



# From SQL to NoSQL A Migration Path

Using the petclinic app and Apache Cassandra on Astra

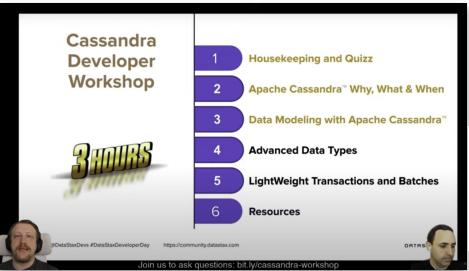


LEVEL  
**UP**  
with the

DataStax  
**Developers**

# Housekeeping

**Livestream:** youtube.com/DataStaxDevs



**YouTube**

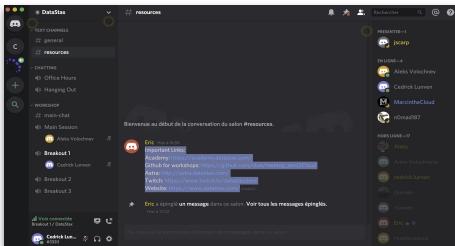


**Twitch**

**Runtime:** dtsx.io/workshop

DataStax  
**Astra**

**Questions:** bit.ly/cassandra-workshop



**Discord**

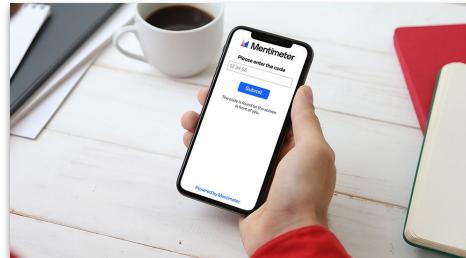


**YouTube**



Available on the iPhone  
**App Store**

**Quizz:** menti.com



**Menti.com**



GET IT ON  
**Google play**

# Weekly Workshops

<https://www.datastax.com/workshops>

The screenshot shows the DataStax Developers YouTube channel homepage. The channel has 8.1k subscribers. It features a large 'Subscribe' button in the center. Below it, there's a grid of video thumbnails for various workshops, including 'Microservices with Cassandra + Spring', 'Advanced Data Modeling in Apache Cassandra™', and 'Apache Cassandra™ Certification Preparation'. To the right, there are sections for 'Upcoming Live Events' and 'Cloud-Native Workshop: Build Spring Microservices with Apache Cassandra™'. The overall theme is 'LEVEL UP with the DataStax Developers'.

**LEVEL UP with the DataStax Developers**

DataStax Developers  
8,1 k abonnés

ACCUEIL VIDÉOS PLAYLISTS COMMUNAUTÉ

Vidéos mises en ligne ▾ TOUT REGARDER

Microservices with Cassandra + Spring (2:23:56)  
Advanced Data Modeling in Apache Cassandra™ (2:41:51)

Building Microservices with Cassandra + Spring (1,1 k vues · Diffusé il y a 20 heures)

Advanced Data Modeling in Apache Cassandra™ (1,3 k vues · Diffusé il y a 1 semaine)

Polska DataOps Meetup: Introduction to Apache... (151 vues · Diffusé il y a 1 semaine)

Bring Streaming to Cassandra with Apache... (703 vues · Diffusé il y a 2 semaines)

Apache Cassandra™ Certification Preparation (Introduction Cassandra™) (1,6 k vues · Diffusé il y a 2 semaines)

Taking your K8s app to the Cloud! (1:56:10)  
RESTful Data Access from a Spring Boot App (5:29)

Example of the week Astra Spring REST (1:07:39)  
Ask Me Anything Hackathon Special (INTERMEDIATE TOPIC)

DataStax HOLIDAY HACKATHON (ASK US ANYTHING) (INTERMEDIATE TOPIC)

Connecting Cassandra & Kubernetes (1:42:17)  
An Easy Backend API with Apache Cassandra™ (BEGINNER TOPIC)

An Easy Backend API with Apache Cassandra™ (667 vues · Diffusé il y a 1 mois)

Learn how to build a Serverless Game! (Feb 24 or Feb 25 | Game Development | Beginner)

START

Cloud-Native Workshop: Build a serverless game with the JAMStack! (THU MAR 11 2021)

Build Microservices with Cassandra & Quarkus (March 11 | Microservices | Beginner)

Cloud-Native Workshop: Build Microservices w/ Apache Cassandra™ + Quarkus! (THU MAR 11 2021)

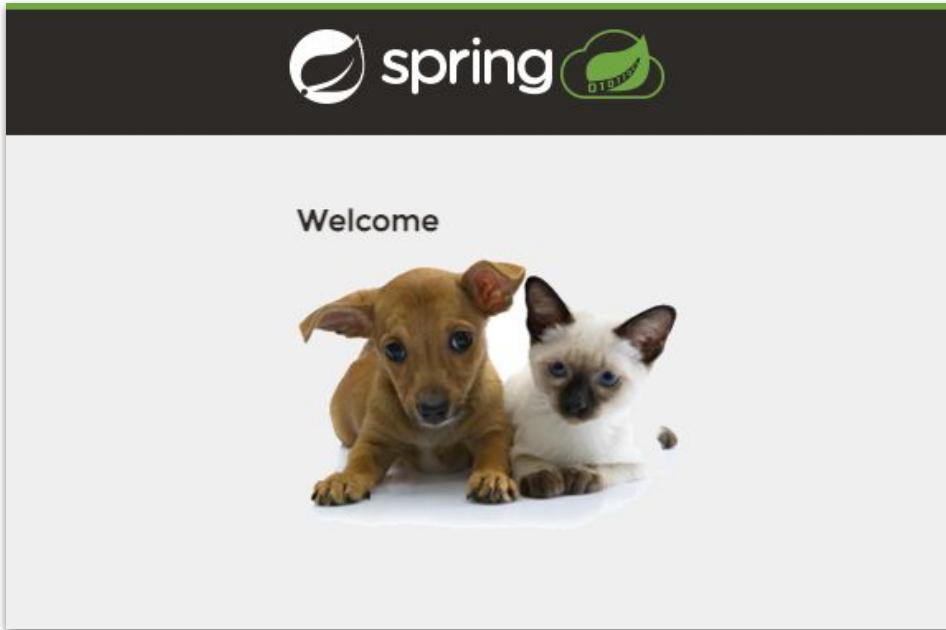
Register Now

Register Now

Register Now

# Housekeeping

App: [spring-petclinic.github.io](http://spring-petclinic.github.io)



# Achievement Unlocked!

[dtsx.io/badges](https://dtsx.io/badges)



**K8ssandra Workshop**

Awarded to **Sylwester Lachiewicz** • slachiewicz@gmail.com  
Issued on Mar 5, 2021

Offered By DataStax Developers

Upgrade Complete! This badge is to certify successful completion of the DataStax K8ssandra Workshop: "Running Apache Cassandra on Kubernetes".

 **Verified**  
Last verified by Badgr on Mar 31, 2021

[Re-verify Badge](#)

**EARNING CRITERIA**  
Recipients must complete the earning criteria to earn this badge

To earn this badge, individuals must complete the following steps during the **K8ssandra Workshop**:

- Attend the lecture
- Complete the practical steps by doing all required exercises

[View External Criteria](#)

**TAGS**  
kubernetes  
cassandra



Aron L.  
Marcin Brzozowski  
Demre Buyuk  
Muthu Krishnan  
Arvind V.  
Parth Trambadiya  
Jasbir Singh

Prateek Jain  
Roozbeh Dargahi  
Andrey Deryabin  
Akshay Wakhare  
Pranav Anant  
Joshi  
Haris  
Juan Alonso

Santosh Nepali  
Jorge Ortiz  
Ankit Bhavsar  
Aneiliya Klevleeva  
Martin Coronel  
Govindasamy  
Ville Kerminen  
Priya Jakhar

Paul Robu  
Avinash Upadhyaya  
Włodzimierz Kozłowski  
Sharath Koushik  
Tom Rota  
Joel Reis  
Francesco Abbate

Sylwester Lachiewicz  
Ankit Bhavsar  
Jasbir Singh

**AND MANY OTHERS!**

# menti.com



Go to [www.menti.com](http://www.menti.com) and use the code 3491 9972

## Inequality predicates are allowed on ...

A bar chart titled "Inequality predicates are allowed on ...". The y-axis represents a score from 1 to 15. The x-axis categories are: "All table columns" (blue bar, value 4), "Partition key columns" (yellow bar, value 3), "clustering key columns" (green bar, value 15, marked with a green checkmark), and "No inequality predicates are allowed" (pink bar, value 1, marked with a red X). The chart is set against a background of a video conference interface showing two participants.

Big paycheck

2:10:19 / 2:26:05

Go to [www.menti.com](http://www.menti.com) and use the code 3491 9972

## Leaderboard

4821 p		spanda
4820 p		Agent X9
4775 p		Sam
4711 p		CCedrickThePresenter
4468 p		shubham
4371 p		aaa
3895 p		vignesh
3877 p		adry
3861 p		Millie
3812 p		Puggie

2:11:07 / 2:26:05

DataStax Developers 6

# HandsOn #1 Create Astra



DataStax

# Astra

**Get your instance here:**

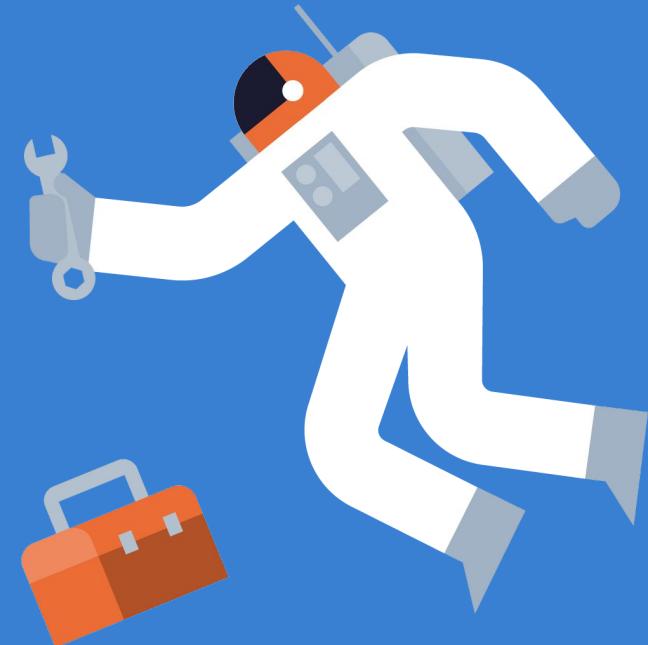
- [dtsx.io/workshop](https://dtsx.io/workshop)



# GitHub

**Repository:**

- [dtsx.io/sql-to-nosql](https://dtsx.io/sql-to-nosql)



# Agenda

**01**

Why switch from  
SQL to NoSQL?

**02**

Three migration  
approaches

**03**

Data modeling SQL  
vs NoSQL

**04**

Migrate your data  
model

**05**

Moving data with  
DSBulk

**06**

Resources

# Agenda

**01**

Why switch from  
SQL to NoSQL?

**02**

Three migration  
approaches

**03**

Data modeling SQL  
vs NoSQL

**04**

Migrate your data  
model

**05**

Moving data with  
DSBulk

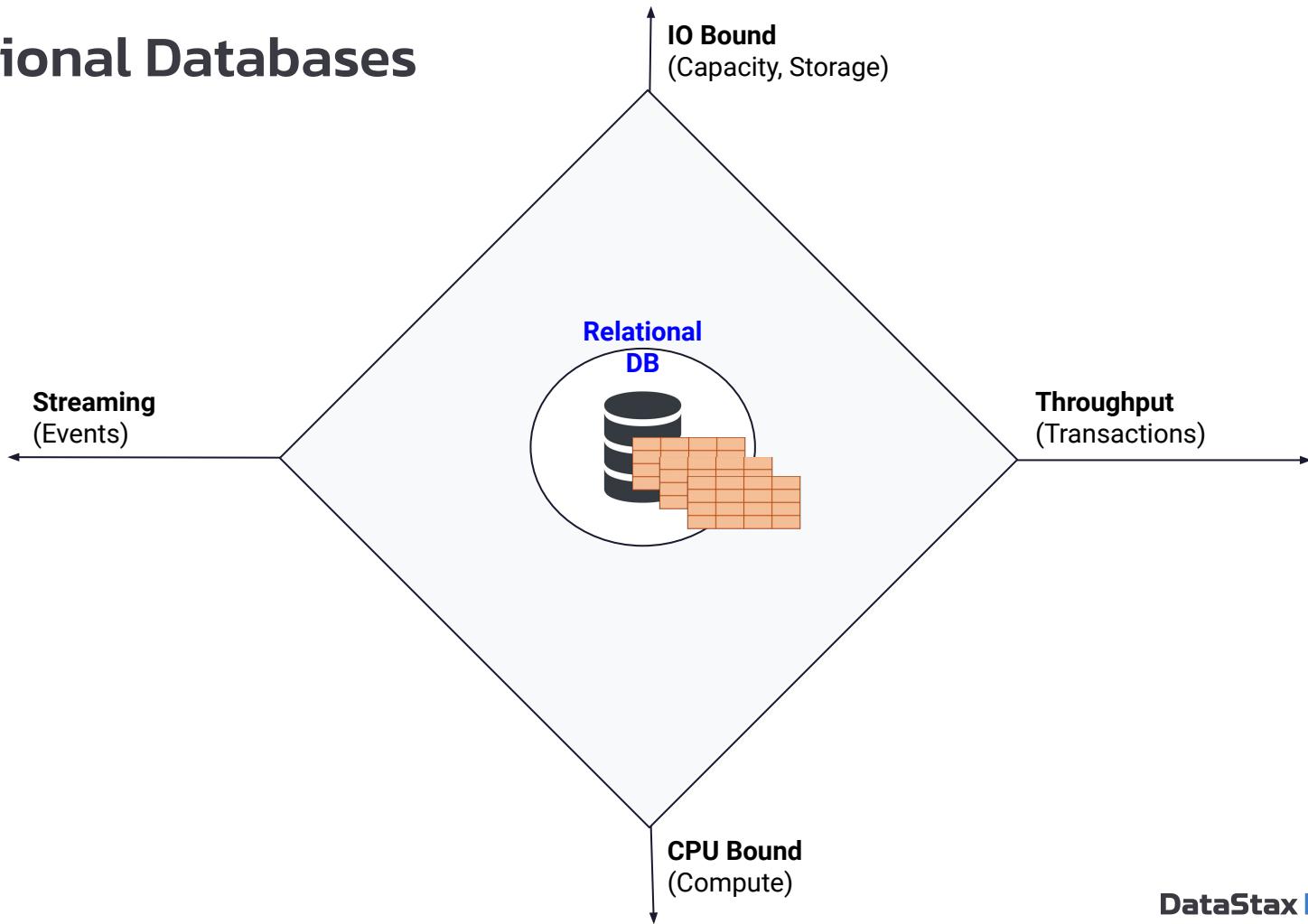
**06**

Resources

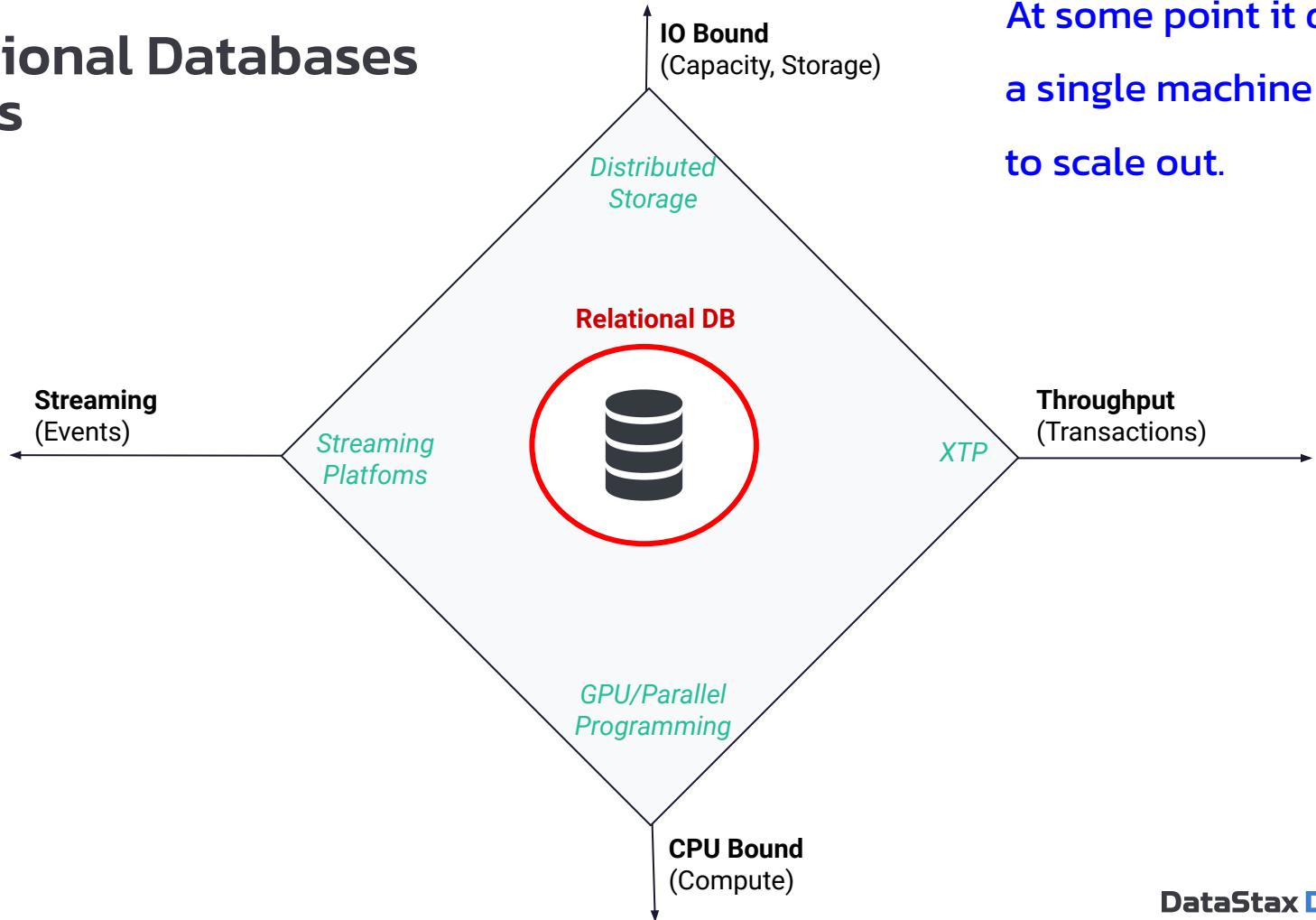
# Why switch from SQL to NoSQL?

Performance, scale, and resiliency

# Relational Databases



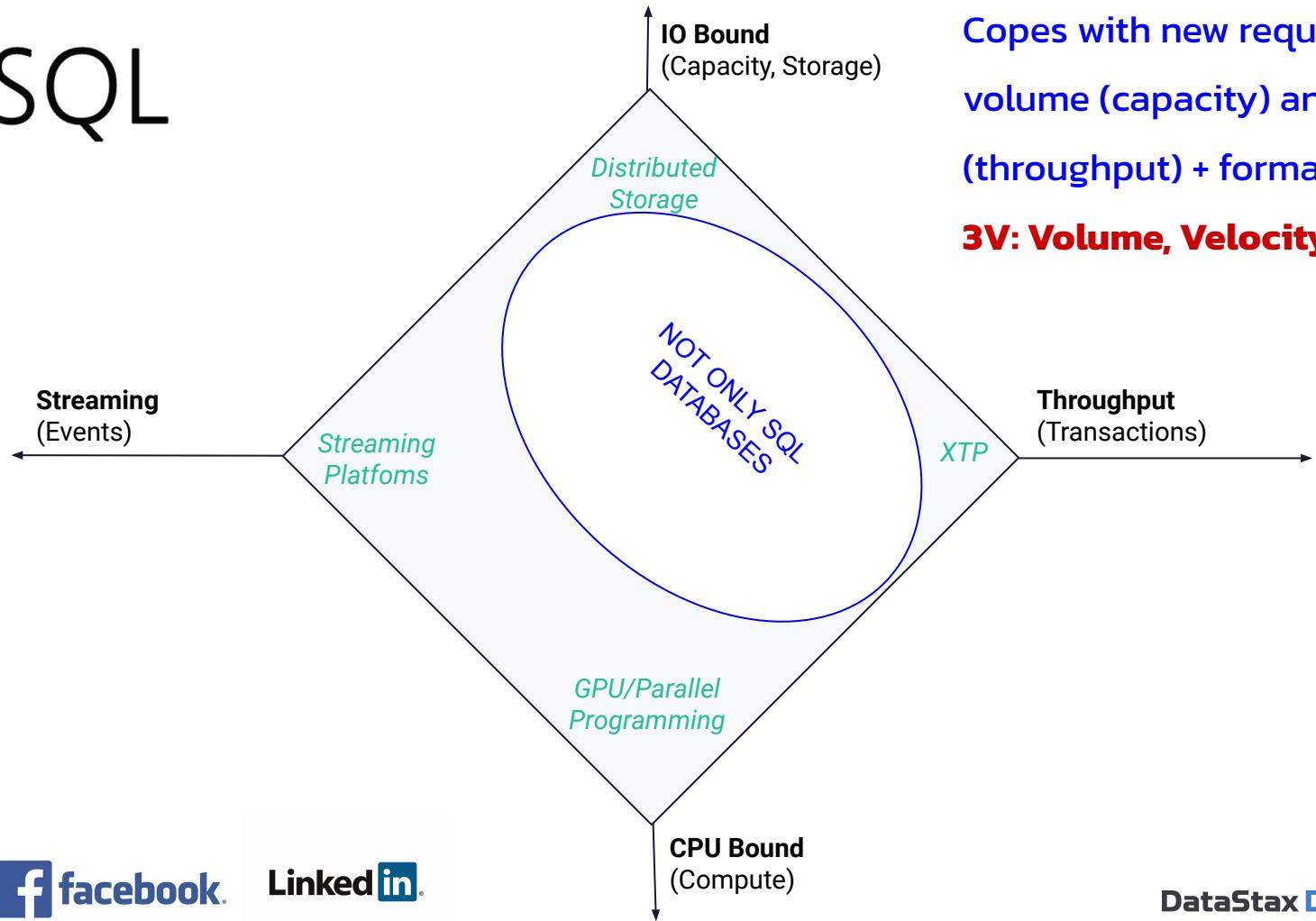
# Relational Databases Limits



At some point it does not fit  
a single machine you need  
to scale out.



# Not Only SQL



Copes with new requirements in volume (capacity) and velocity (throughput) + format (variety)

**3V: Volume, Velocity, Variety**

# Agenda

**01**

Why switch from  
SQL to NoSQL?

**02**

Three migration  
approaches

**03**

Data modeling SQL  
vs NoSQL

**04**

Migrate your data  
model

**05**

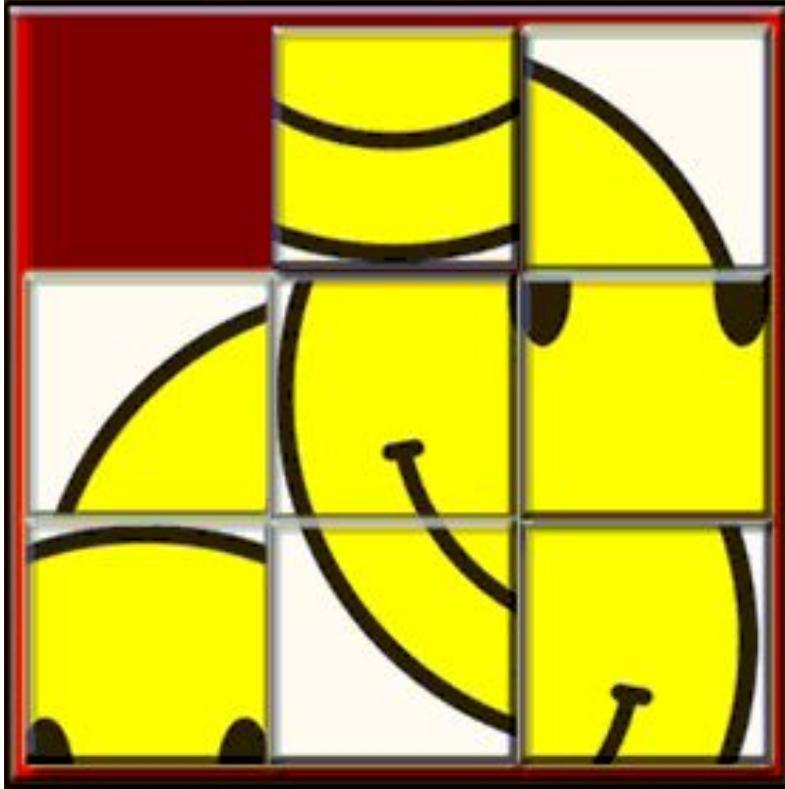
Moving data with  
DSBulk

**06**

Resources

# 42 application & data migration approaches

Let's get started!!! (this is going to take a while)



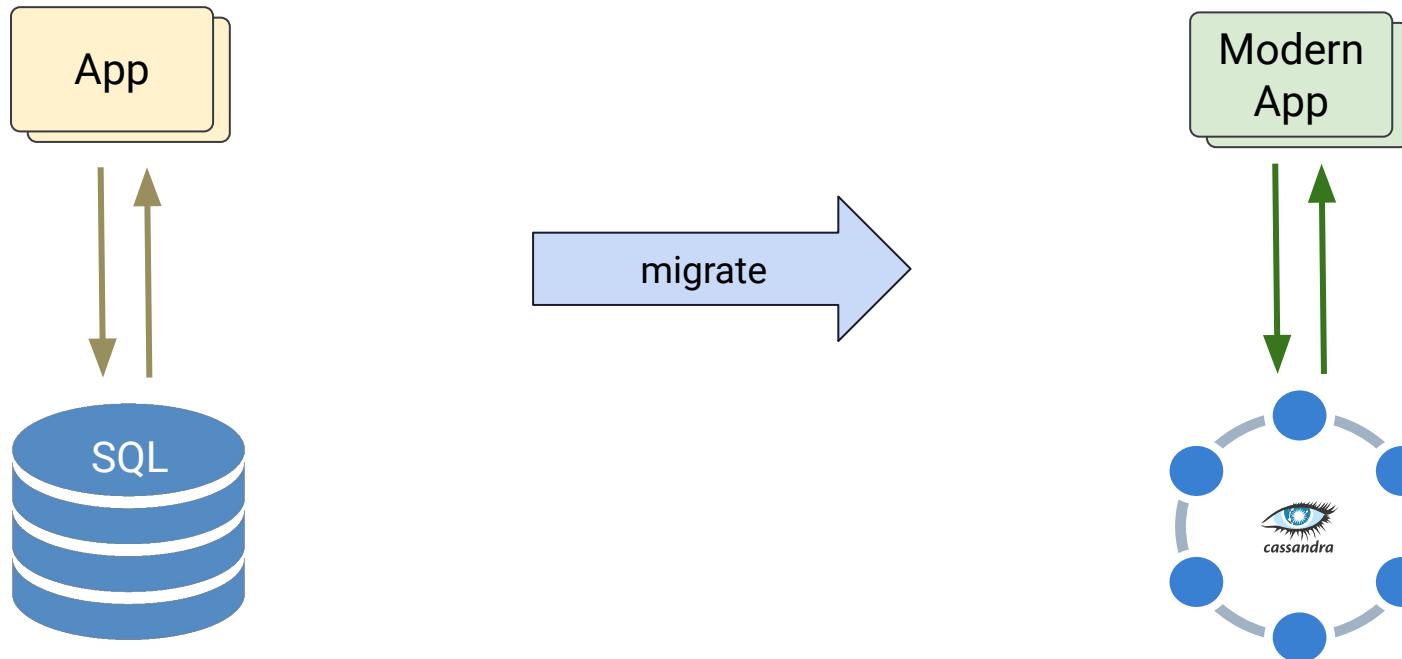
Migrating from SQL To NoSQL can sometimes feel like...

...one of those slide puzzles where you have to reason out the order of operations to get it right.

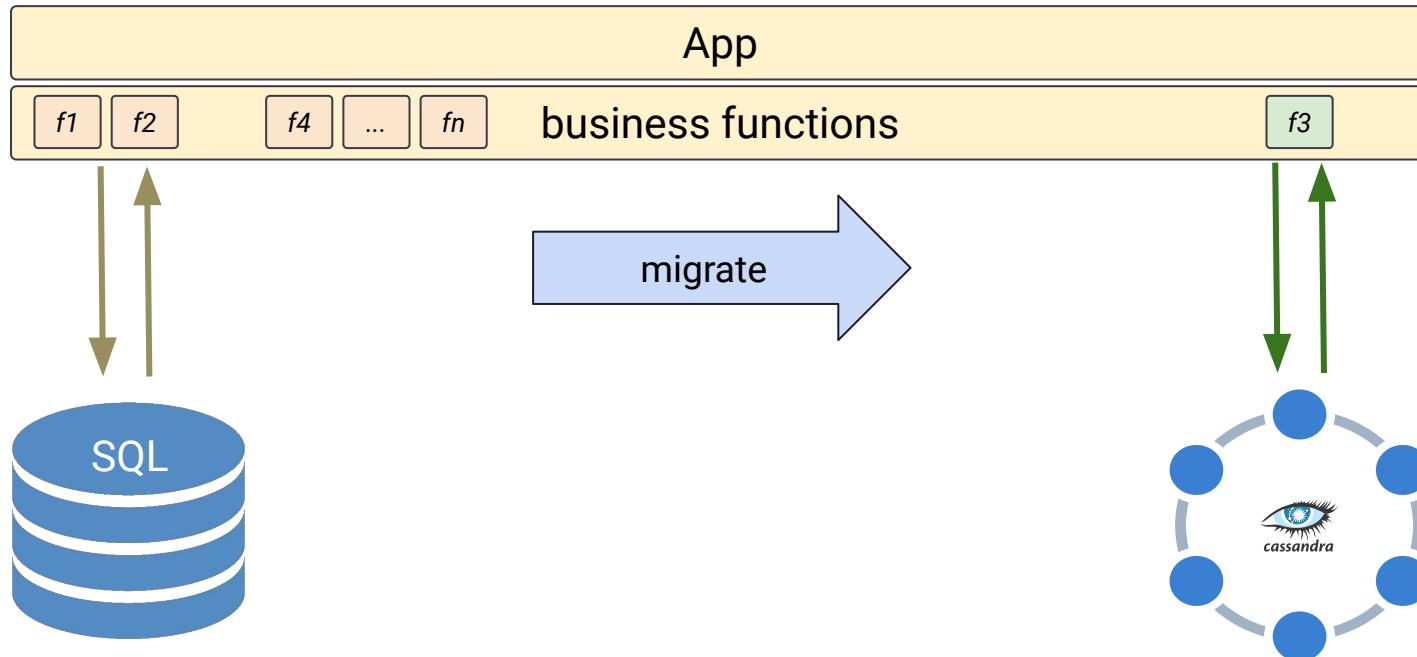
Our goal is to help you solve for smiley face.

# The problem definition

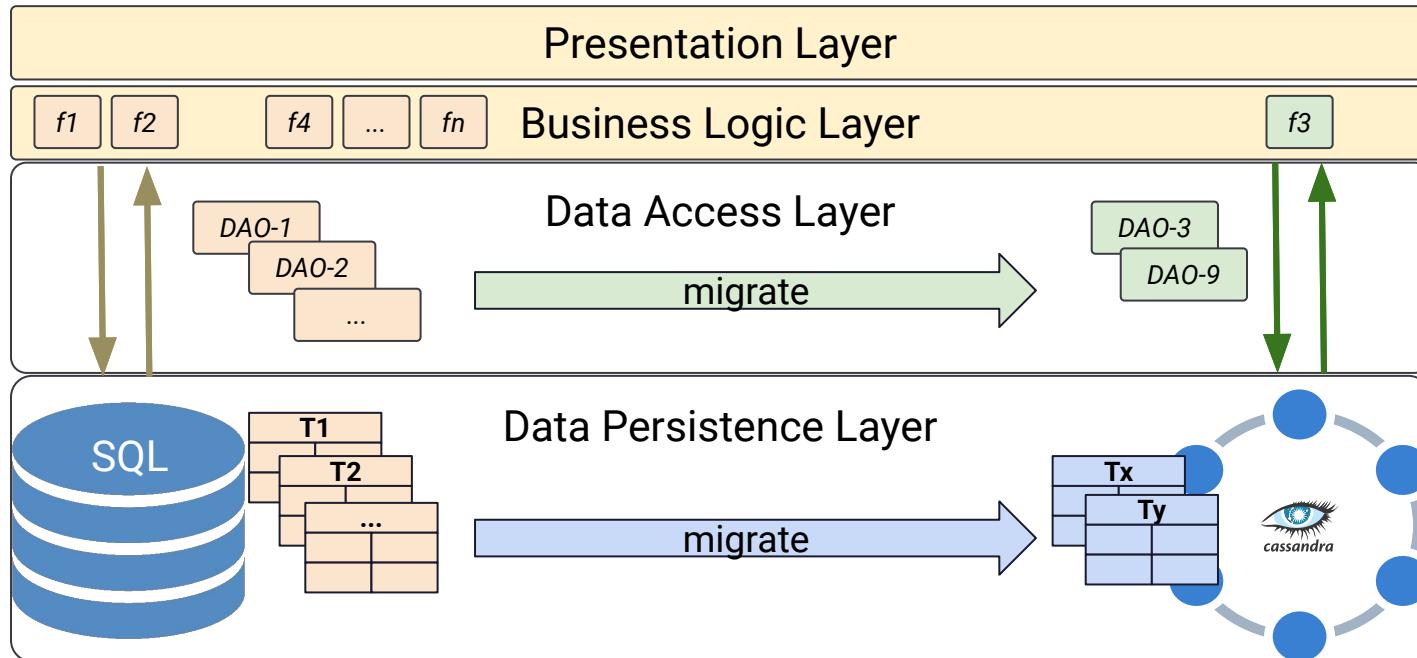
# The Migration Problem Definition in an Ideal World



# The Migration Problem Definition in the Real World



# The Migration Problem Definition Details



# The three migration approaches

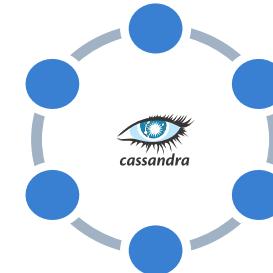
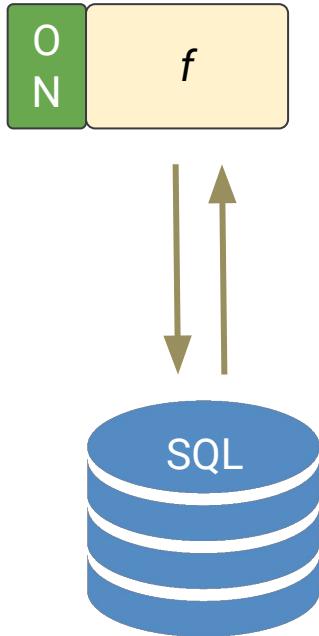
# The Three Approaches

1. Offline Migration
2. Zero-Downtime Migration with Shadow Writes
3. Minimal-Downtime Migration with Dual Reads

# Approach 1: Offline Migration

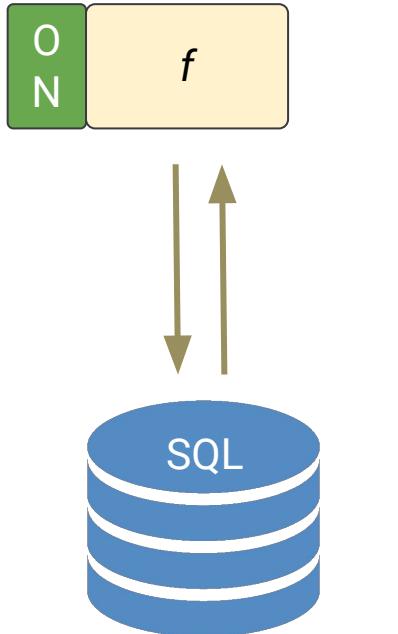
# Offline Migration

Approach 1

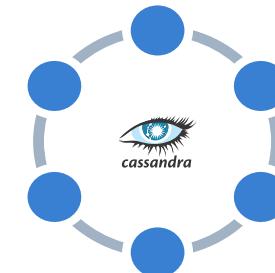


# Offline Migration

Approach 1

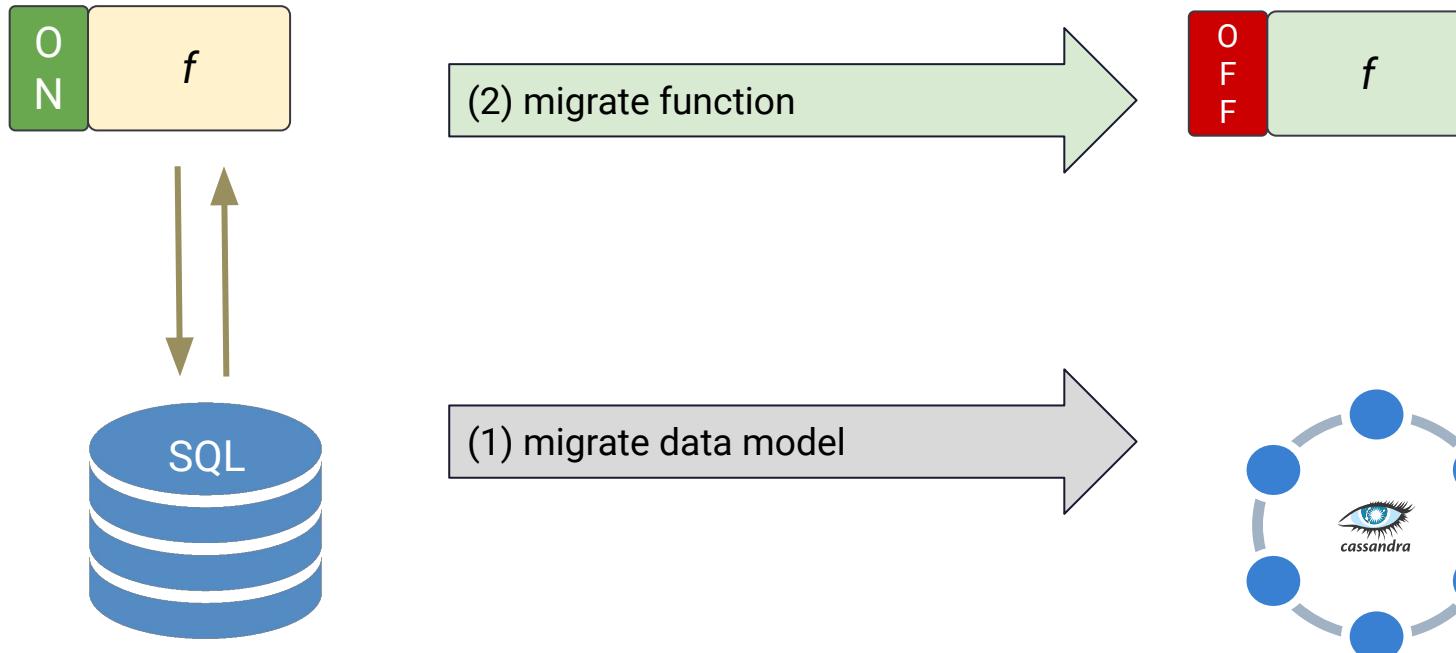


(1) migrate data model



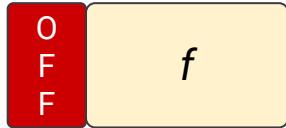
# Offline Migration

Approach 1

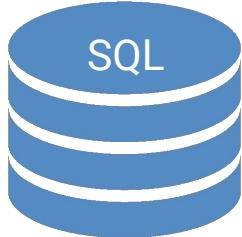
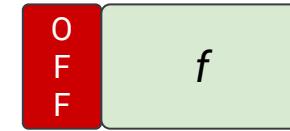


# Offline Migration

Approach 1

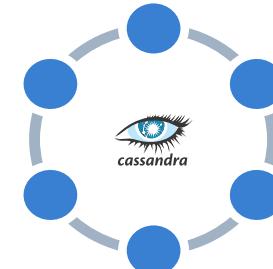


(2) migrate function



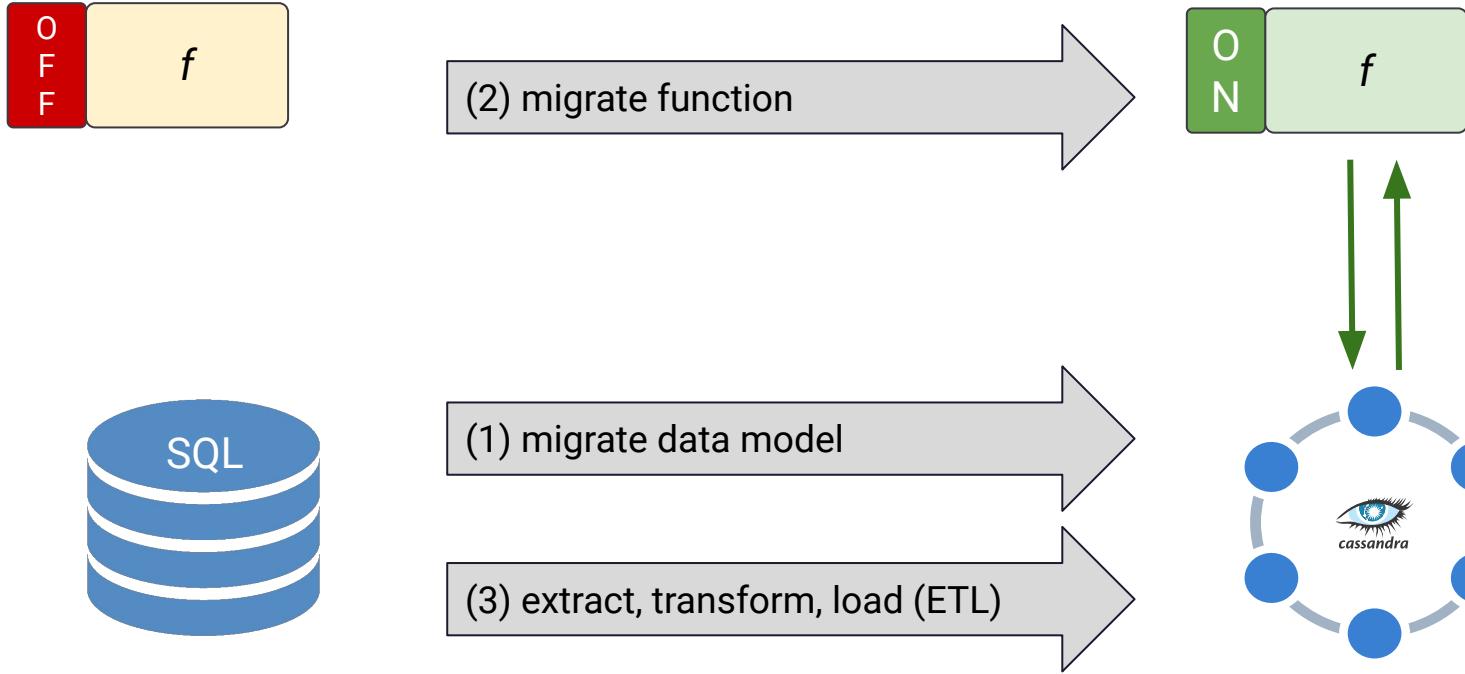
(1) migrate data model

(3) extract, transform, load (ETL)



# Offline Migration

## Approach 1

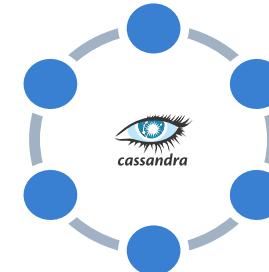
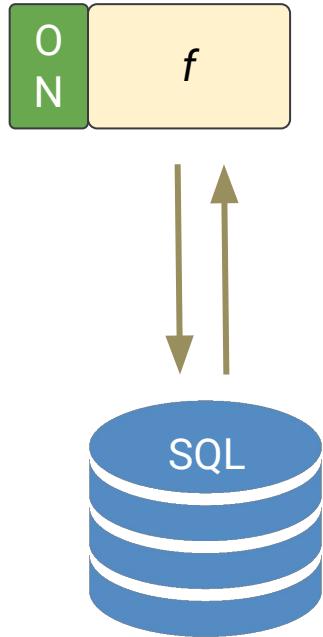


# Approach 2: Zero-Downtime Migration with Shadow Writes

Shadow writes

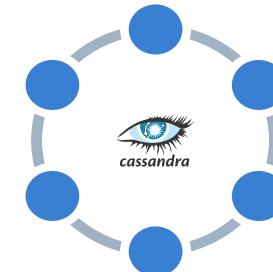
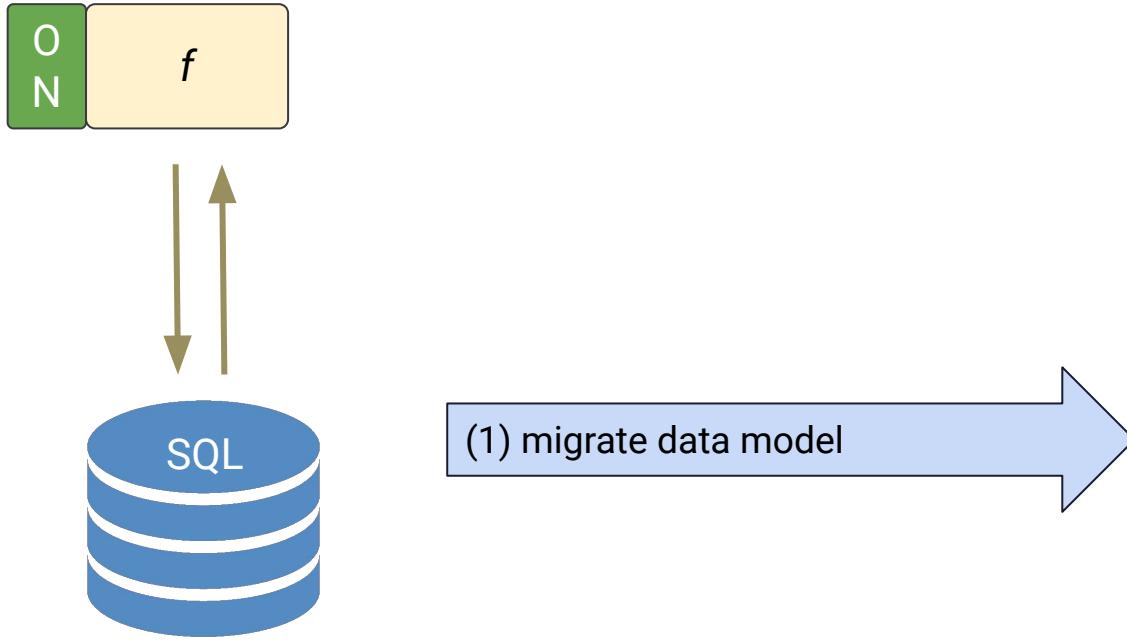
# Zero-Downtime Migration with Shadow Writes

Approach 2



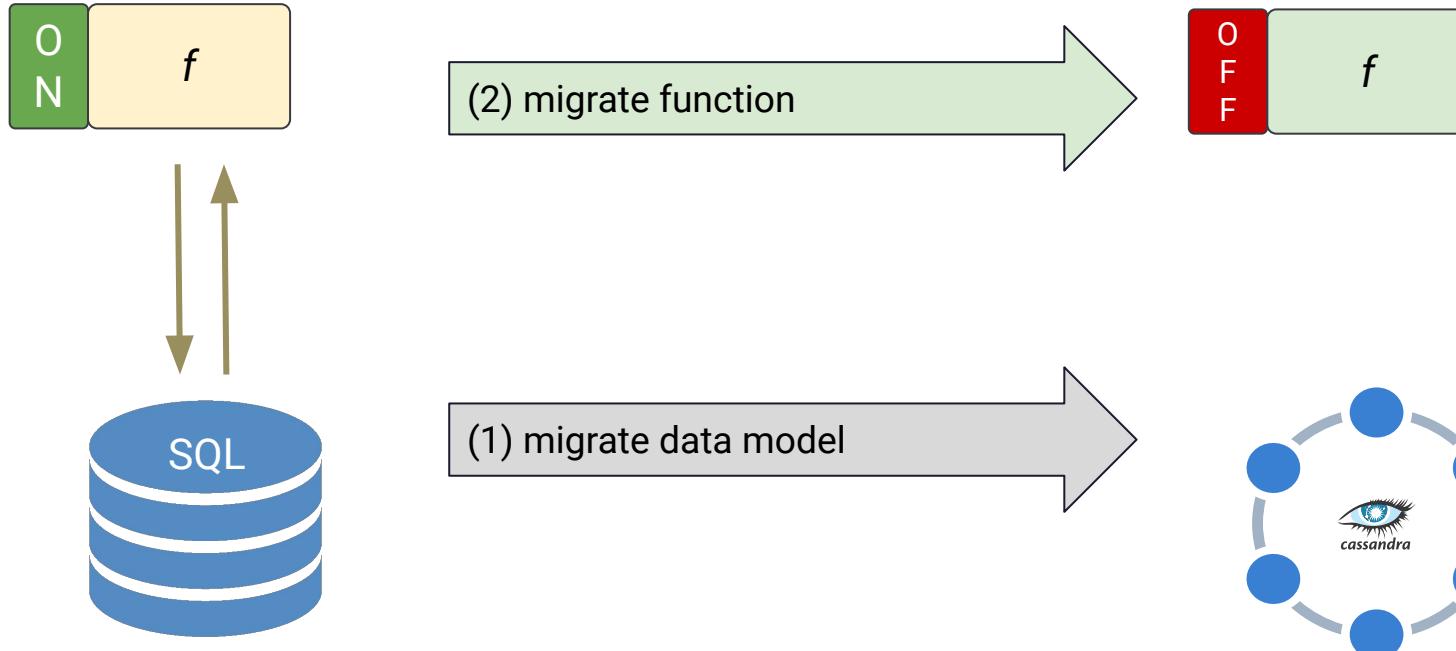
# Zero-Downtime Migration with Shadow Writes

Approach 2



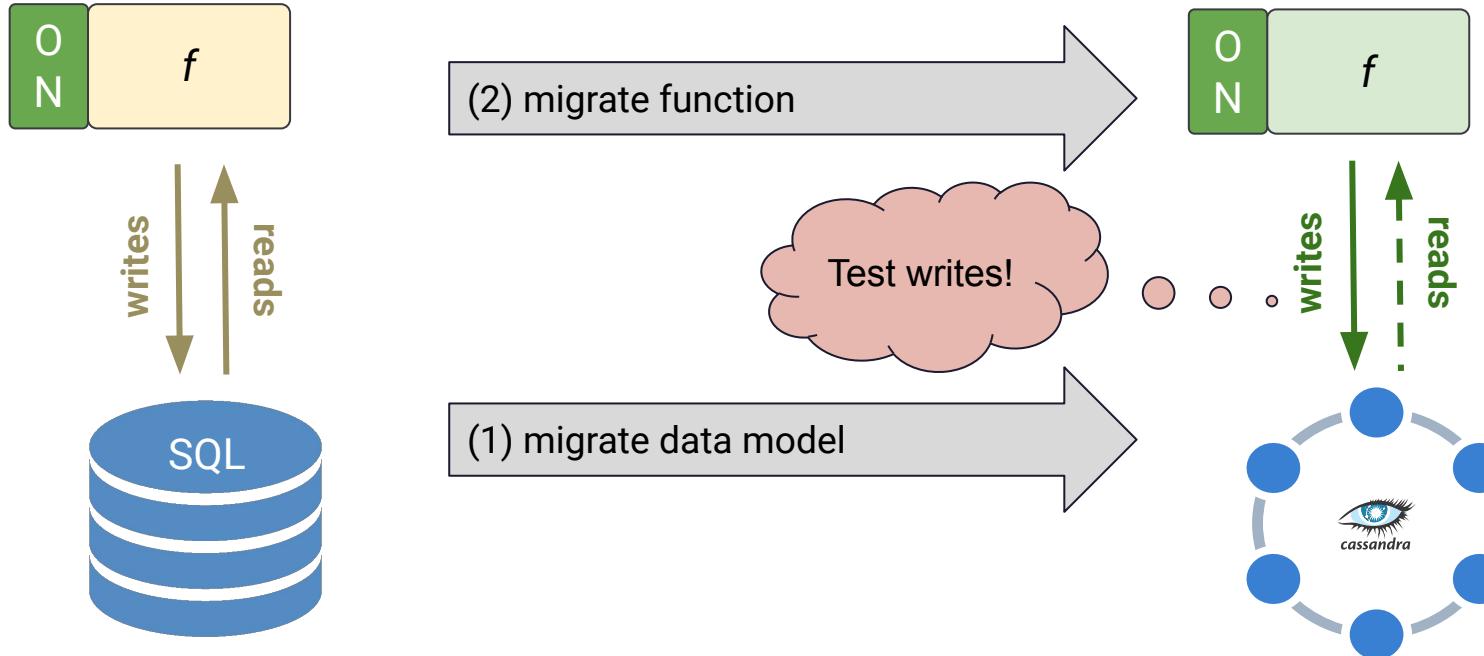
# Zero-Downtime Migration with Shadow Writes

Approach 2



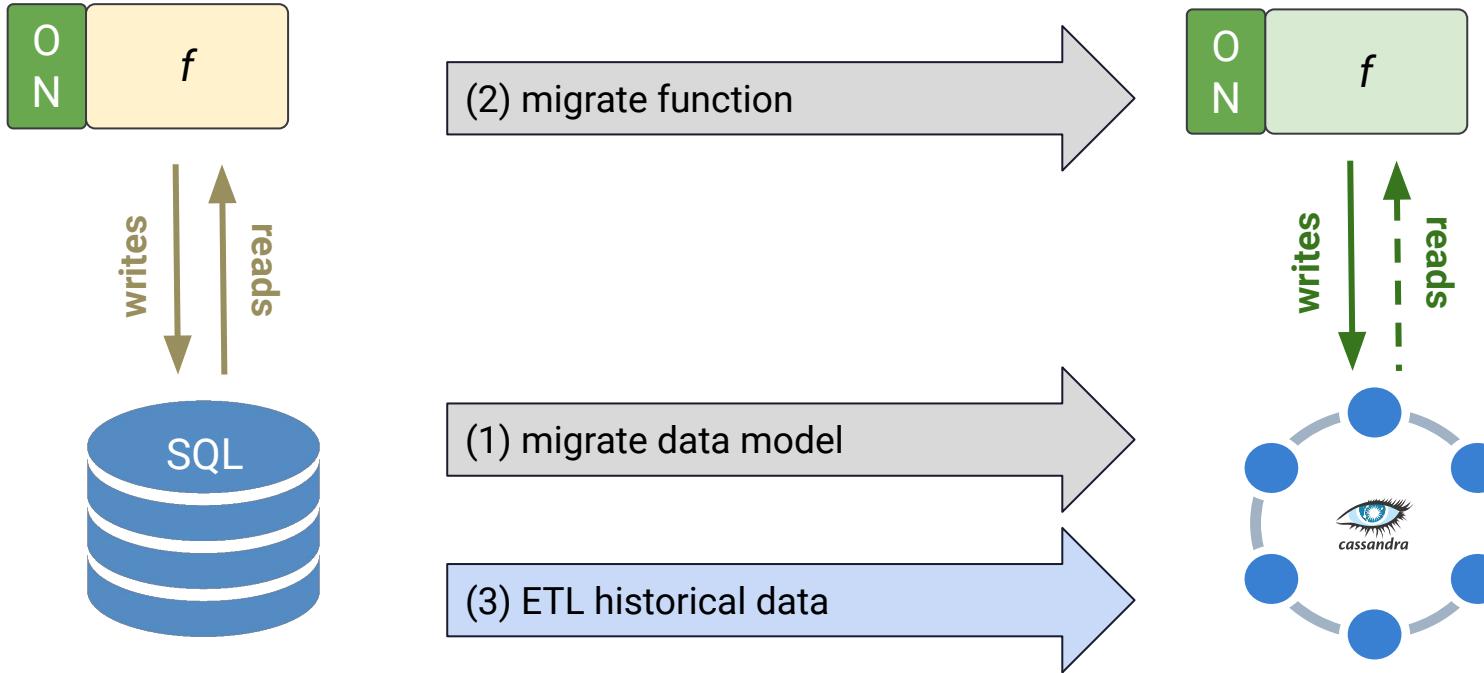
# Zero-Downtime Migration with Shadow Writes

Approach 2



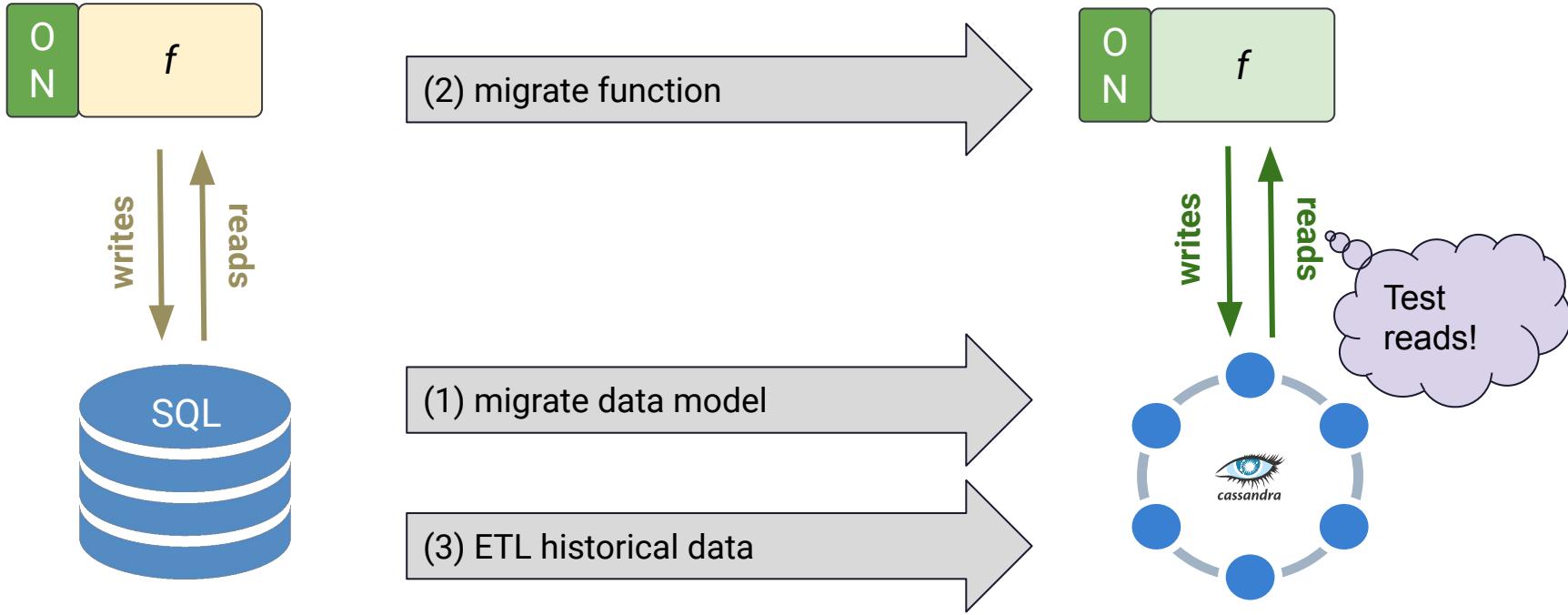
# Zero-Downtime Migration with Shadow Writes

Approach 2



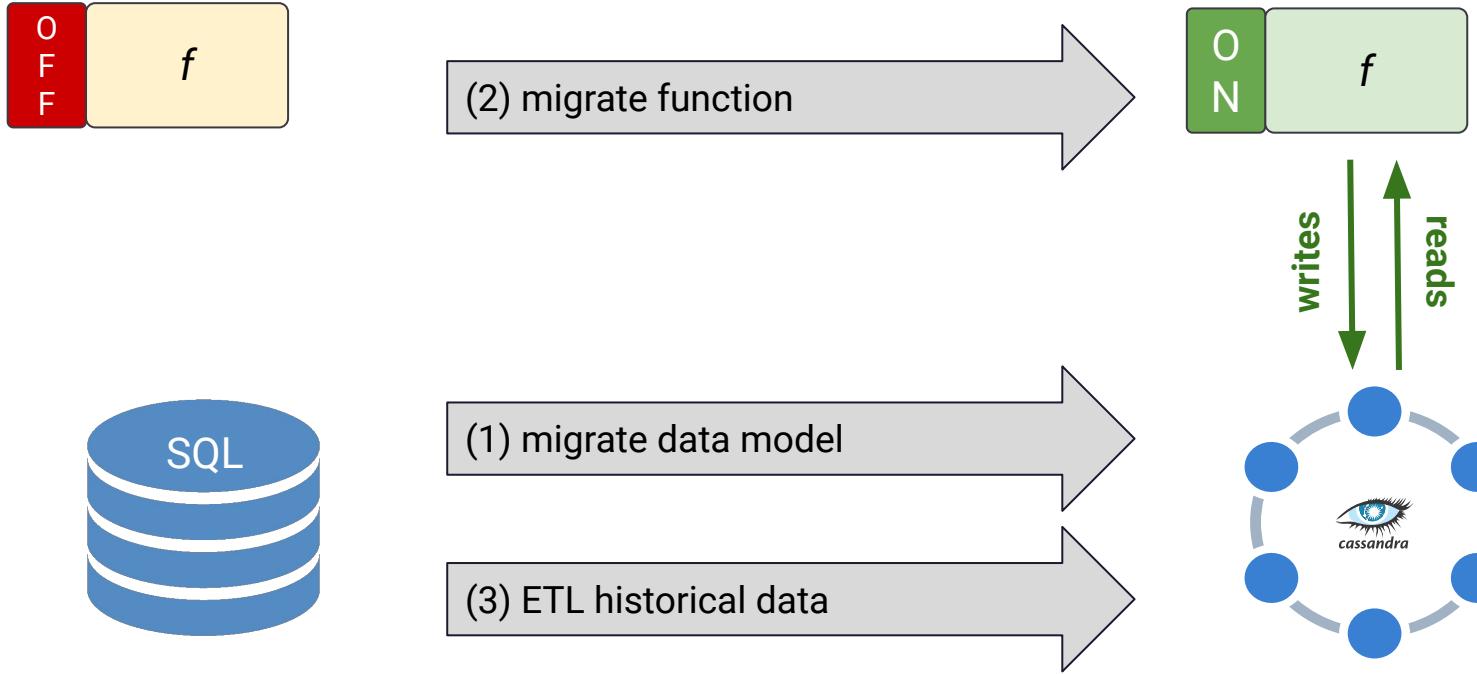
# Zero-Downtime Migration with Shadow Writes

Approach 2



# Zero-Downtime Migration with Shadow Writes

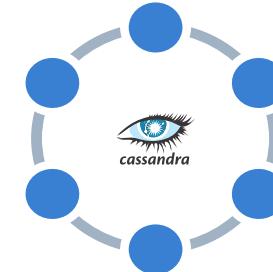
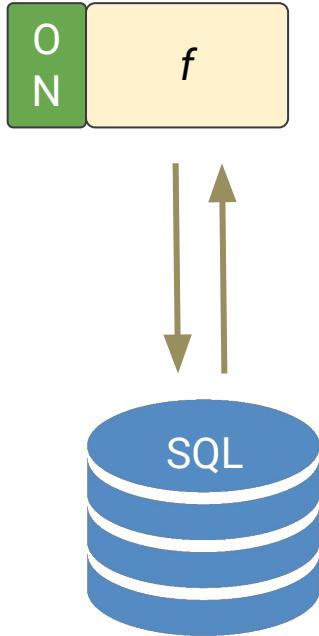
Approach 2



# Approach 3: Minimal-Downtime Migration with Dual Reads

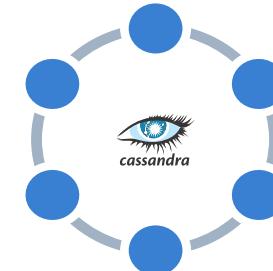
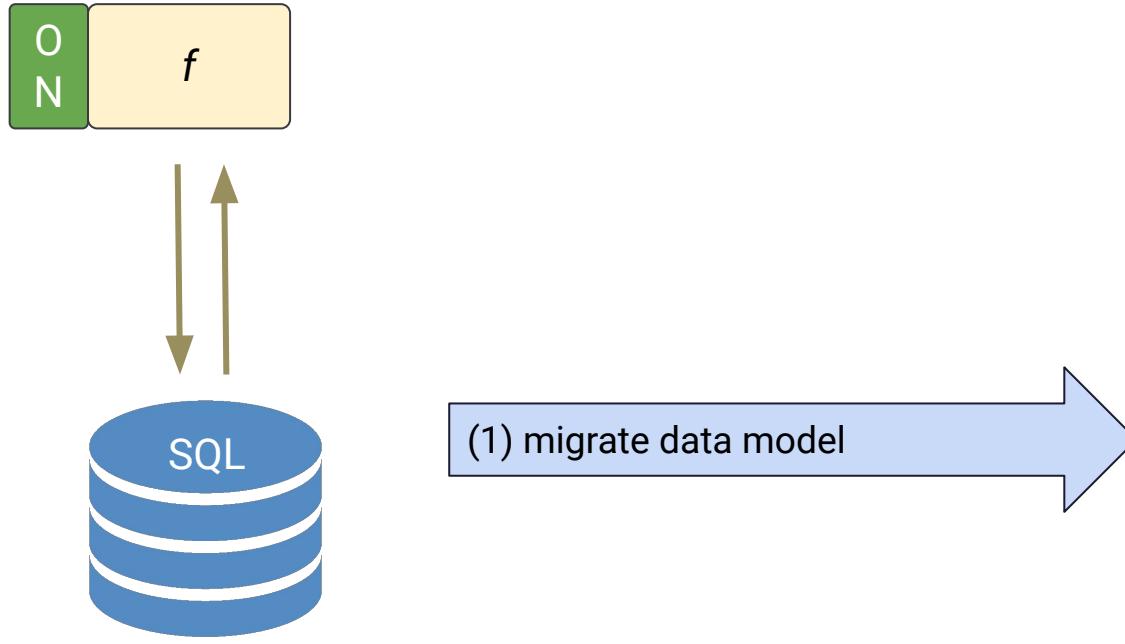
# Minimal-Downtime Migration with Dual Reads

Approach 3



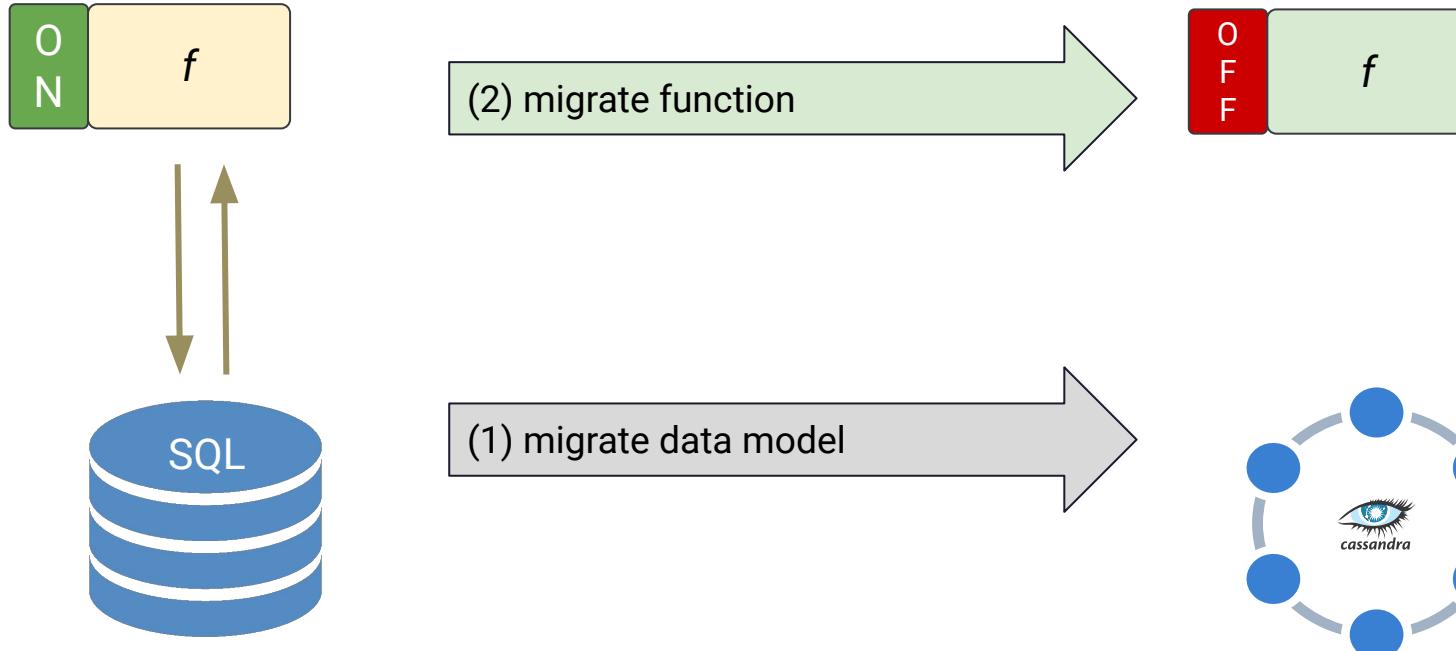
# Minimal-Downtime Migration with Dual Reads

Approach 3



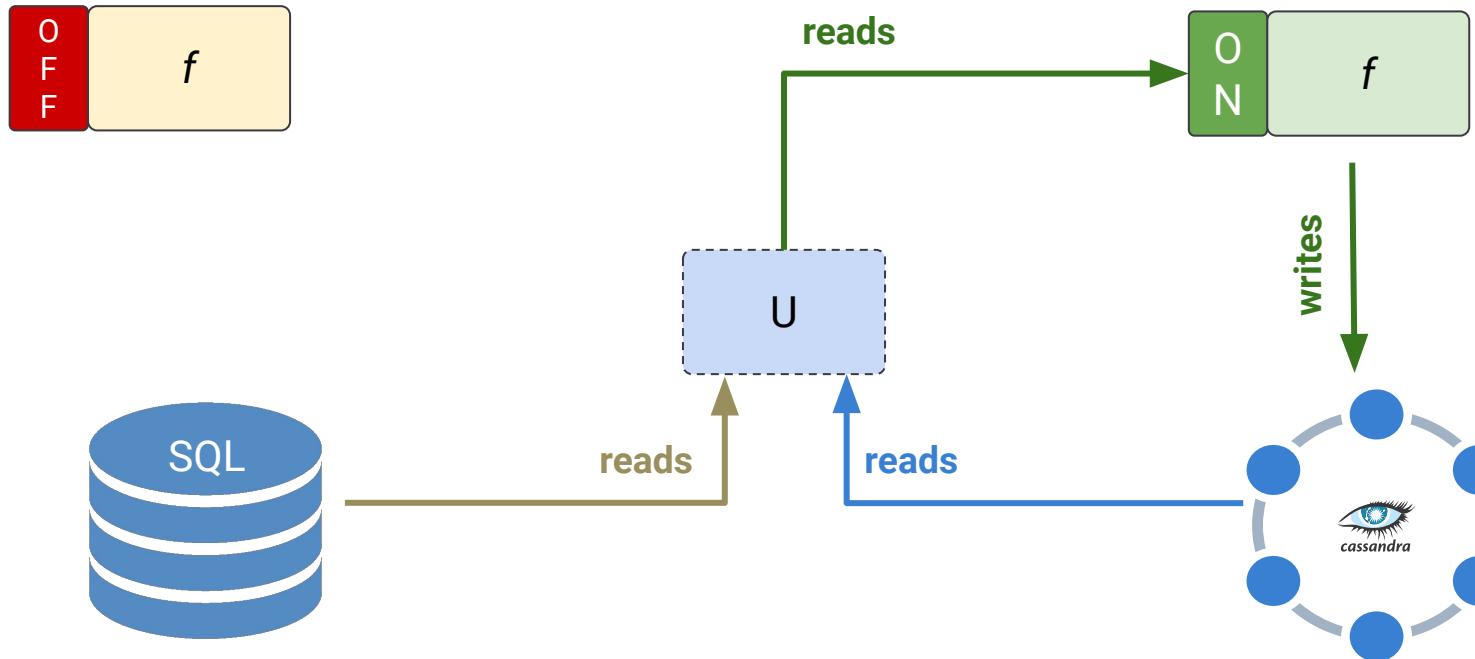
# Minimal-Downtime Migration with Dual Reads

Approach 3



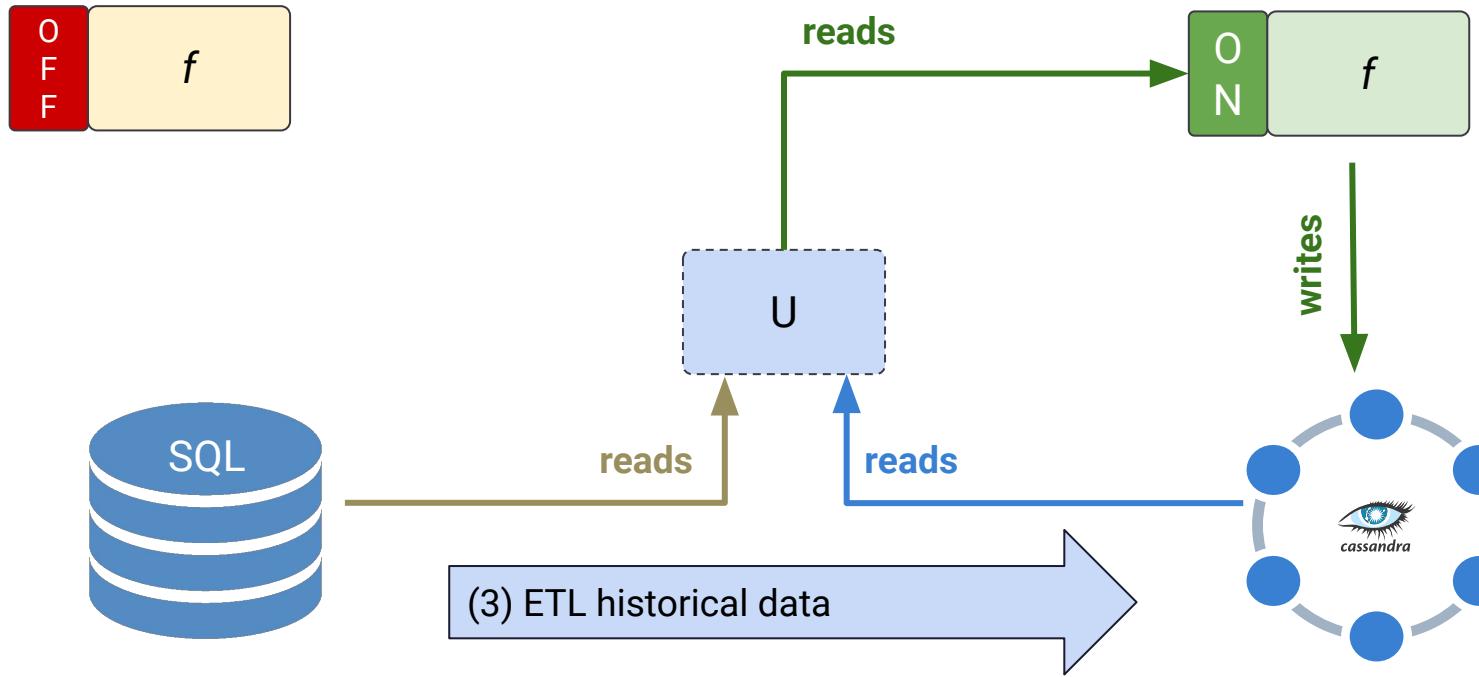
# Minimal-Downtime Migration with Dual Reads

Approach 3



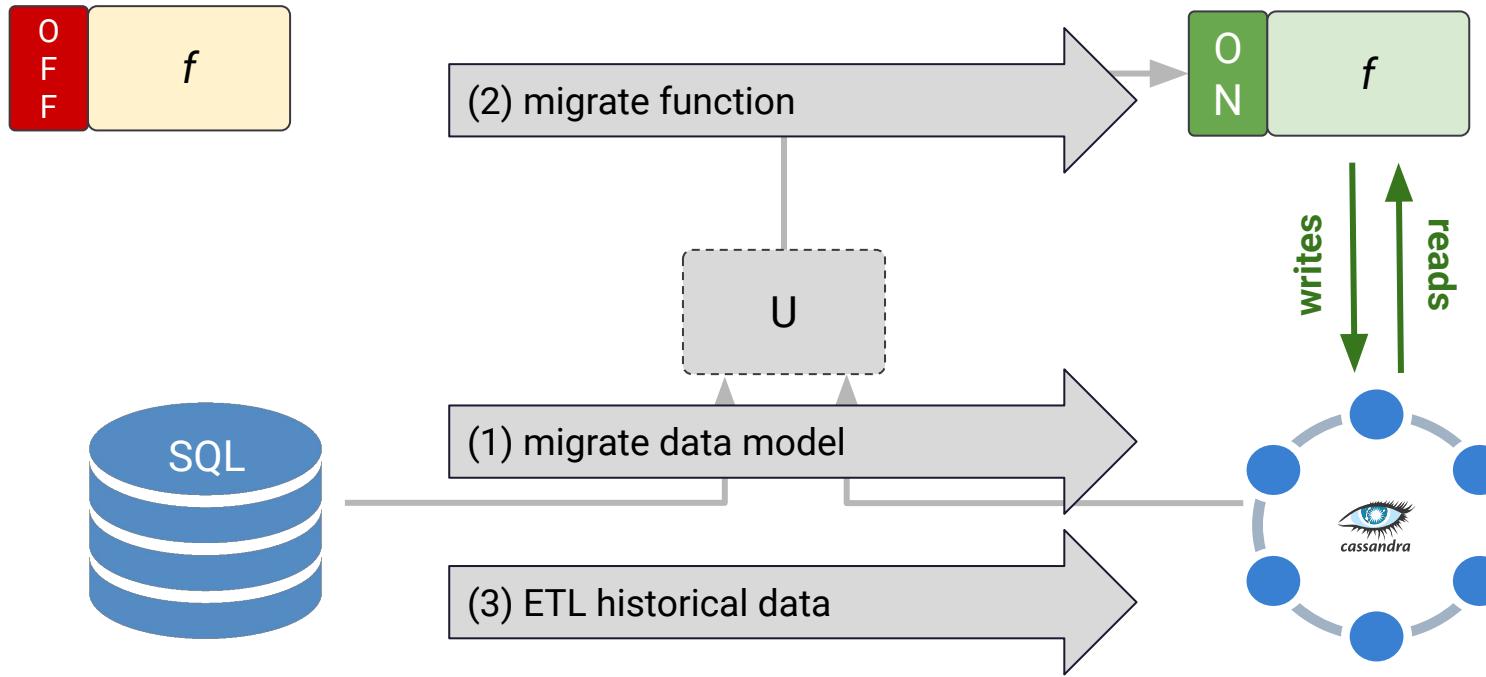
# Minimal-Downtime Migration with Dual Reads

Approach 3



# Minimal-Downtime Migration with Dual Writes

Approach 3



# Common Tasks and Challenges

- Migrate data model
  - SQL vs NoSQL data modeling
  - Query-driven table schema design
  - Denormalization, data duplication, data nesting
- Migrate function
  - Business logic does not change
  - Focus on CRUD operations, DAO and microservices
- Migrate data
  - Shadowed traffic, service proxies
  - ETL, Import/export utilities, DSBulk, Spark

# Agenda

**01**

Why switch from  
SQL to NoSQL?

**02**

Three migration  
approaches

**03**

Data modeling SQL  
vs NoSQL

**04**

Migrate your data  
model

**05**

Moving data with  
DSBulk

**06**

Resources

# Data modeling SQL vs NoSQL

“This is the way” - mando

# Normalization

"Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. It was first proposed by Edgar F. Codd as part of his relational model."

**PROS:** Simple write, Data Integrity  
**CONS:** Slow read, Complex Queries



Employees			
userId	deptId	firstName	lastName
1	1	Edgar	Codd
2	1	Raymond	Boyce

Departments	
departmentId	department
1	Engineering
2	Math

# Denormalization

"Denormalization is a strategy used on a database to increase performance. In computing, denormalization is the process of trying to improve the read performance of a database, at the expense of losing some write performance, by adding redundant copies of data"

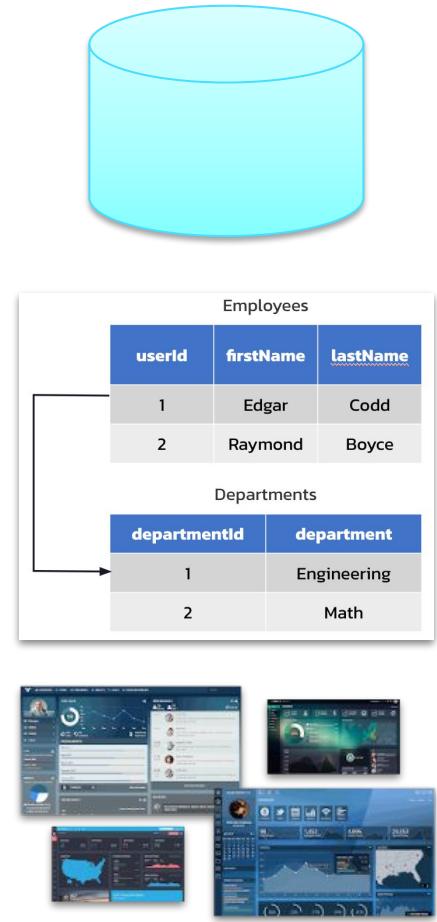
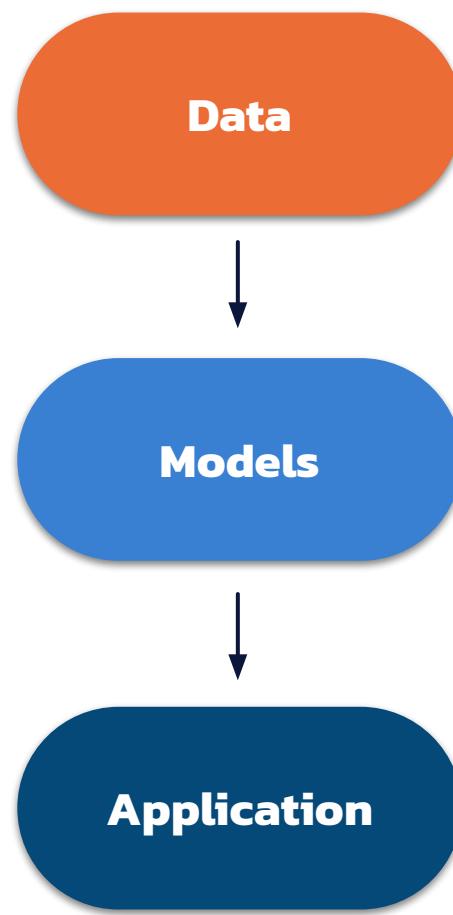
**PROS:** Quick Read, Simple Queries

**CONS:** Multiple Writes, Manual Integrity

Employees			
userId	firstName	lastName	department
1	Edgar	Codd	Engineering
2	Raymond	Boyce	Engineering
3	Sage	Lahja	Math
4	Juniper	Jones	Botany

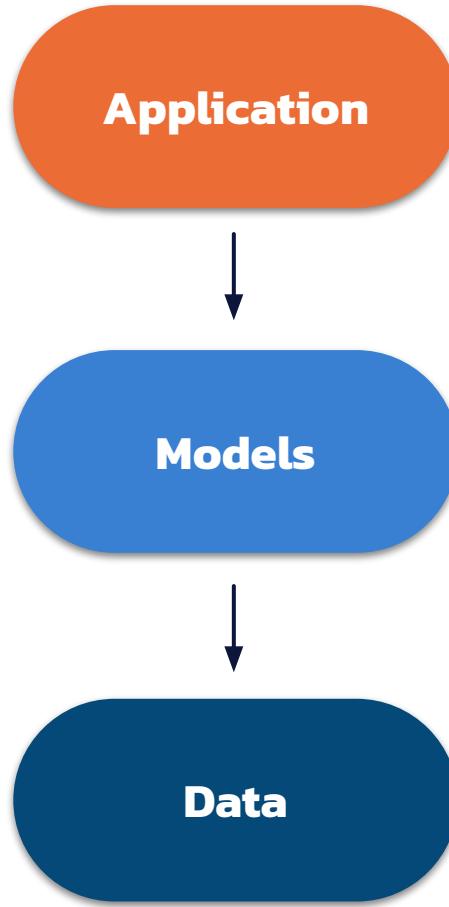
# Relational Data Modelling

1. Analyze raw data
2. Identify entities, their properties and relations
3. Design tables, using **normalization** and foreign keys.
4. Use JOIN when doing queries to join normalized data from multiple tables

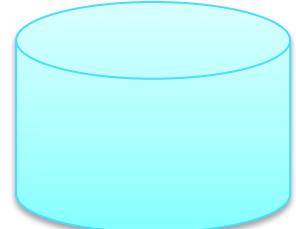


# NoSQL Data Modelling

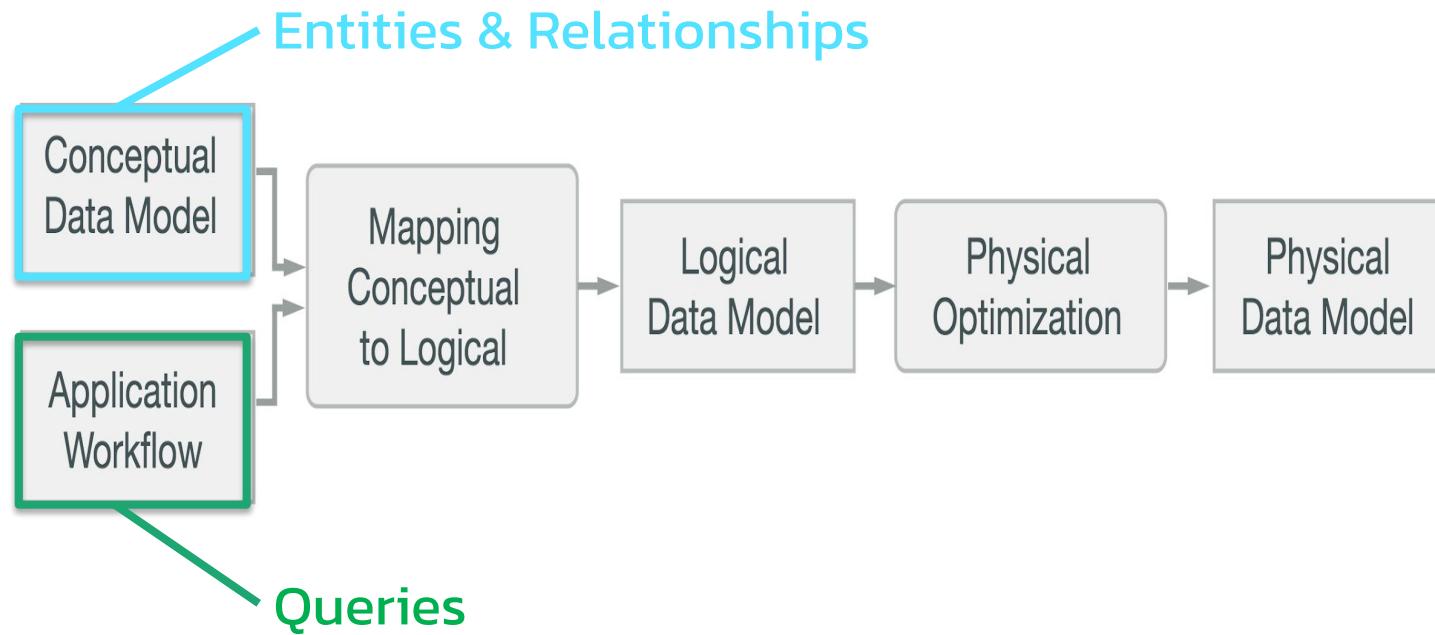
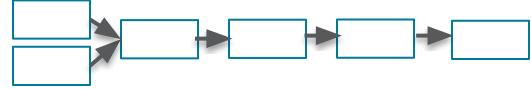
1. Analyze user behaviour  
(customer first!)
2. Identify workflows, their dependencies and needs
3. Define Queries to fulfill these workflows
4. Knowing the queries, design tables, using **denormalization**.
5. Insert and update multiple copies of data that may have resulted due to denormalization

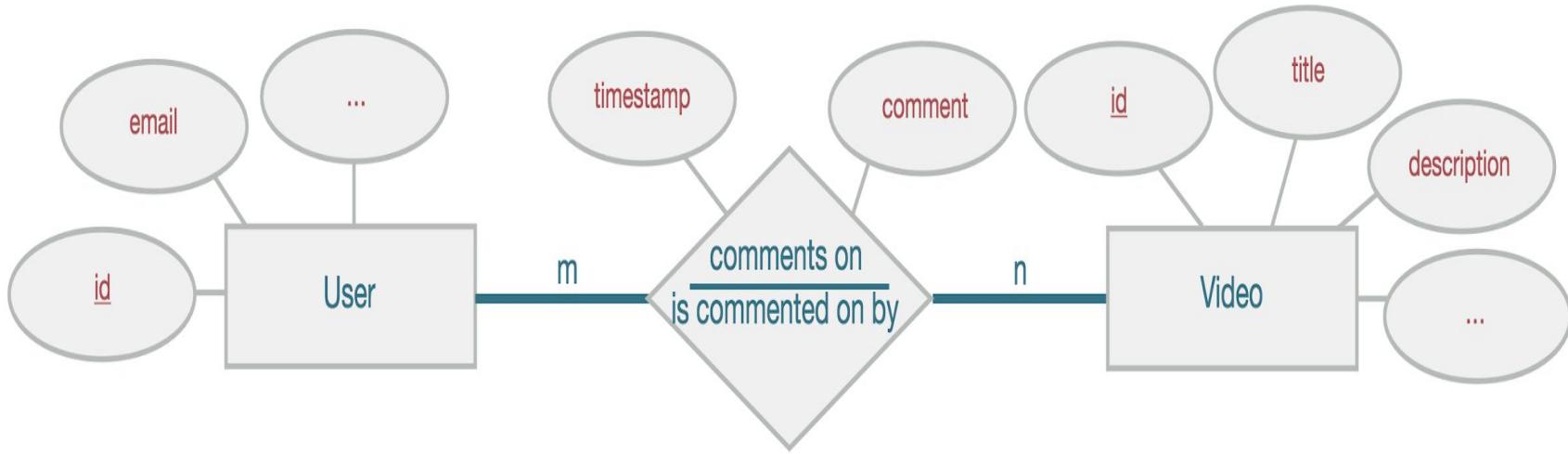
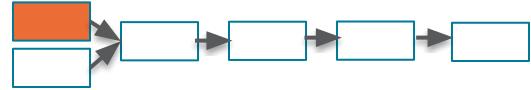


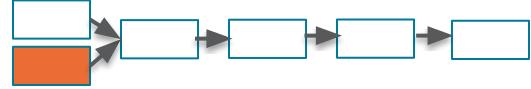
Employees			
userId	firstName	lastName	department
1	Edgar	Codd	Engineering
2	Raymond	Boyce	Math
3	Sage	Lahja	Math
4	Juniper	Jones	Botany



The Disney+ logo, featuring the word "Disney" in its signature script font with a registered trademark symbol, and the "+" sign in a bold, sans-serif font.







## Use-Case I:

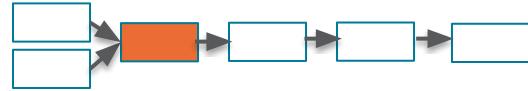
A User opens a Profile

**WF1:** Find **comments** related to target **user** using its identifier,  
most recent first

## Use-Case II:

A User opens a Video Page

**WF2:** Find **comments** related to target **video** using its identifier,  
most recent first

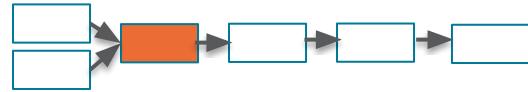


**Query I:** Find **comments** posted for a **user** with a known id (show most recent first)



**Query II:** Find **comments** for a **video** with a known id (show most recent first)



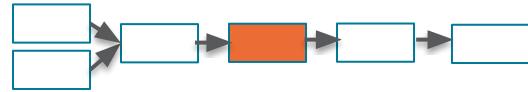


```
SELECT * FROM comments_by_user  
WHERE userid = <some UUID>
```



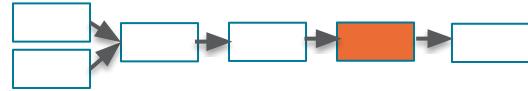
```
SELECT * FROM comments_by_video  
WHERE videoid = <some UUID>
```





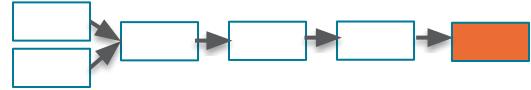
comments_by_user	
userid	K
creationdate	C ↓
commentid	C ↑
videoid	
comment	

comments_by_video	
videoid	K
creationdate	C ↓
commentid	C ↑
userid	
comment	



comments_by_user		
userid	UUID	K
commentid	TIMEUUID	C ↓
videoid	UUID	
comment	TEXT	

comments_by_video		
videoid	UUID	K
commentid	TIMEUUID	C ↓
userid	UUID	
comment	TEXT	



```
CREATE TABLE IF NOT EXISTS comments_by_user (
    userid uuid,
    commentid timeuuid,
    videoid uuid,
    comment text,
    PRIMARY KEY ((userid), commentid)
) WITH CLUSTERING ORDER BY (commentid DESC);
```

```
CREATE TABLE IF NOT EXISTS comments_by_video (
    videoid uuid,
    commentid timeuuid,
    userid uuid,
    comment text,
    PRIMARY KEY ((videoid), commentid)
) WITH CLUSTERING ORDER BY (commentid DESC);
```



# Agenda

**01**

Why switch from  
SQL to NoSQL?

**02**

Three migration  
approaches

**03**

Data modeling SQL  
vs NoSQL

**04**

Migrate your data  
model

**05**

Moving data with  
DSBulk

**06**

Resources

# Migrate your data model

Data modeling is both art and science

# Owners

**Who actually owns who here?**

# Owner - Application View



## Owners

Name	Address	City	Telephone	Pets
Heather Jones-Gilardi	AnotherPlace Dr.	Orlando	987654321	
David Gilardi	Nowhere St.	Orlando	123456789	Carrabba Shocker
<a href="#">Add Owner</a>				

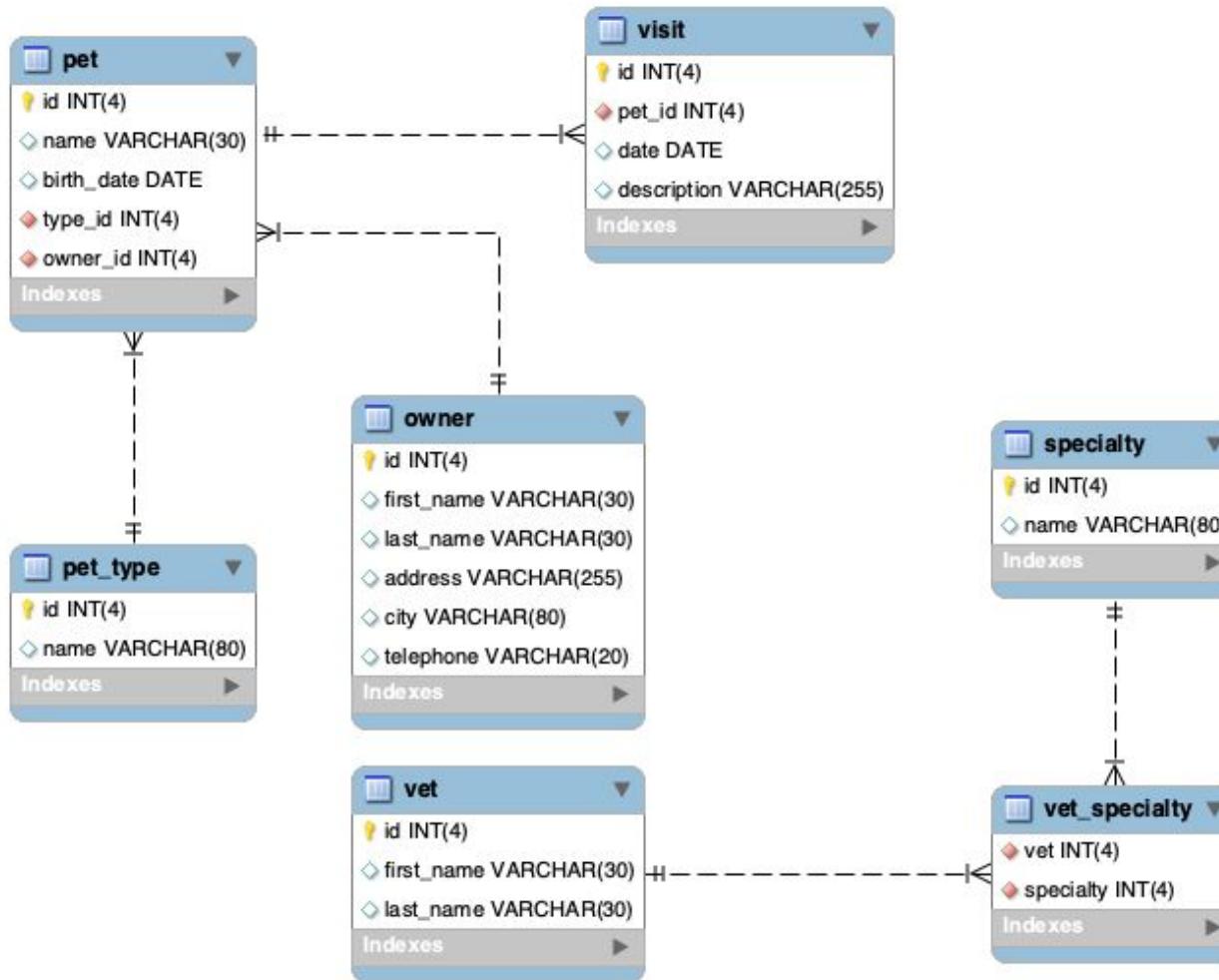
## **Use-Case:**

A User opens the **owner** “ALL” page

## **Workflow:**

List all **owners**

# SQL



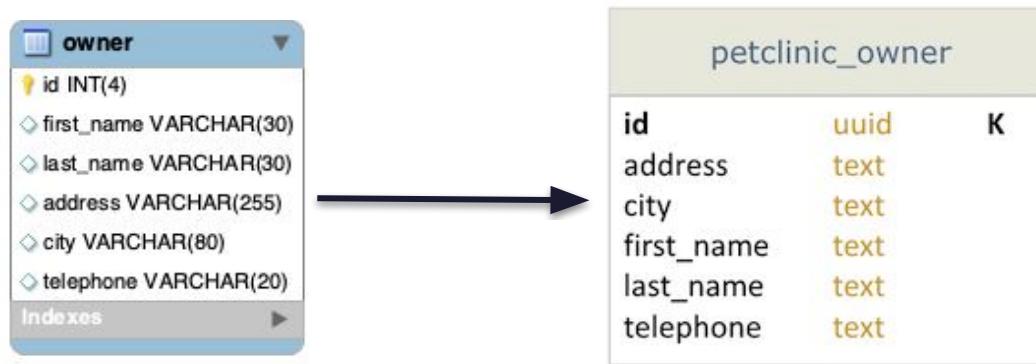
## Owner - SQL

owner	
id	INT(4)
first_name	VARCHAR(30)
last_name	VARCHAR(30)
address	VARCHAR(255)
city	VARCHAR(80)
telephone	VARCHAR(20)
Indexes	

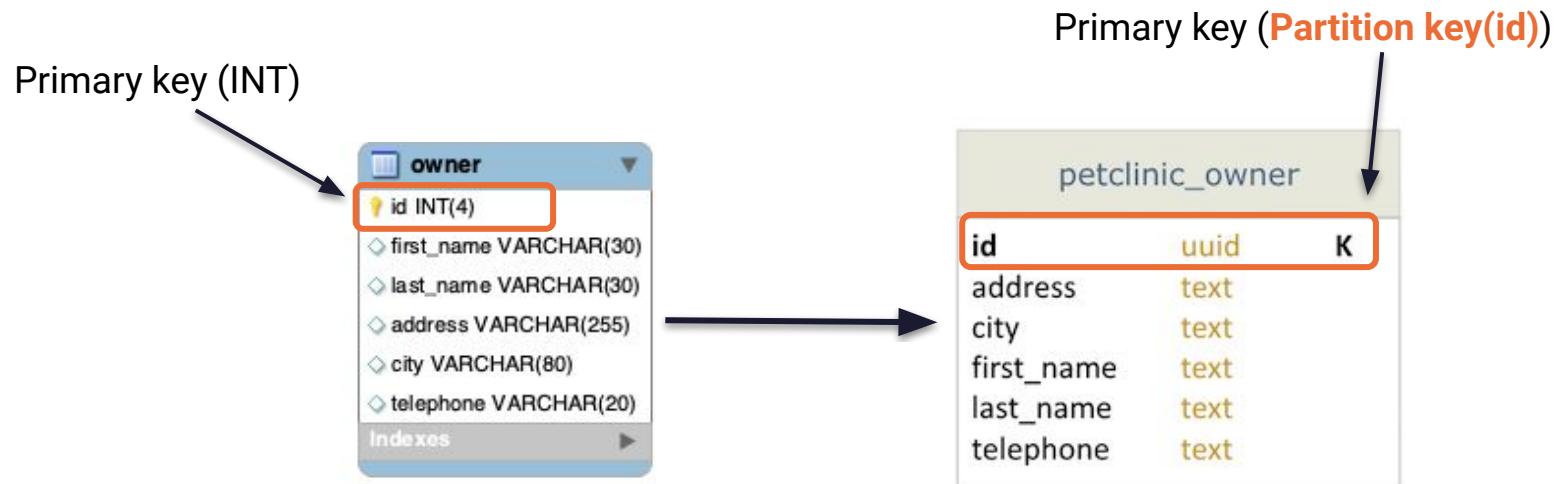
A simple list of owners and their properties

Primary key: **id**

## Owner – SQL to CQL



## Owner – SQL to CQL



*UUID's make good partition keys  
Row is unique  
One read gets ALL owner information*

# Owner Details

Use case 1: **Owner to Pet**

# Owner to Pet - Application View

## Owners

Name	Address	City	Telephone	Pets
Heather Jones-Gilardi	AnotherPlace Dr.	Orlando	987654321	
David Gilardi	Nowhere St.	Orlando	123456789	Carrabba Shocker
<a href="#">Add Owner</a>				

# Owner to Pet - Application View

## Owner Information

Name  
Address  
City  
Telephone

David Gilardi  
Nowhere St.  
Orlando  
123456789

< Back    Edit Owner    Add New Pet

## Pets and Visits

Name Shocker  
Birth Date 2021/03/29  
Type dog

Edit Pet    Delete Pet    Add Visit

Visit Date	Description	Actions	
2019/03/25	yearly visit	Edit Visit	Delete Visit
2019/03/25	yearly visit	Edit Visit	Delete Visit

Name Carrabba  
Birth Date 2014/05/23  
Type dog

Edit Pet    Delete Pet    Add Visit

Visit Date	Description	Actions	
2020/03/25	yearly visit	Edit Visit	Delete Visit
2019/03/25	yearly visit	Edit Visit	Delete Visit

# Owner to Pet - Application View

## Owner Information

Name  
Address  
City  
Telephone

David Gilardi  
Nowhere St.  
Orlando  
123456789

< Back   Edit Owner   Add New Pet

One owner

## Pets and Visits

Name Shocker  
Birth Date 2021/03/29  
Type dog

Edit Pet   Delete Pet   Add Visit

Name Carrabba  
Birth Date 2014/05/23  
Type dog

Edit Pet   Delete Pet   Add Visit

Visit Date	Description	Actions	
2019/03/25	yearly visit	Edit Visit	Delete Visit
2019/03/25	yearly visit	Edit Visit	Delete Visit

Visit Date	Description	Actions	
2020/03/25	yearly visit	Edit Visit	Delete Visit
2019/03/25	yearly visit	Edit Visit	Delete Visit

Multiple pets

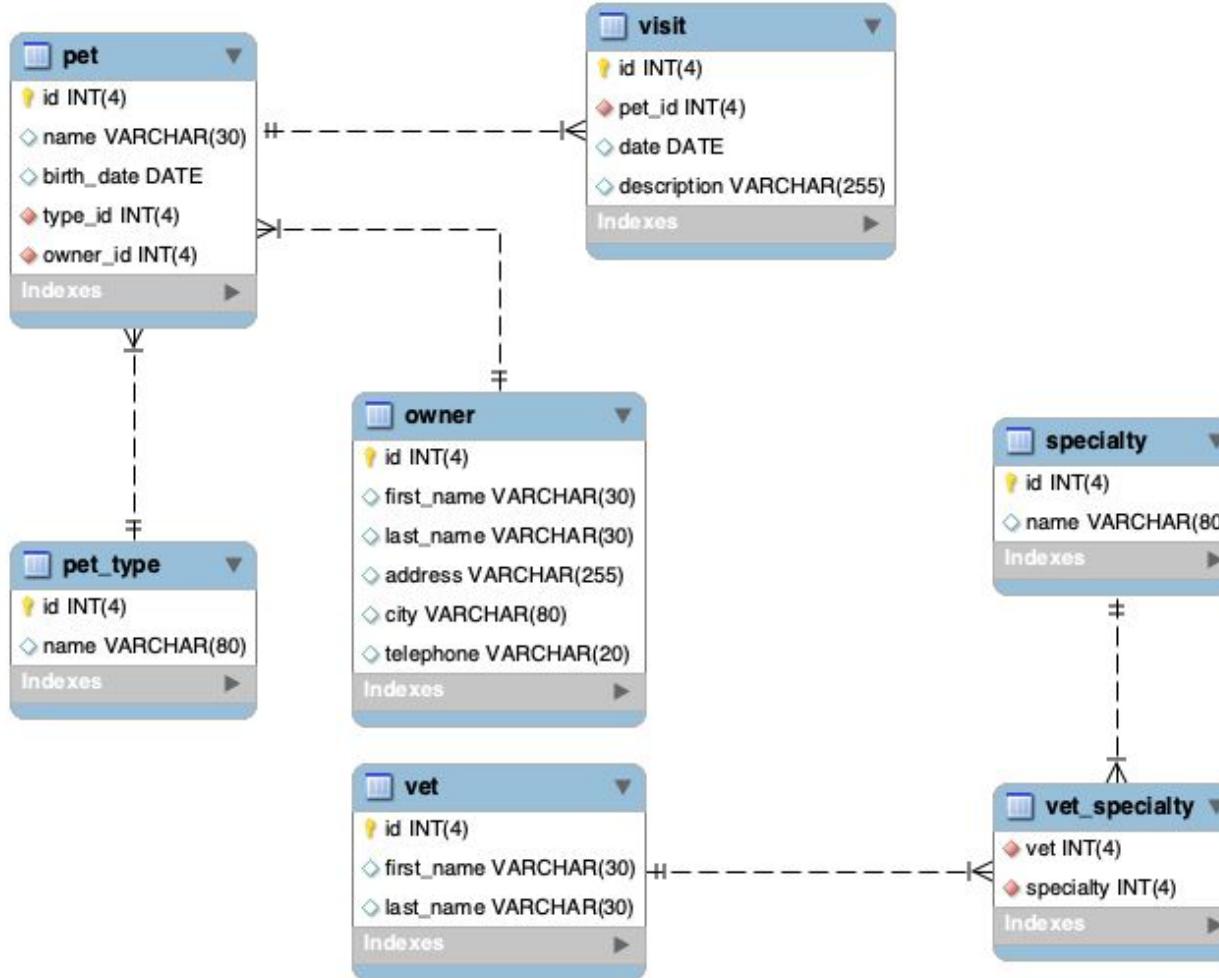
## **Use-Case:**

A User opens the owner detail page

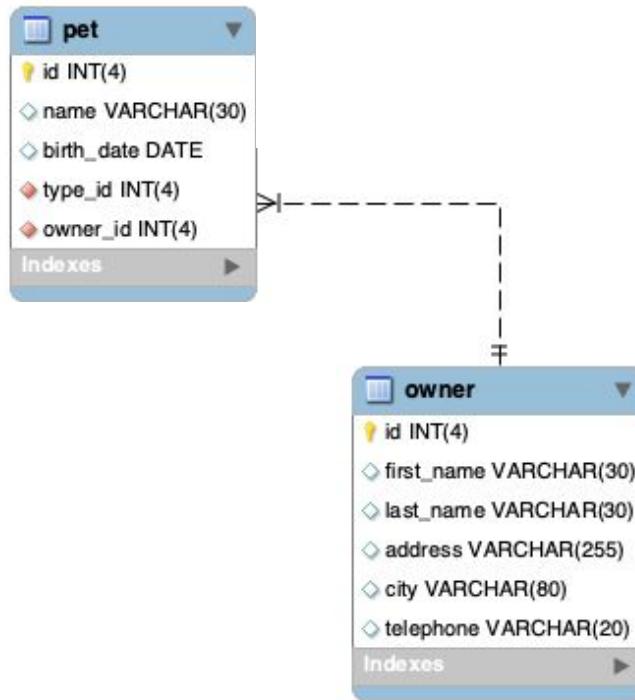
## **Workflow:**

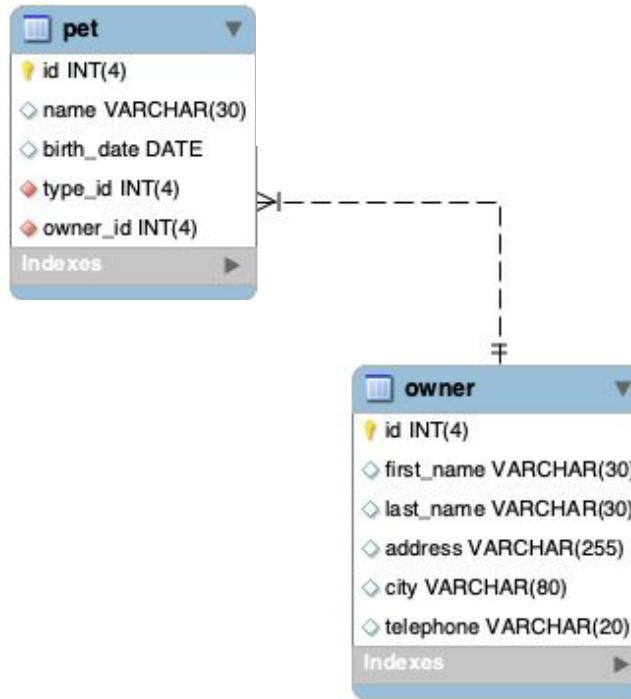
Find **owner** and any **pets** related to target  
**owner** using its identifier (**owner\_id**)

# SQL



## Owner to Pet - SQL





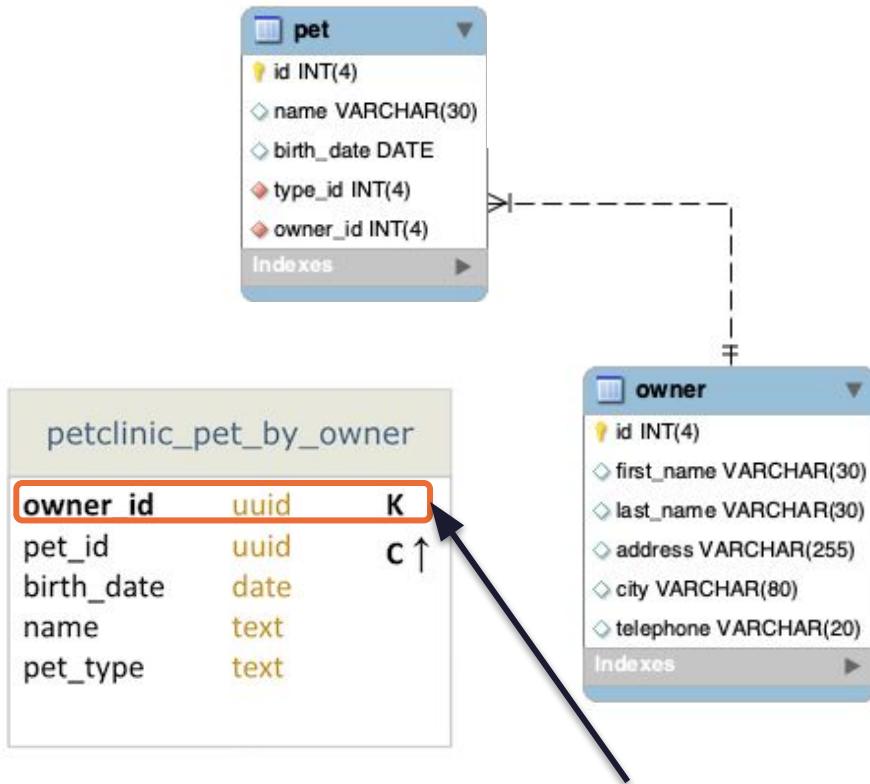
## Owner to Pet - SQL

### One to Many relationship

An owner can have multiple pets, but a pet can only have one owner

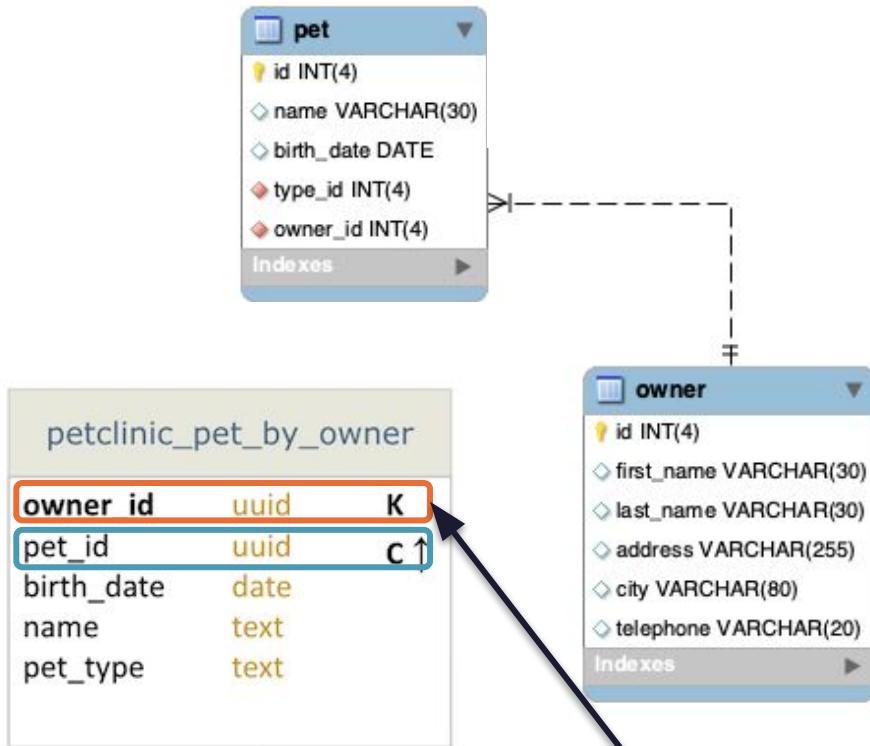
relationship: **id** to **owner\_id** (foreign key)

## Owner to Pet - SQL to CQL



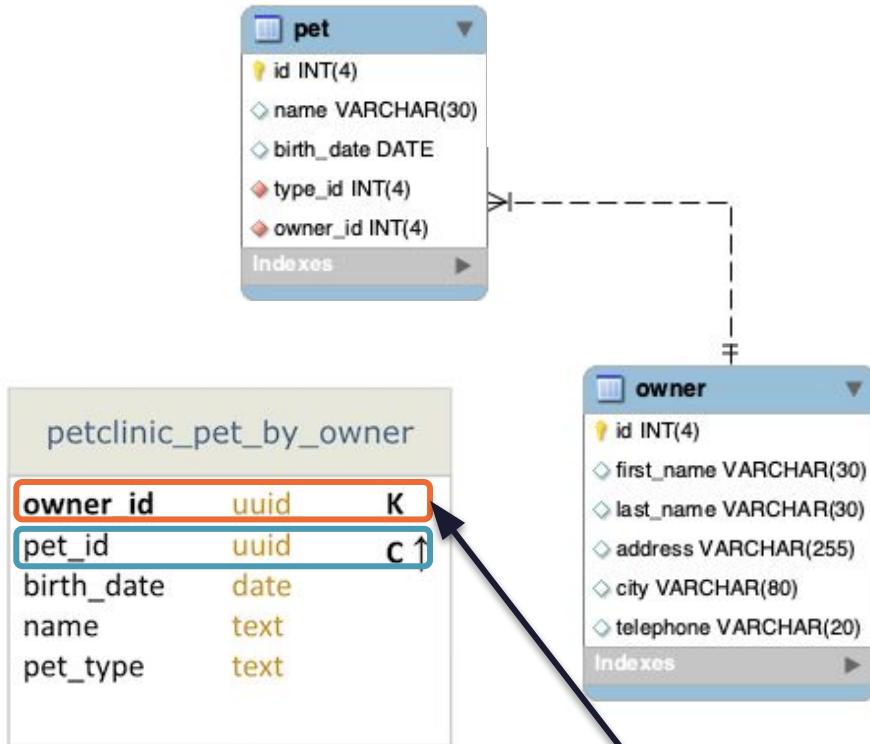
Primary key (**Partition key(owner\_id)**)

## Owner to Pet - SQL to CQL



Primary key ( )  
Partition key(**owner\_id**)  
Clustering column(**pet\_id**)

# Owner to Pet – SQL to CQL



Primary key (

Partition key(`owner_id`)  
Clustering column(`pet_id`))

When using **multiple attributes**  
(`birth_date`, `name`, `pet_type`)

Using a clustering column = **multi-row partition**

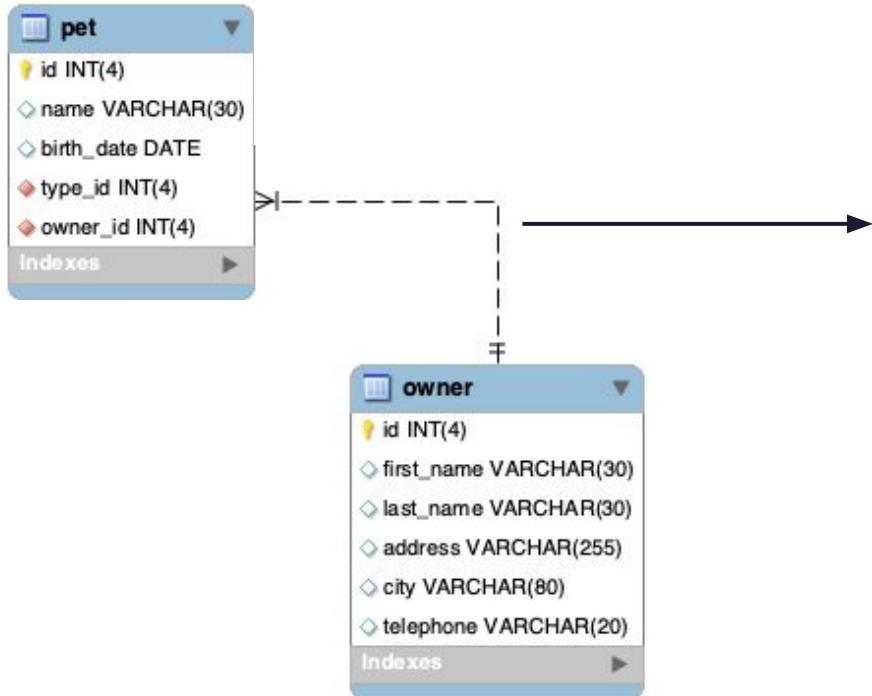
Means for every owner partition there  
can be **1 to n rows of pets** on a single  
read

**One to Many**

# What's another option?

# Another option

- Make owner fields STATIC



petclinic_pet_by_owner		
owner_id	uuid	K
pet_id	uuid	C ↑
birth_date	date	
name	text	
pet_type	text	
address	text	S
city	text	S
first_name	text	S
last_name	text	S
telephone	text	S

# Owner Details

Use case 2: Pet to Visit

# Owner to Pet - Application View

## Owner Information

Name  
Address  
City  
Telephone

David Gilardi  
Nowhere St.  
Orlando  
123456789

< Back   Edit Owner   Add New Pet

One owner

## Pets and Visits

Name Shocker  
Birth Date 2021/03/29  
Type dog

Edit Pet   Delete Pet   Add Visit

Name Carrabba  
Birth Date 2014/05/23  
Type dog

Edit Pet   Delete Pet   Add Visit

Visit Date	Description	Actions	
2019/03/25	yearly visit	Edit Visit	Delete Visit
2019/03/25	yearly visit	Edit Visit	Delete Visit

Visit Date	Description	Actions	
2020/03/25	yearly visit	Edit Visit	Delete Visit
2019/03/25	yearly visit	Edit Visit	Delete Visit

Multiple pets

# Owner to Pet - Application View

## Owner Information

Name  
Address  
City  
Telephone

David Gilardi  
Nowhere St.  
Orlando  
123456789

< Back   Edit Owner   Add New Pet

## Pets and Visits

Name Shocker  
Birth Date 2021/03/29  
Type dog

Edit Pet   Delete Pet   Add Visit

Name Carrabba  
Birth Date 2014/05/23  
Type dog

Edit Pet   Delete Pet   Add Visit

One owner

Multiple visits per each pet

Visit Date	Description	Actions	
2019/03/25	yearly visit	Edit Visit	Delete Visit
2019/03/25	yearly visit	Edit Visit	Delete Visit

Visit Date	Description	Actions	
2020/03/25	yearly visit	Edit Visit	Delete Visit
2019/03/25	yearly visit	Edit Visit	Delete Visit

Multiple pets

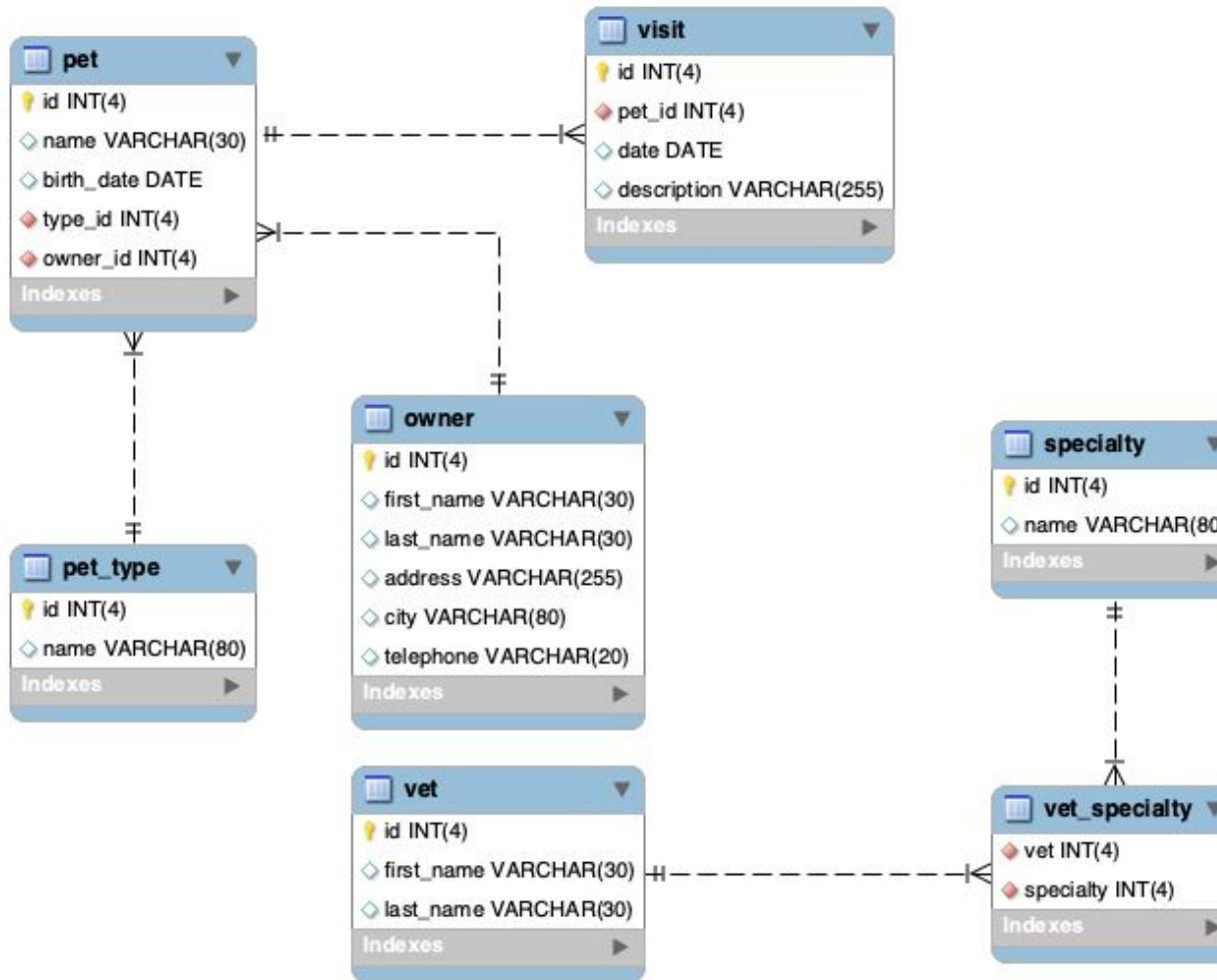
## **Use-Case:**

A User opens the owner detail page

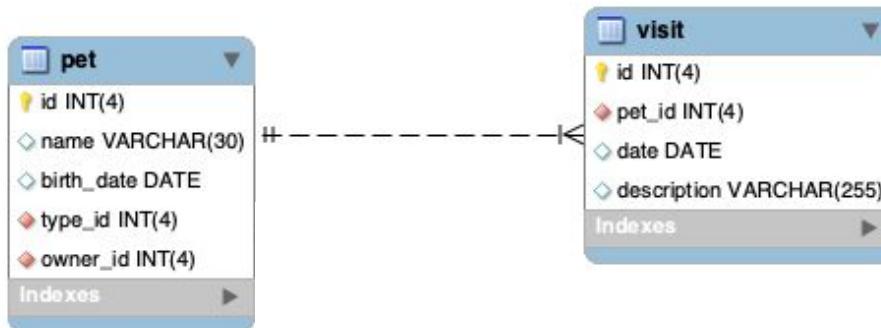
## **Workflow:**

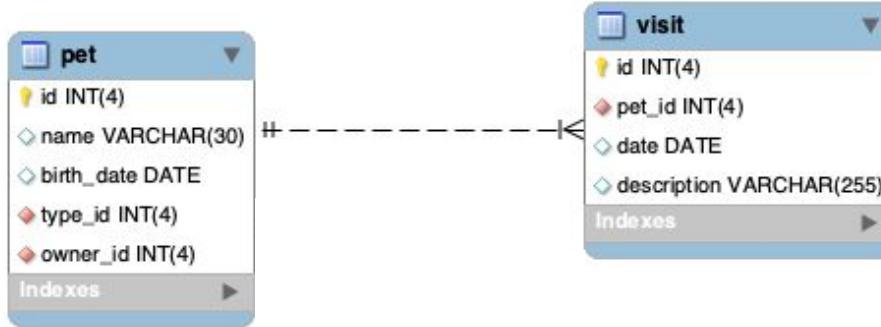
Find **pets** and any **visits** related to target pet using its identifier (pet\_id)

# SQL



# Pet to Visit - SQL





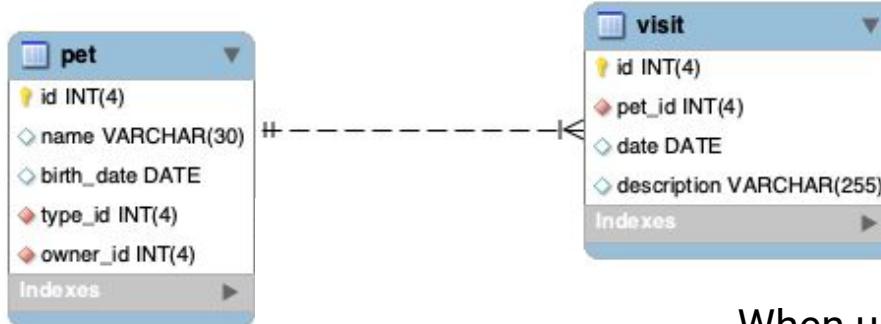
## Pet to Visit - SQL

### One to Many relationship

A pet can have multiple visits, but a visit can only have one pet

relationship: **id** to **pet\_id** (foreign key)

# Pet to Visit - SQL



When using **multiple attributes**  
(visit\_date, description, last\_name,  
telephone)

petclinic_visit_by_pet		
pet_id	uuid	K
visit_id	uuid	C ↑
visit_date	date	
description	text	
last_name	text	
telephone	text	

Using a clustering column = **multi-row partition**

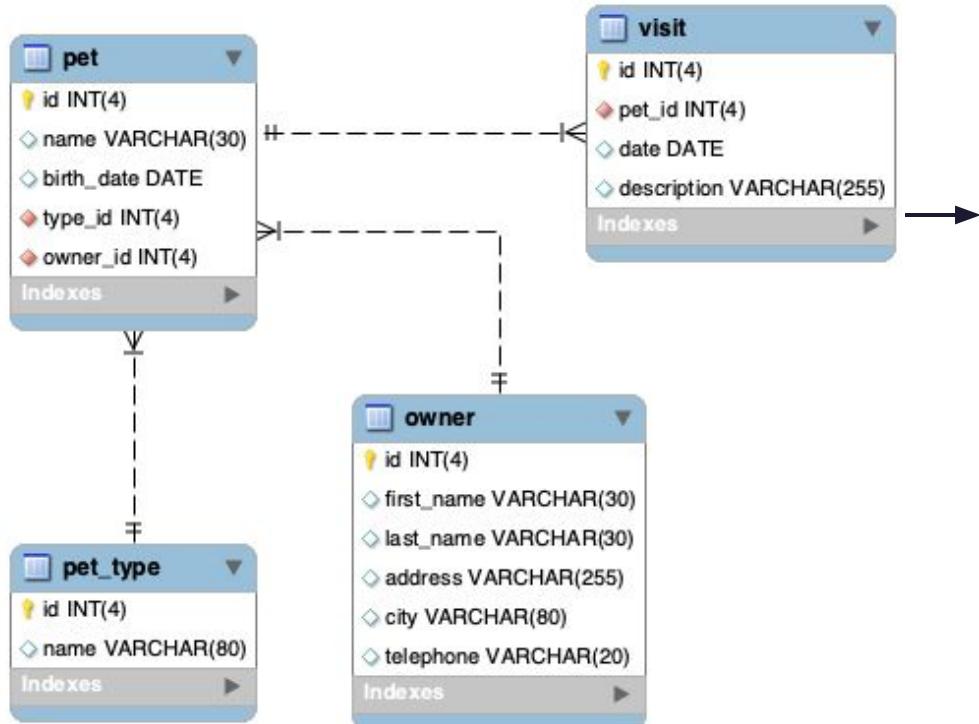
Means for every pet partition there can  
be **1 to n rows of visits** on a single read

Primary key (  
**Partition key(pet\_id)**  
**Clustering column(visit\_id)**)

# What's another option?

# Another option

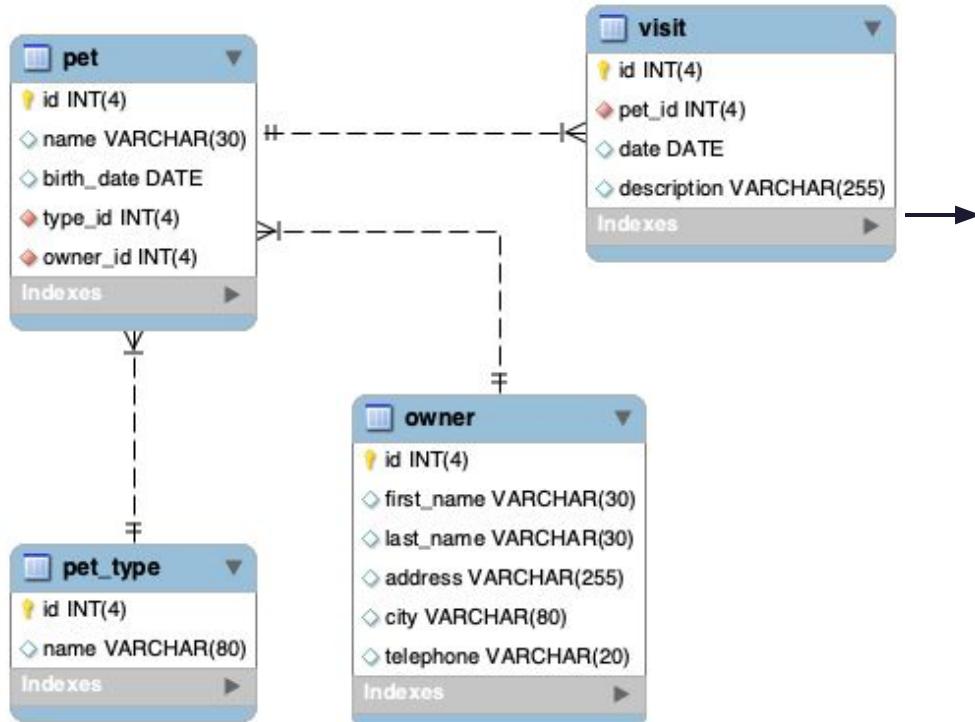
- Denormalize even more



petclinic_pet_by_owner		
owner_id	uuid	K
pet_id	uuid	C ↑
visit_id	uuid	C ↑
pet_birth_date	date	
pet_name	text	
pet_type	text	
owner_address	text	S
owner_city	text	S
owner_first_name	text	S
owner_last_name	text	S
owner_telephone	text	S
visit_date	date	
visit_description	text	

# Another option

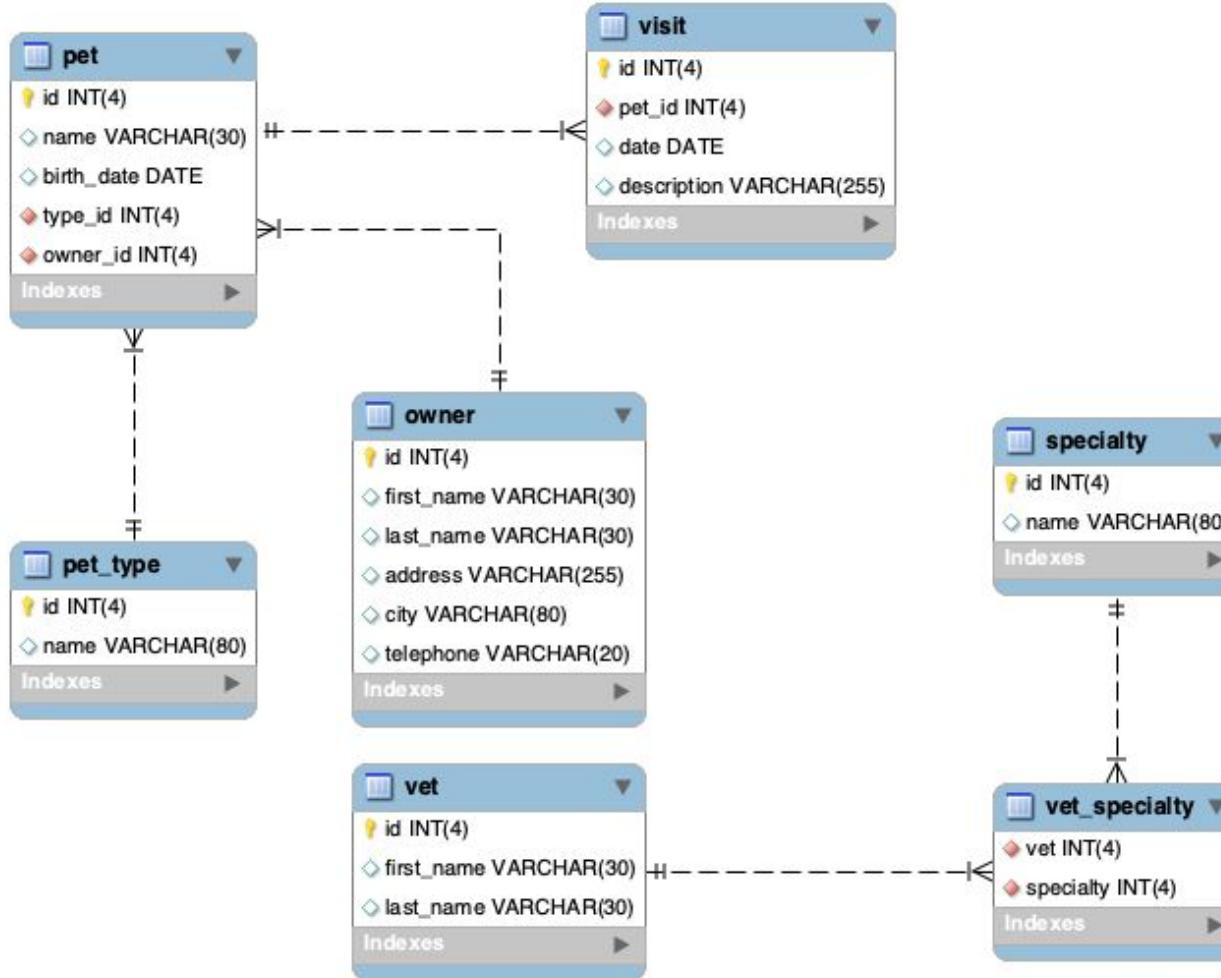
- Use a document



```
{  
  "data":{  
    "Joey":{  
      "firstname":"Joey",  
      "lastname":"Doe",  
      "weights":{  
        "reps":15,  
        "type":"bench press",  
        "weight":150  
      }  
    },  
    "Janet":{  
      "email":"janet.doe@gmail.com",  
      "favorite color":"grey",  
      "firstname":"Janet",  
      "lastname":"Doe"  
    },  
    "3ffc7ae6-c42d-46de-b52b-b5e77ae6a87b":{  
      "id":"some-stuff",  
      "other":"This is nonsensical stuff."  
    }  
  }  
}
```

# Final models

SQL to NoSQL



petclinic_owner		
<b>id</b>	uuid	K
address	text	
city	text	
first_name	text	
last_name	text	
telephone	text	

petclinic_pet_by_owner		
<b>owner_id</b>	uuid	K
<b>pet_id</b>	uuid	C↑
birth_date	date	
name	text	
pet_type	text	

petclinic_visit_by_pet		
<b>pet_id</b>	uuid	K
visit_id	uuid	C↑
visit_date	date	
description	text	
last_name	text	
telephone	text	

petclinic_vet		
<b>id</b>	uuid	K
first_name	text	
last_name	text	
specialties	set<text>	

petclinic_vet_by_specialty		
<b>specialty</b>	text	K
<b>vet_id</b>	uuid	C↑
first_name	text	
last_name	text	

petclinic_reference_lists		
<b>list_name</b>	text	K
values	set<text>	

# HandsOn #2 Create Data Model



DataStax

Astra

Get your instance here:

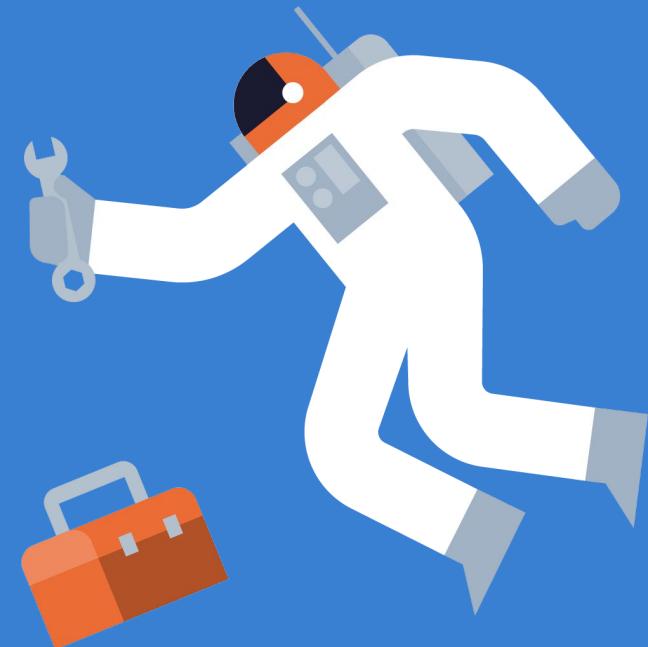
- [dtsx.io/workshop](https://dtsx.io/workshop)



GitHub

Repository:

- [dtsx.io/sql-to-nosql](https://dtsx.io/sql-to-nosql)



# Agenda

**01**

Why switch from  
SQL to NoSQL?

**02**

Three migration  
approaches

**03**

Data modeling SQL  
vs NoSQL

**04**

Migrate your data  
model

**05**

Moving data with  
DSBulk

**06**

Resources

# Moving data with DSBulk

# HandsOn #3 and #4



DataStax

Astra

**Get your instance here:**

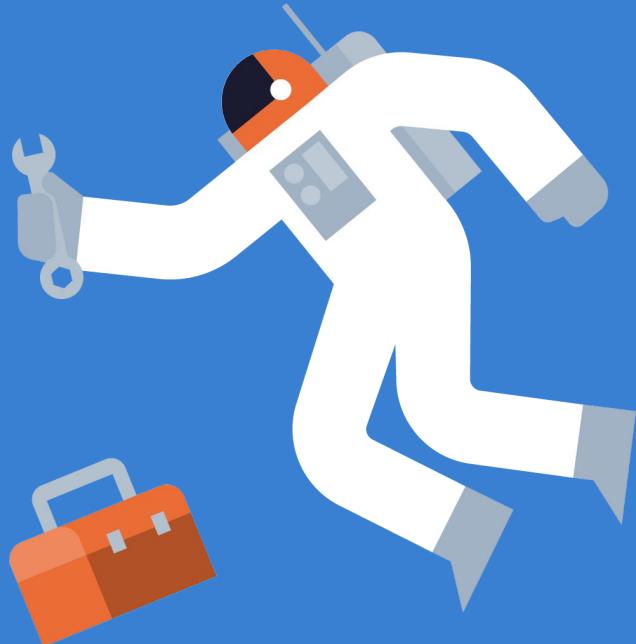
- [dtsx.io/workshop](https://dtsx.io/workshop)



GitHub

**Repository:**

- [dtsx.io/sql-to-nosql](https://dtsx.io/sql-to-nosql)



# menti.com



Go to [www.menti.com](http://www.menti.com) and use the code 3491 9972

## Inequality predicates are allowed on ...

A bar chart titled "Inequality predicates are allowed on ...". The y-axis represents the number of allowed inequality predicates, ranging from 1 to 15. The x-axis categories are: "All table columns" (blue bar, value 4), "Partition key columns" (yellow bar, value 3), "clustering key columns" (green bar, value 15, marked with a green checkmark), and "No inequality predicates are allowed" (pink bar, value 1, marked with a red X). The chart is set against a background of a video conference interface showing two participants.

Big paycheck

2:10:19 / 2:26:05

Go to [www.menti.com](http://www.menti.com) and use the code 3491 9972

## Leaderboard

4821 p		spanda
4820 p		Agent X9
4775 p		Sam
4711 p		CCedrickThePresenter
4468 p		shubham
4371 p		aaa
3895 p		vignesh
3877 p		adry
3861 p		Millie
3812 p		Puggie

2:11:07 / 2:26:05

DataStax Developers 101

# Agenda

**01**

Why switch from  
SQL to NoSQL?

**02**

Three migration  
approaches

**03**

Data modeling SQL  
vs NoSQL

**04**

Migrate your data  
model

**05**

Moving data with  
DSBulk

**06**

Resources

## **Awesome-cassandra - Cassandra from Relational**

- Multiple blogs
  - <https://github.com/anant/awesome-cassandra#cassandra-from-relational>

## **Best Practices for Migrating from a Relational Data Platform to Apache Cassandra™**

- Blog
  - <https://www.datastax.com/blog/best-practices-migrating-relational-data-platform-apache-cassandra>
- Free on-demand webinar
  - <https://www.datastax.com/resources/webinar/best-practices-migrating-relational-data-platform-apache-cassandra>

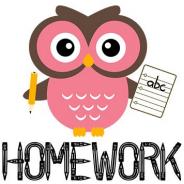
## **Migrating RDBMS Monoliths Into a Transformational Data Architecture**

- Free on-demand webinar
  - <https://www.datastax.com/resources/webinar/migrating-rdbms-monoliths-transformational-data-architecture>

# Developers

The screenshot shows the DataStax Developers YouTube channel page. At the top, there's a navigation bar with tabs for Academy, DataStax, Community, Events, Live Coding, About, and Login or Sign Up. Below the navigation is a large banner with the text "LEVEL UP with the DataStax Developers". The main content area features a "Leaderboard" section with a list of names and profile pictures, and a "Level up with the DataStax Developers!" video thumbnail. Below these are sections for "Upcoming live streams" featuring various workshops and a "Bootiful Cassandra" session.

- ✓ [Academy.datastax.com](https://academy.datastax.com)
- ✓ [datastax.com/dev](https://datastax.com/dev)
- ✓ [community.datastax.com](https://community.datastax.com)
- ✓ **Datastax Developers  
YouTube Channel**



# Homework ([dtsx.io/sql-to-nosql](https://dtsx.io/sql-to-nosql))

[Complete hands-on through to #4]

- Use DSbulk to load your SQL data into your NoSQL Cassandra data model

[Try other content] –

<https://www.datastax.com/dev/scenario/try-it-out-cassandra-data-modeling>

- Go to [datastax/dev](https://datastax.com/dev) and use the try-it-out



**DataStax**

Products Success Learn Try For Free Q

---

DATASTAX FOR DEVELOPERS

**Learn How to Succeed with Apache Cassandra™**

Build your next-generation applications with the NoSQL database that has proven high performance, linear scalability and zero downtime across on-premises, hybrid, and multi-cloud environments.

**LEVEL UP**

Video: Level up with DataStax Developers!

**Try It Out**  
Write your first Cassandra query and get your hands on the technology.

[Try Out Cassandra](#)

**What is Cloud Native?**  
Learn how NoSQL, Apache Cassandra, and DataStax enable cloud-native data.

[Learn About Cloud Native](#)

**What is Cassandra?**  
Understand Cassandra's origins, features, benefits, and future possibilities.

[Learn About Cassandra](#)

# Certifications

<https://www.datastax.com/dev/certifications>



## COMING SOON! Apache Cassandra Operations in Kubernetes Certification

As teams work to containerize and deploy applications using Kubernetes, there's increasing interest in running Cassandra in Kubernetes alongside applications as well. We're developing a new certification program to help teams level up their skills to run Cassandra successfully in cloud-native deployments. The Apache Cassandra Operations in Kubernetes Certification will cover: running Cassandra in Docker containers, understanding how Cassandra maps to Kubernetes, and how to deploy and run Cassandra on Kubernetes using Kubernetes operators and other monitoring and management tools.

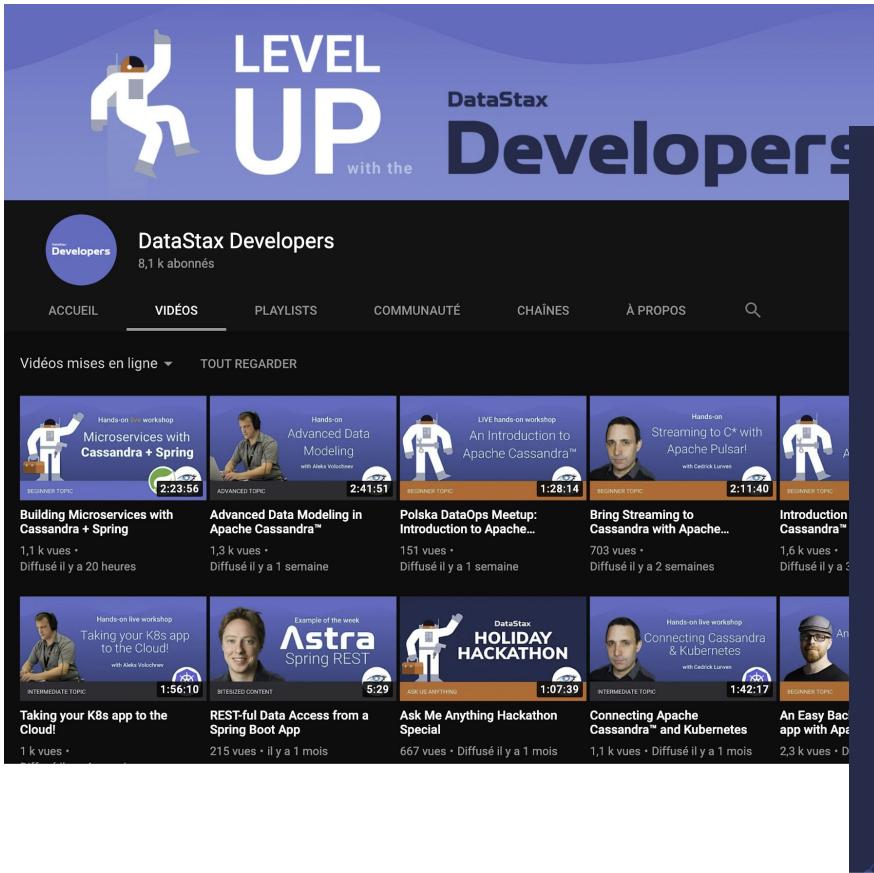
Sound interesting? Sign up to get notified about this program.

[SIGN UP NOW](#)

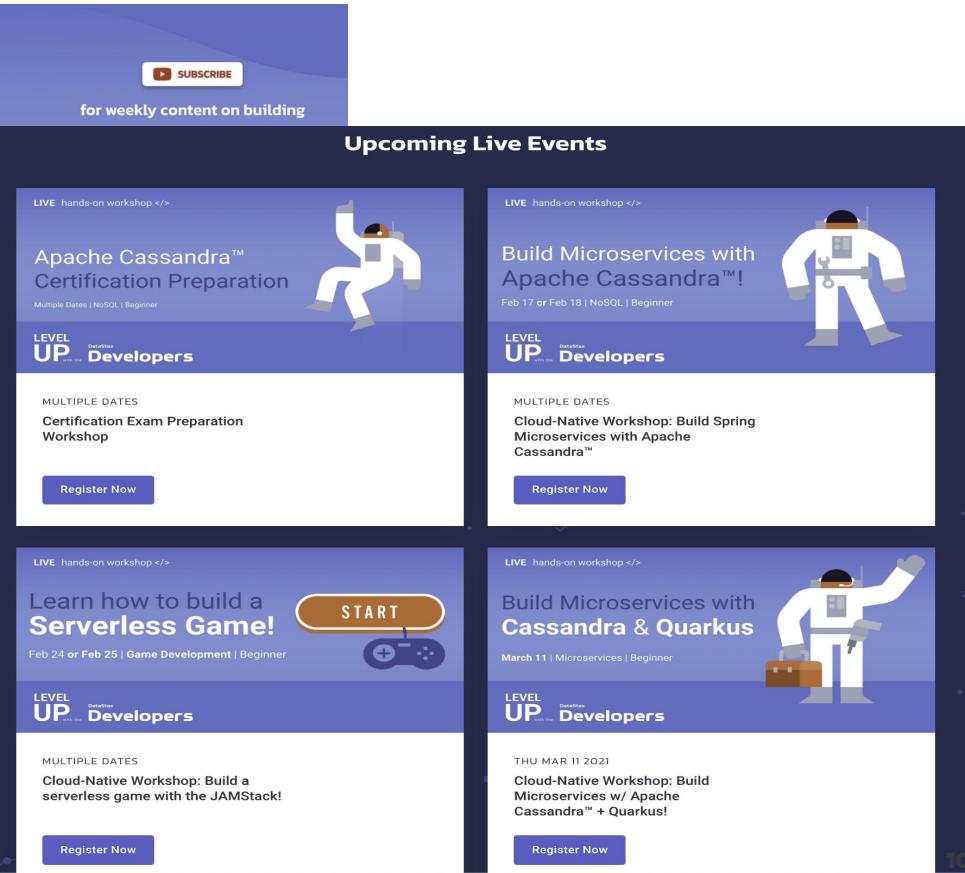
Vouchers (145\$ each), valid 3 months, with 2 attempts will be given to people who apply and register to the 3 episodes.

# Weekly Workshops

<https://www.datastax.com/workshops>



The screenshot shows the DataStax Developers YouTube channel homepage. The channel has 8,1k subscribers. The main banner features a white astronaut-like character climbing a wall, with the text "LEVEL UP with the DataStax Developers". Below the banner are several video thumbnails for various workshops, including "Microservices with Cassandra + Spring", "Advanced Data Modeling", and "Apache Cassandra™ Certification Preparation". A "SUBSCRIBE" button and a "for weekly content on building" message are visible.



The screenshot shows the DataStax Developers website's "Upcoming Live Events" section. It lists five events:

- Apache Cassandra™ Certification Preparation** (Multiple Dates | NoSQL | Beginner)
- Build Microservices with Apache Cassandra™!** (Feb 17 or Feb 18 | NoSQL | Beginner)
- Certification Exam Preparation Workshop** (Multiple Dates)
- Cloud-Native Workshop: Build Spring Microservices with Apache Cassandra™** (Multiple Dates)
- Learn how to build a Serverless Game!** (Feb 24 or Feb 25 | Game Development | Beginner)
- Build Microservices with Cassandra & Quarkus** (March 11 | Microservices | Beginner)

Each event listing includes a "Register Now" button and a small illustration of the white astronaut character.

# Join our 10k Discord Community

## The Fellowship of the RINGS

<https://bit.ly/cassandra-workshop>

thank you everyone! we hope to see you again in a future workshop! 🎉

Jack Fryer Aujourd'hui à 14:30  
NEW WORKSHOP ALERT

Cassandra meets Kubernetes!

Come and meet K8ssandra! A cloud-native distribution of Apache Cassandra™ built for running on Kubernetes

<https://www.eventbrite.co.uk/e/cloud-native-workshop-connecting-cassandra-and-kubernetes-tickets-142078180663>

Eventbrite

**Cloud-Native Workshop: Connecting Cassandra and Kubernetes!**

Come and meet K8ssandra! A cloud-native distribution of Apache Cassandra™ built for running on Kubernetes

LIVE hands-on workshop </>

**Apache Cassandra™ meets Kubernetes!**

March 3 or March 4 | Cloud-native | Beginner

LEVEL UP Developers

Cedrick Lunven Aujourd'hui à 16:18  
Hey Community I will be live tonight for 50 min talk on Spring + Cassandra to the Virtual Java User Group

<https://www.youtube.com/watch?v=nuyPKDQn1gl> come and say hi ?

Envoyer un message à #main-chat-room

PRESENTER—3

- Aleks Volochnev
- David Jones-Gilardi
- jscarp

HELPER—1

- John Sanda

EN LIGNE—222

- Abhiprada
- Absurdism
- hiya
- Adalberto
- aditya\_dhunna
- adnaneCord
- Adrigunz
- Aemilius Gaurav
- Aemilius gaurav
- Aguvas
- ajscilingo
- akashTaxvisor

# DataStax Developers

## Thank you!



@hadesarchitect  
@clun  
@SonicDMG



@hadesarchitect  
@clunven  
@SonicDMG



@hadesarchitect  
@clunven  
@david-gilardi

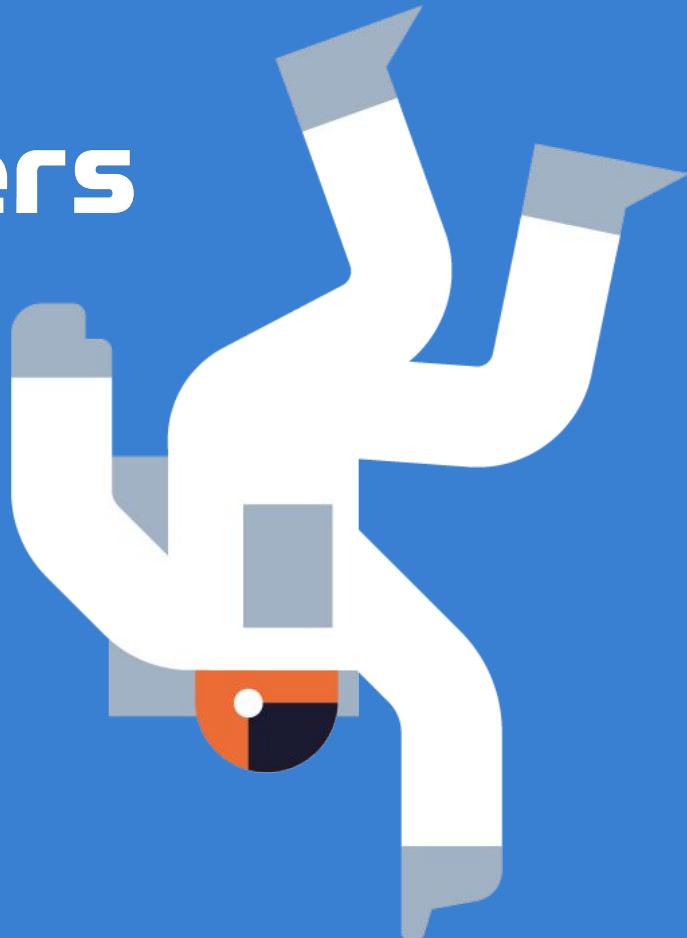


# DataStax Developers

Thank you!



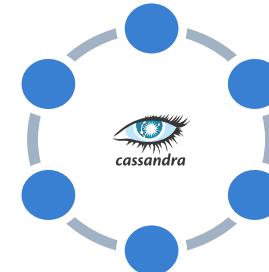
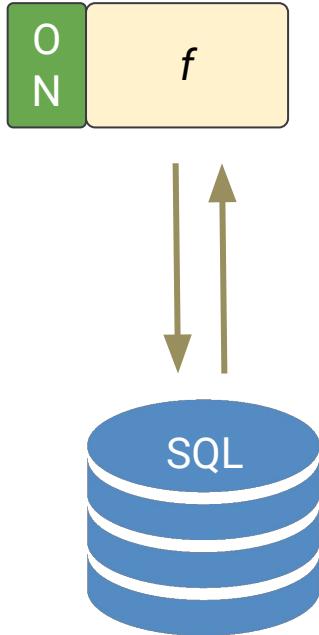
Subscribe



# Approach 2: Zero-Downtime Migration with CDC

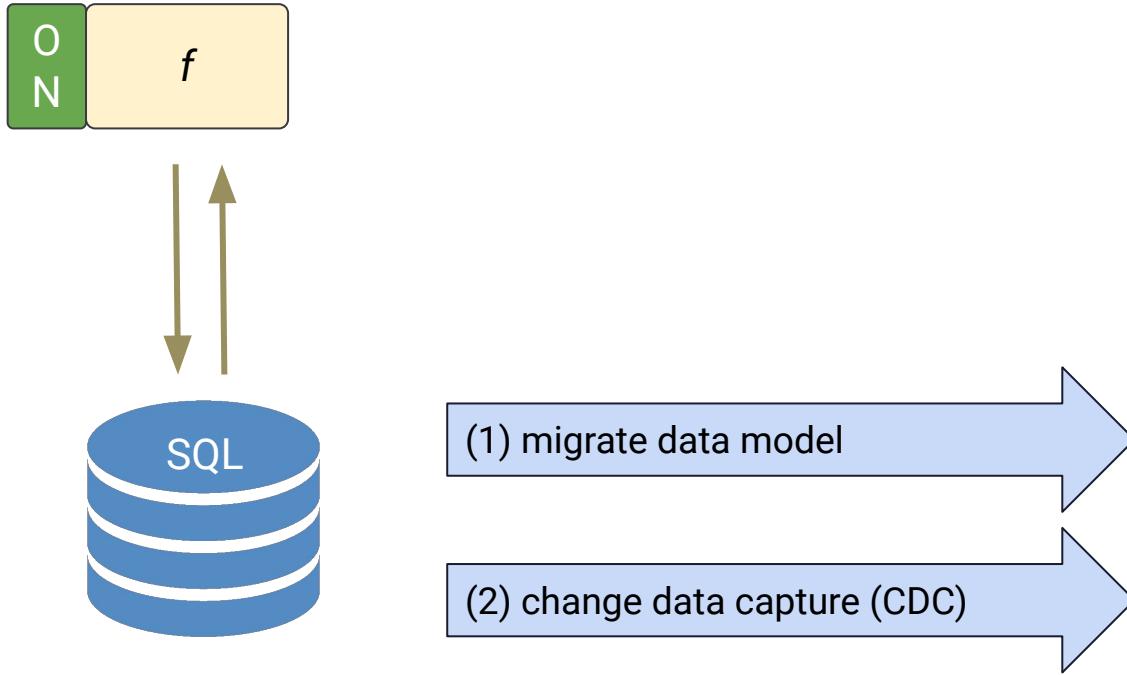
# Zero-Downtime Migration with CDC

Approach 2



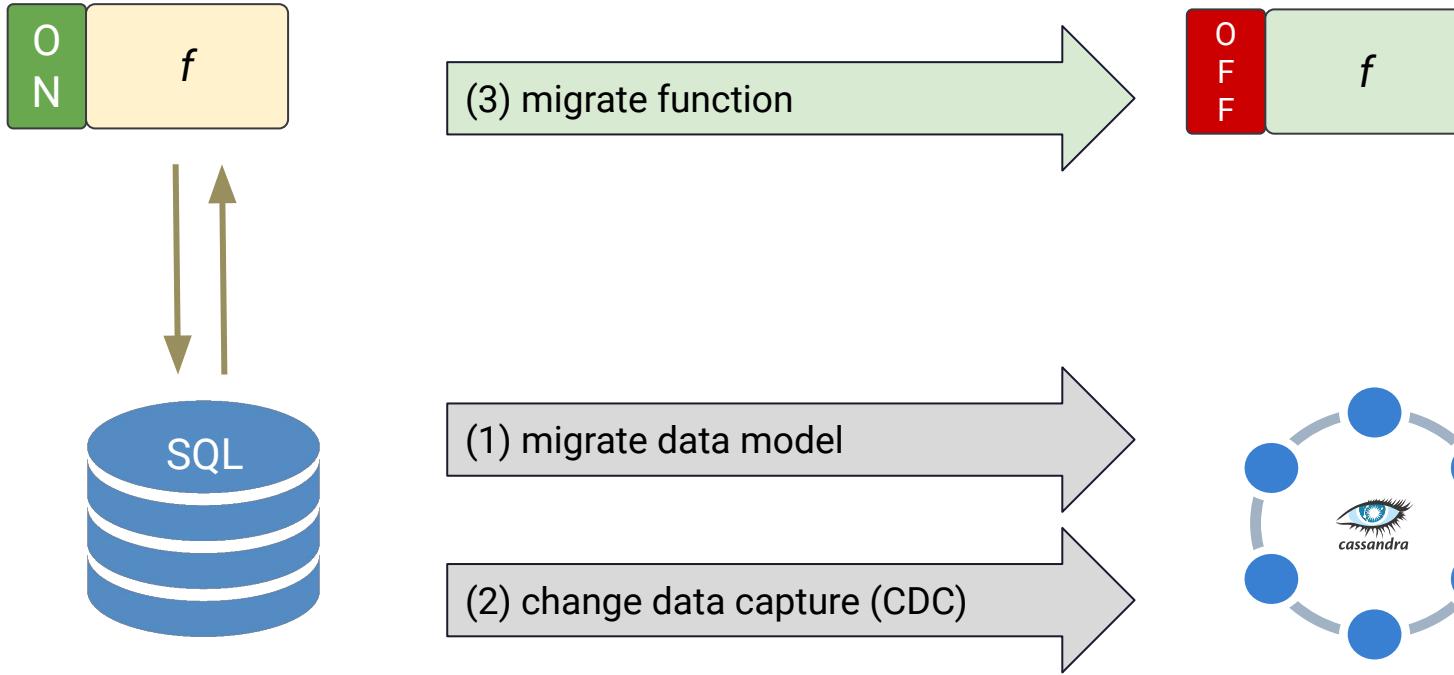
# Zero-Downtime Migration with CDC

Approach 2



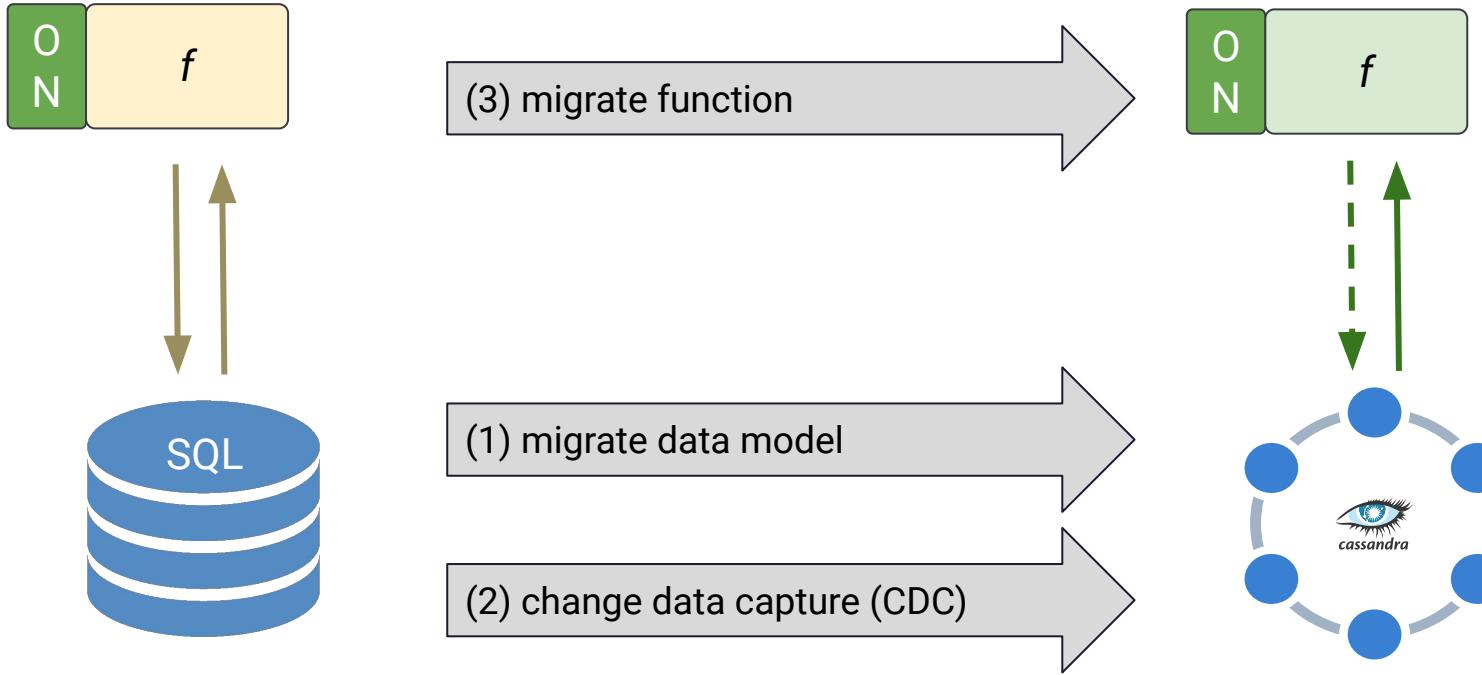
# Zero-Downtime Migration with CDC

Approach 2



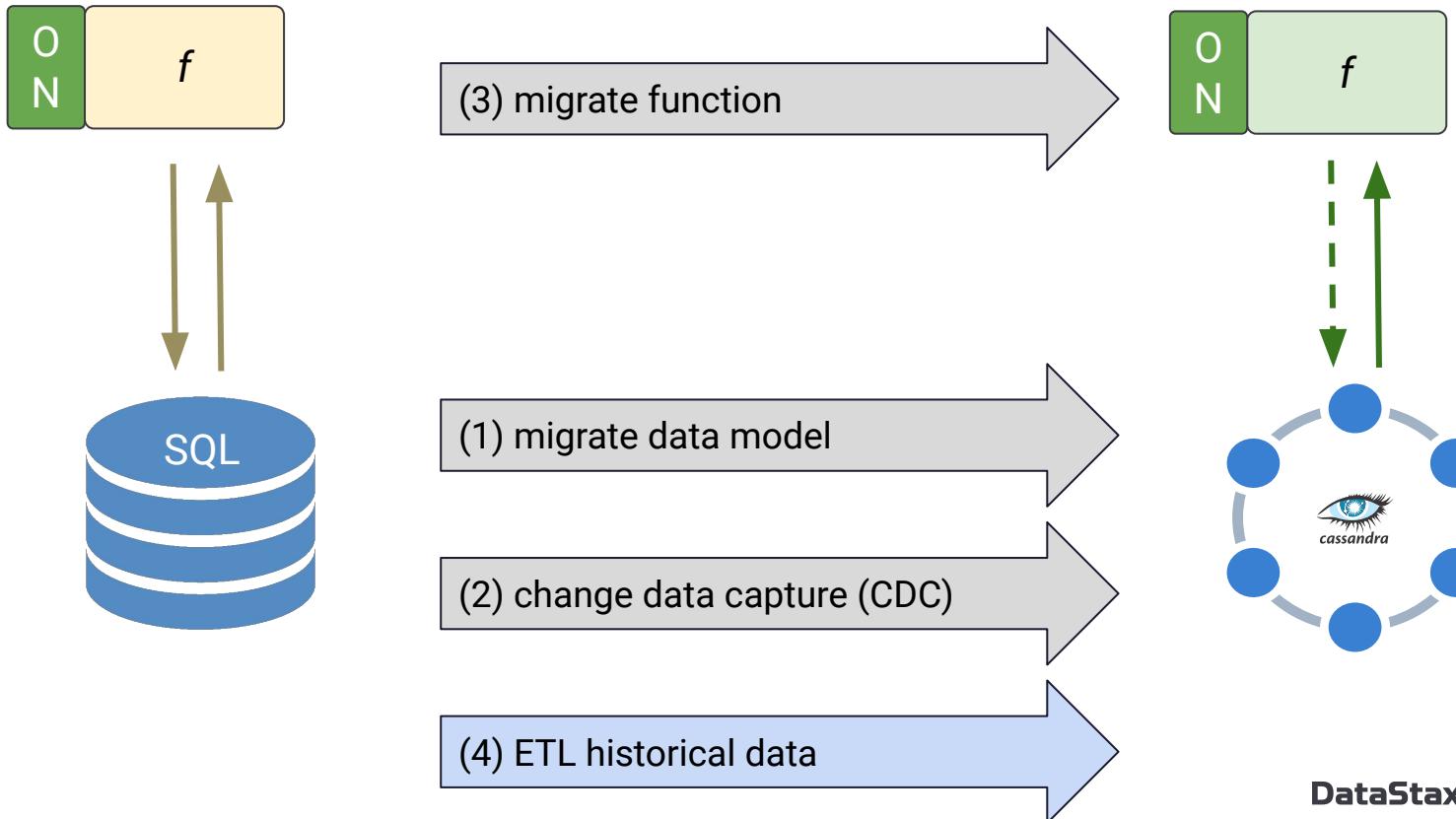
# Zero-Downtime Migration with CDC

Approach 2



# Zero-Downtime Migration with CDC

Approach 2



# Zero-Downtime Migration with CDC

Approach 2

