys

- Natural Keys
- Human Readable
- Deterministic
- Semantic

Key/Value: Example keys

- author::matt
- author::matt::blogs
- blog::csharp_8_features
- blog::csharp_8_features::comments

N1QL

OPTION ONE: *RELATED DATA IS REFERENCED*

Get all Platinum Users

```
SELECT      firstName, lastName
FROM        users
WHERE       status = “Platinum”;
```

Get all users with a Visa or MasterCard account

```
SELECT      firstName, lastName
FROM        users u INNER JOIN accounts a
ON KEYS     u.accounts
WHERE       a.type = “Visa” OR
            a.type = “MasterCard”;
```

Get all users with a billing address in California

```
SELECT      firstName, lastName
FROM        users u INNER JOIN addresses a
ON KEYS     u.addresses.shipping
WHERE       a.state = “CA”;
```

Understanding your Query Plan

```
1  SELECT t.*
2  FROM `travel-sample` t
3  WHERE t.type = 'airline'
4  AND t.name LIKE '%US%';
```

Index Currently Used

CREATE INDEX def_type ON `travel-sample`(`type`)

Index Recommendations

CREATE INDEX adv_name_type ON `travel-sample`(`name`) WHERE `type` = 'airline'

Create & Build Index

Full Text Search

submarine

Search

show advanced query settings

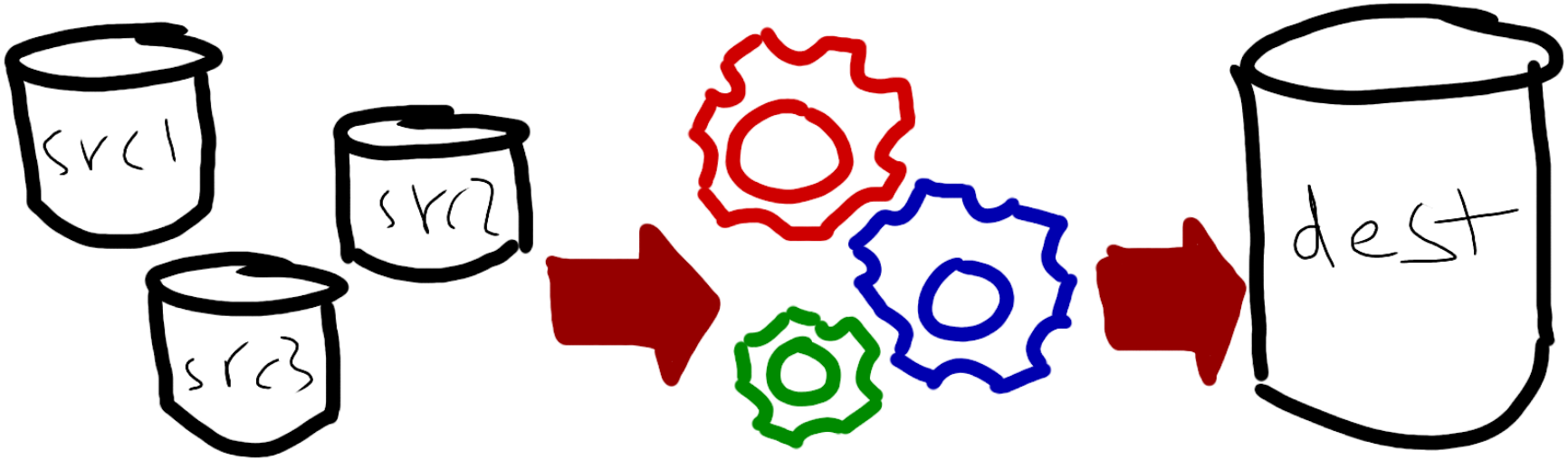
full text query syntax help

Accessing your data: Strategies and recommendation

Concept	Strategies & Recommendations
Key-Value Operations provide the best possible performance	• Create an effective key naming strategy • Create an optimized data model
Full Text Search is well-suited to text	• Facets / ranges / geography • Language aware
N1QL queries provide the most flexibility – everything else	• Query data regardless of how it is modeled • Good indexing is vital

4 | Migrating Data

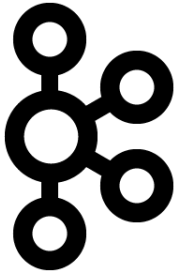
Migration options: Requirements



ETL / data cleanse / data enrichment

Migration options: Tools

Informatica™



APACHE kafka®
A distributed streaming platform

Migration options: BYO

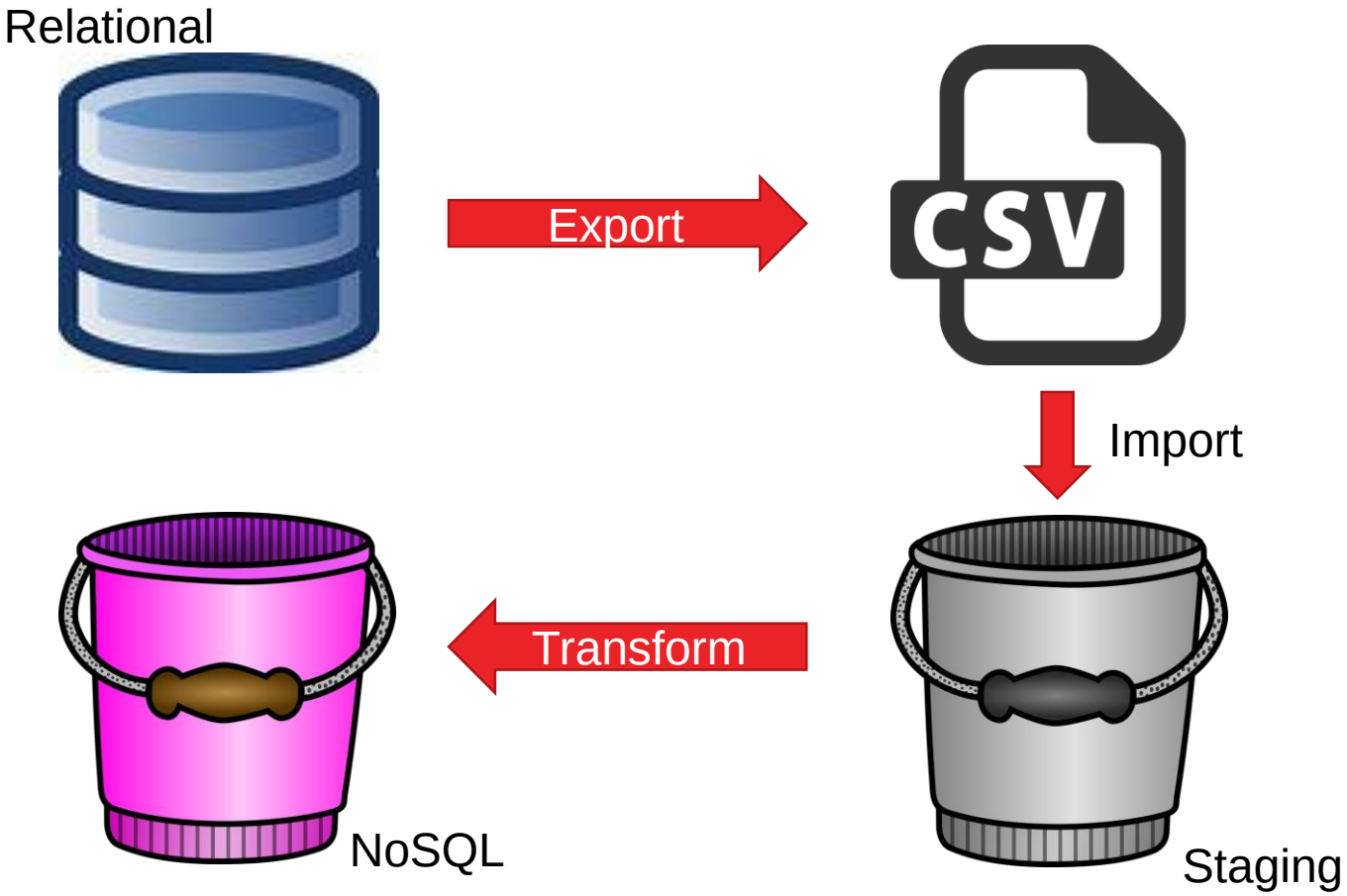
cbimport

A utility for importing data into a Couchbase cluster

mongoimport

ORACLE®
GOLDENGATE®

Migration options: KISS

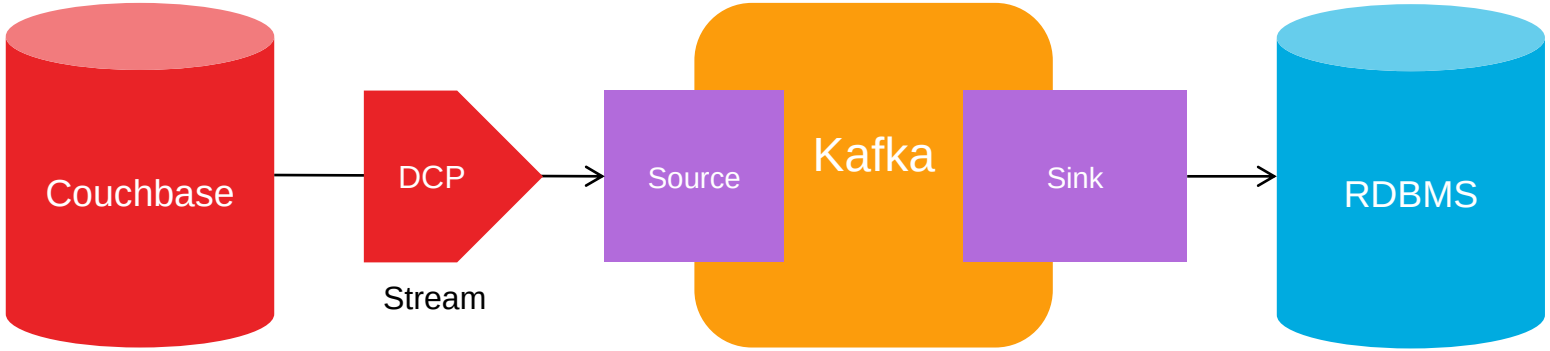


Migration Recommendations: Align

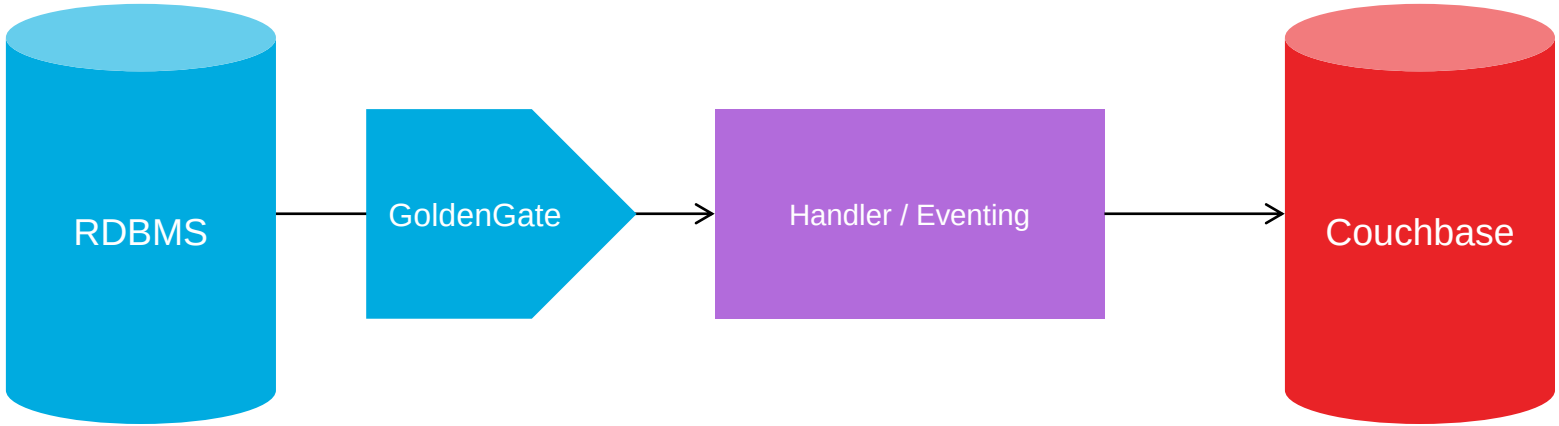
Migration Recommendations: Expect Failure

Migration Recommendations: Ensure

Sync NoSQL and relational? Automatic Replication

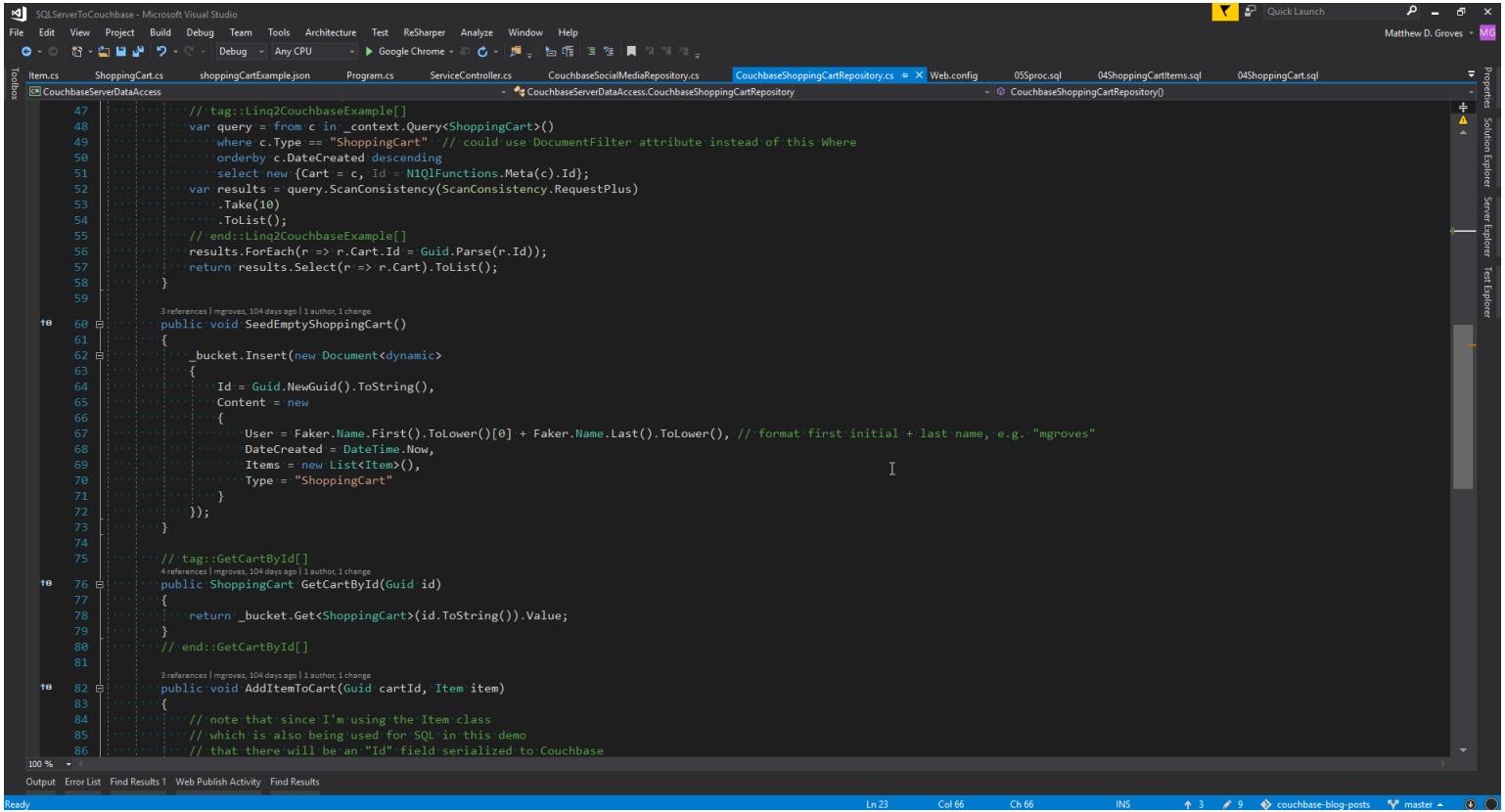


How can you sync NoSQL and relational?



https://github.com/mahurtado/CouchbaseGoldenGateAdapter

Sync NoSQL and relational? Manual.



```
// tag::Linq2CouchbaseExample[]
var query = from c in _context.Query<ShoppingCart>()
        where c.Type == "ShoppingCart" // could use DocumentFilter attribute instead of this Where
        orderby c.DateCreated descending
        select new {Cart = c, Id = N1QlFunctions.Meta(c).Id};
var results = query.ScanConsistency(ScanConsistency.RequestPlus)
        .Take(10)
        .ToList();
// end::Linq2CouchbaseExample[]
results.ForEach(r => r.Cart.Id = Guid.Parse(r.Id));
return results.Select(r => r.Cart).ToList();
}

3 references | mgroves, 104 days ago | 1 author, 1 change
public void SeedEmptyShoppingCart()
{
    _bucket.Insert(new Document<dynamic>
    {
        Id = Guid.NewGuid().ToString(),
        Content = new
        {
            User = Faker.Name.First().ToLower()[0] + Faker.Name.Last().ToLower(), // format first initial + last name, e.g. "mgroves"
            DateCreated = DateTime.Now,
            Items = new List<Item>(),
            Type = "ShoppingCart"
        }
    });
}

// tag::GetCartById[]
4 references | mgroves, 104 days ago | 1 author, 1 change
public ShoppingCart GetCartById(Guid id)
{
    return _bucket.Get<ShoppingCart>(id.ToString()).Value;
}
// end::GetCartById[]

3 references | mgroves, 104 days ago | 1 author, 1 change
public void AddItemToCart(Guid cartId, Item item)
{
    // note that since I'm using the Item class
    // which is also being used for SQL in this demo
    // that there will be an "Id" field serialized to Couchbase
```

5 | Summary

Pick the right application

Proof of Concept

Summary

Match the data access method to requirements

Resources: Blog posts

https://blog.couchbase.com/proof-of-concept-move-relational/

https://blog.couchbase.com/json-data-modeling-rdbms-users/

Resources: Me!

- @mgroves
- twitch.tv/matthewdgroves
- forums.couchbase.com

Frequently Asked Questions

1. How is Couchbase different than Mongo?
2. Is Couchbase the same thing as CouchDb?
3. How tall are you? Do you play basketball?
4. What is the Couchbase licensing situation?
5. Is Couchbase a Managed Cloud Service (DBaaS)?

Managed Cloud Server (DBaaS)

Couchbase marries the world's most powerful NoSQL technology with the capabilities and expertise of Rackspace, the number one managed cloud provider, to deliver one solution for a unified customer experience.

Google Cloud Platform

Run Couchbase Managed Cloud in any of the three major public cloud platforms

< Back

MongoDB vs Couchbase

Couchbase

- Architecture
  - Memory first architecture
  - Master-master architecture
  - Auto-sharding
- Features
  - SQL (N1QL)
  - Full Text Search
  - Analytics (NoETL)

< Back

Licensing

Couchbase Server Community

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

Couchbase Server Enterprise

- Source code is 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

CouchDB and Couchbase

memcached

< Back