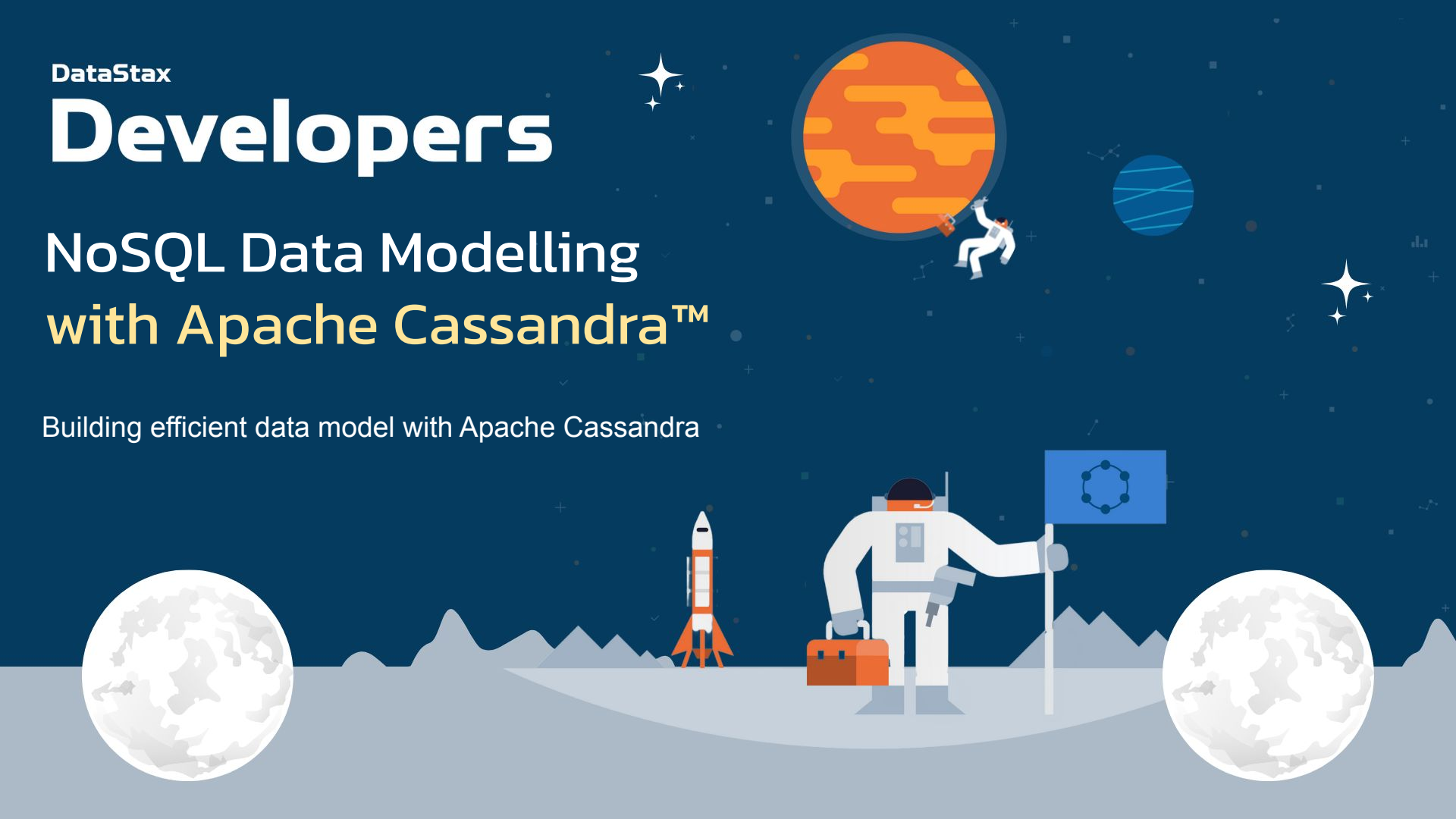
DataStax

Developers

NoSQL Data Modelling with Apache Cassandra™

Building efficient data model with Apache Cassandra



@aleks-volochnev

@HadesArchitect

- IT Exorcist
- Apache Cassandra™ certified
- Cloud Architect certified

Developer Advocate Lead

dtsx.io/aleks

cassandra

Aleksandr Volochnev

Developer Advocate

@stefano-lottini

@hemidactylus

@rsprrs

- Developer/Architect
- Apache Cassandra™ certified

- Background in computational physics
- Distributed systems
- Love to teach and communicate

dtsx.io/stefano

Stefano Lottini

Cedrick Lunven

Aleksandr Volochnev

Jack Fryer

Kirsten Hunter

Stefano Lottini

David Gilardi

Ryan Welford

Rags Srinivas

Sonia Siganporia

R

S

DataStax Developers Crew

01

Intro to Bootcamp 2022
HouseKeeping

02

Week I Refresh
Apache Cassandra in 2 words

03

Keys, Keys, Keys
How do you define keys

04

Partitioning in details
Rules of Partitioning

05

Modelling Methodology
Building efficient Data Model

06

What's next?
Quiz, Homework, Next week

Agenda

WEEK 1

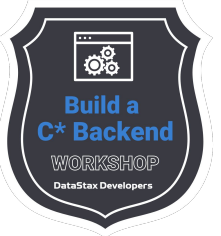
January 5th - January 11th

WEEK 2

January 12th - January 18th

WEEK 3

January 19th-January 25th



WEEK 4

January 26th-February 1st

Introducing 2022 Bootcamp

#1 Introduction to NoSQL Databases and Apache Cassandra™

- ~2H - Live lessons (same but different timezones)
  - January 5th 6 PM CET (= 9am PST = 11 am CST = 12 pm EST)
  - January 6th 9 AM CET (= 12am PST = 1am CST = 3am EST)
- ~3H - Homeworks
  - Hands-on with DBaas Cassandra service AstraDB
  - Learning materials: Cassandra Fundamentals exercices 1 to 8
  - Fill the week 1 test

WEEK 2

January 12h - January 18th

WEEK 3

January 19th-January 25th

WEEK 4

January 26th-February 1st

Introducing 2022 Bootcamp

#1 Introduction to NoSQL Databases and Apache Cassandra™

- ~2H - Live lessons (same but different timezones)
  - January 5th 6 PM CET (= 9am PST = 11 am CST = 12 pm EST)
  - January 6th 9 AM CET (= 12am PST = 1am CST = 3am EST)
- ~3H - Homeworks
  - Hands-on with DBaas Cassandra service AstraDB
  - Learning materials: Cassandra Fundamentals exercices 1 to 8
  - Fill the week 1 test

#2 Design Cassandra Data Models for Full-Stack Applications

- ~2H - Live lessons (twice the same for different timezones)
  - January 12th 6 PM CET (= 9am PST = 11 am CST = 12 pm EST)
  - January 13th 9 AM CET (= 12am PST = 1am CST = 3am EST)
- ~3H - Homeworks
  - Hands-on with DBaas Cassandra service AstraDB
  - Learning materials: Cassandra Fundamentals exercices 9 to 14
  - Learning materials: DS220 Modules 1 to 20 (Table Features)
  - Fill the week 2 test

WEEK 3

January 19th-January 25th

WEEK 4

January 26th-February 1st

Introducing 2022 Bootcamp

#1 Introduction to NoSQL Databases and Apache Cassandra™

- ~2H - Live lessons (same but different timezones)
  - January 5th 6 PM CET (= 9am PST = 11 am CST = 12 pm EST)
  - January 6th 9 AM CET (= 12am PST = 1am CST = 3am EST)
- ~3H - Homeworks
  - Hands-on with DBaas Cassandra service AstraDB
  - Learning materials: Cassandra Fundamentals exercices 1 to 8
  - Fill the week 1 test

#2 Design Cassandra Data Models for Full-Stack Applications

- ~2H - Live lessons (twice the same for different timezones)
  - January 12th 6 PM CET (= 9am PST = 11 am CST = 12 pm EST)
  - January 13th 9 AM CET (= 12am PST = 1am CST = 3am EST)
- ~3H - Homeworks
  - Hands-on with DBaas Cassandra service AstraDB
  - Learning materials: Cassandra Fundamentals exercices 9 to 14
  - Learning materials: DS220 Modules 1 to 20 (Table Features)
  - Fill the week 2 test

#3 Build a Full-Stack Backend with a NoSQL database

- ~2H - Live lessons (same but different timezones)
  - January 19th 6 PM CET (= 9am PST = 11 am CST = 12 pm EST)
  - January 20th 9 AM CET (= 12am PST = 1am CST = 3am EST)
- ~3H - Homeworks
  - Coding with DBaas Cassandra service AstraDB
  - DS220 Modules 21 to 40 (Collections-> Physical Data Model)
  - Fill the week 3 test

WEEK 4

January 26th-February 1st

Introducing 2022 Bootcamp

#1 Introduction to NoSQL Databases and Apache Cassandra™

- ~2H - Live lessons (same but different timezones)
  - January 5th 6 PM CET (= 9am PST = 11 am CST = 12 pm EST)
  - January 6th 9 AM CET (= 12am PST = 1am CST = 3am EST)
- ~3H - Homeworks
  - Hands-on with DBaas Cassandra service AstraDB
  - Learning materials: Cassandra Fundamentals exercices 1 to 8
  - Fill the week 1 test

#2 Design Cassandra Data Models for Full-Stack Applications

- ~2H - Live lessons (twice the same for different timezones)
  - January 12th 6 PM CET (= 9am PST = 11 am CST = 12 pm EST)
  - January 13th 9 AM CET (= 12am PST = 1am CST = 3am EST)
- ~3H - Homeworks
  - Hands-on with DBaas Cassandra service AstraDB
  - Learning materials: Cassandra Fundamentals exercices 9 to 14
  - Learning materials: DS220 Modules 1 to 20 (Table Features)
  - Fill the week 2 test

#3 Build a Full-Stack Backend with a NoSQL database

- ~2H - Live lessons (same but different timezones)
  - January 19th 6 PM CET (= 9am PST = 11 am CST = 12 pm EST)
  - January 20th 9 AM CET (= 12am PST = 1am CST = 3am EST)
- ~3H - Homeworks
  - Coding with DBaas Cassandra service AstraDB
  - DS220 Modules 21 to 40 (Collections-> Physical Data Model)
  - Fill the week 3 test

#4 Build Api and Microservices for Apache Cassandra™

- ~2H - Live lessons (same but different timezones)
  - January 26th 6 PM CET (= 9am PST = 11 am CST = 12 pm EST)
  - January 27th 9 AM CET (= 12am PST = 1am CST = 3am EST)
- ~3H - Homeworks
  - Coding with DBaas Cassandra service AstraDB
  - Learning materials: DS220 Modules 42 to 56
  - Fill the week 4 test

1 **Attend one of the 2 LIVE STREAMED workshop (Wednesday or Thursday)**

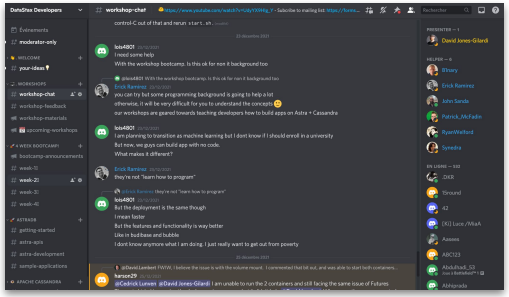
4-week bootcamp Housekeeping

Livestream: youtube.com/DataStaxDevs

YouTube

Twitch

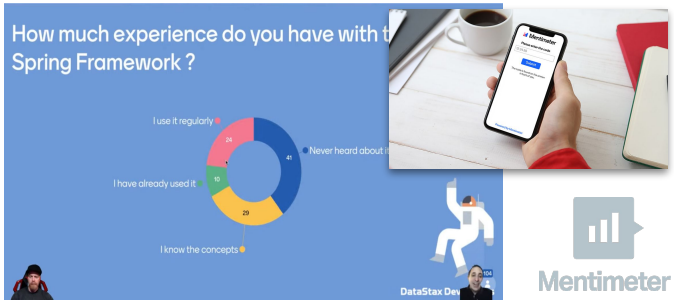
Questions: https://dtsx.io/discord



YouTube

Discord

Games and quizzes: menti.com



Mentimeter

1 Attend the live sessions

1. Attend one of the 2 LIVE STREAMED workshop (Wednesday or Thursday)
2. Complete the workshop labs

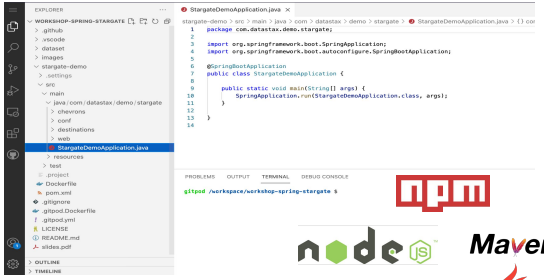
4-week bootcamp Housekeeping

AstraDB astra.datastax.com

Database + GraphQL + PlayGround

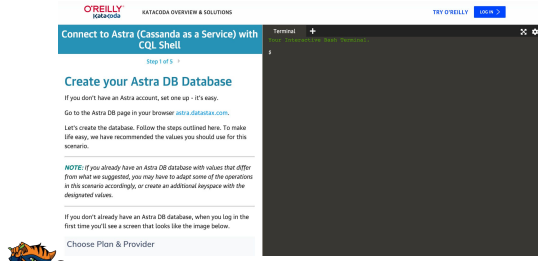
DataStax Astra DB

Gitpod IDE: https://gitpod.io

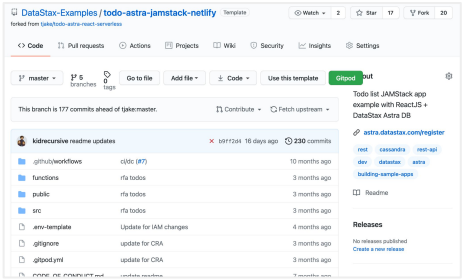


Gitpod

Katacoda katacoda.com/datastax



Katacoda



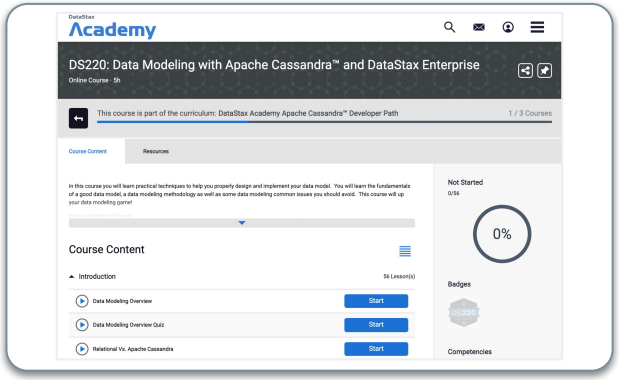
GitHub

2 Complete Workshops Labs

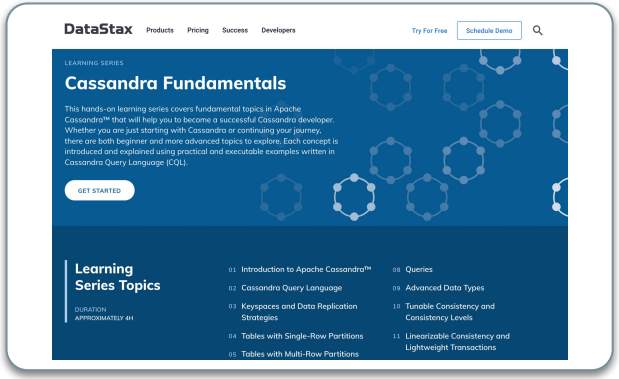
1. Attend one of the 2 LIVE STREAMED workshop (Wednesday or Thursday)
2. Complete the workshop labs
3. Complete the Learning materials

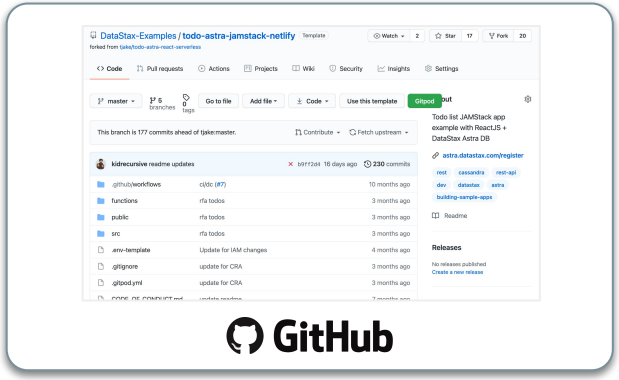
4-week bootcamp Housekeeping

Academy academy.datastax.com



Datastax /dev datastax.com/dev





3 **Do your homeworks**

1. Attend one of the 2 LIVE STREAMED workshop (Wednesday or Thursday)
2. Complete the workshop labs
3. Complete the Learning materials
4. Submit the homework (google form)

4-week bootcamp Housekeeping

Intro to NoSQL Homework

Welcome and thank you! Here you can submit your homework for the datastax developers "Intro to NoSQL" workshop. In case of any questions please contact organisers at https://dtsx.io/aleks or just send an email to aleksandr.volochnev@datastax.com

- Workshop materials: https://github.com/datastaxdevs/workshop-introduction-to-nosql
- Discord chat: https://dtsx.io/discord

cedrick.lunven@datastax.com Switch account

The name and photo associated with your Google account will be recorded when you upload files and submit this form. Only the email you enter is part of your response.

* Required

Email *

Your email

4 **Pass the Weekly Test**

menti.com


Apache
CASSANDRA
25
DS COMICS
DATASTAX

Go to www.menti.com and use the code 3491 9972
Inequality predicates are allowed on ...
4
3
15
1
All table columns
Partition key columns
clustering key columns
No inequality predicates are allowed
2:10:19 / 2:26:05

Go to www.menti.com and use the code 3491 9972
Leaderboard
4821 p spanda
4820 p Agent X9
fastest
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

01

Intro to Bootcamp 2022
HouseKeeping

02

Week I Refresh
Apache Cassandra in 2 words

03

Keys, Keys, Keys
How do you define keys

04

Partitioning in details
Rules of Partitioning

05

Modelling Methodology
Building efficient Data Model

06

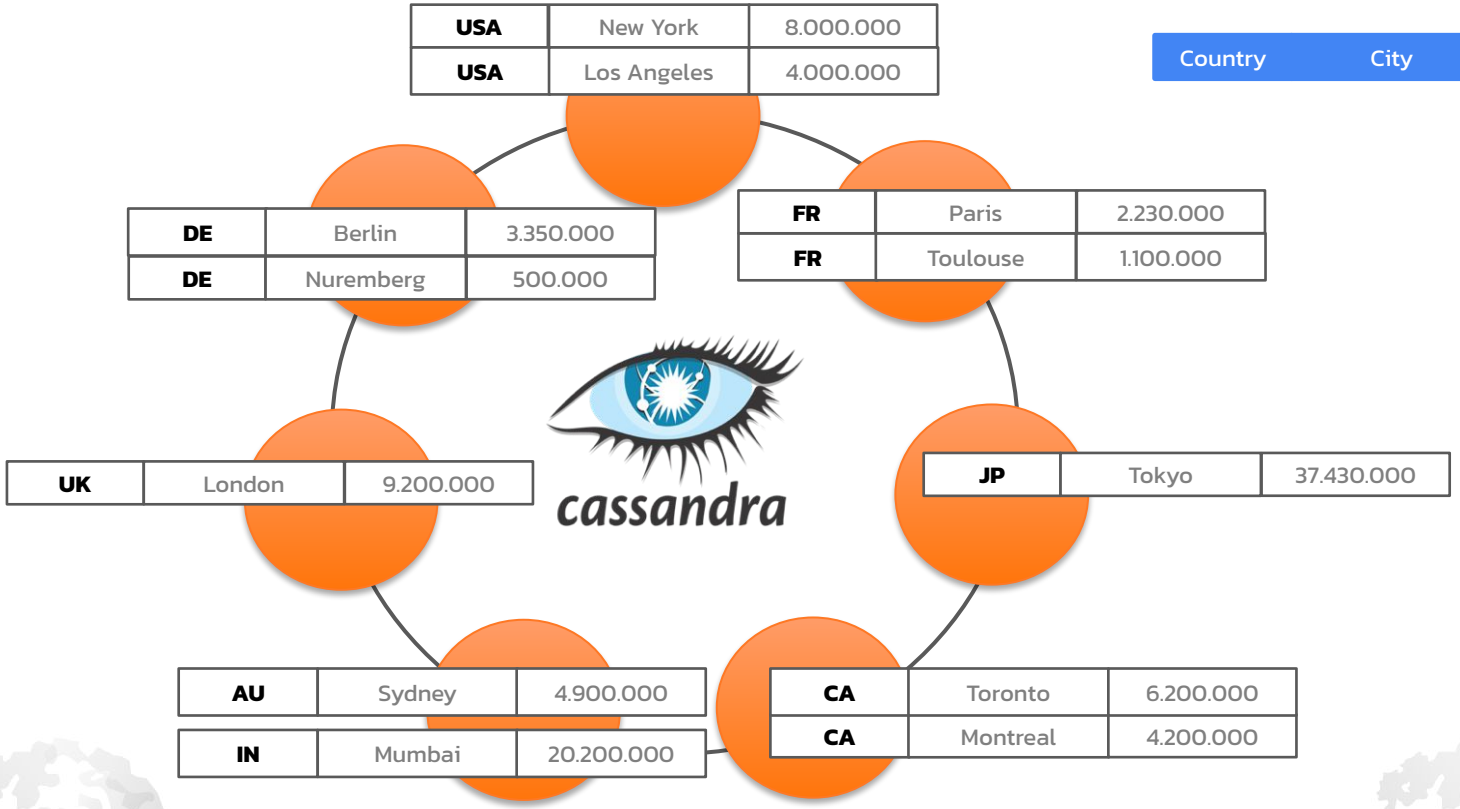
What's next?
Quiz, Homework, Next week

Agenda

Apache Cassandra

NoSQL Distributed Decentralised Database Management System

- Big Data Ready
- Read / Write Performance
- Linear Scalability
- Highest Availability
- Self-Healing and Automation
- Geographical Distribution
- Platform Agnostic
- Vendor Independent

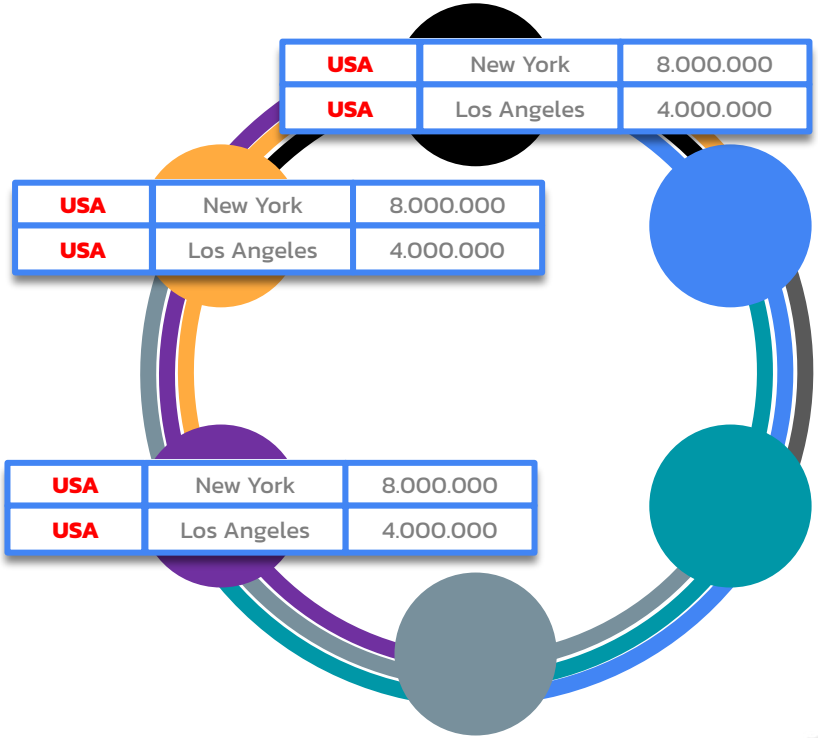


Country	City	Population

Data is Partitioned

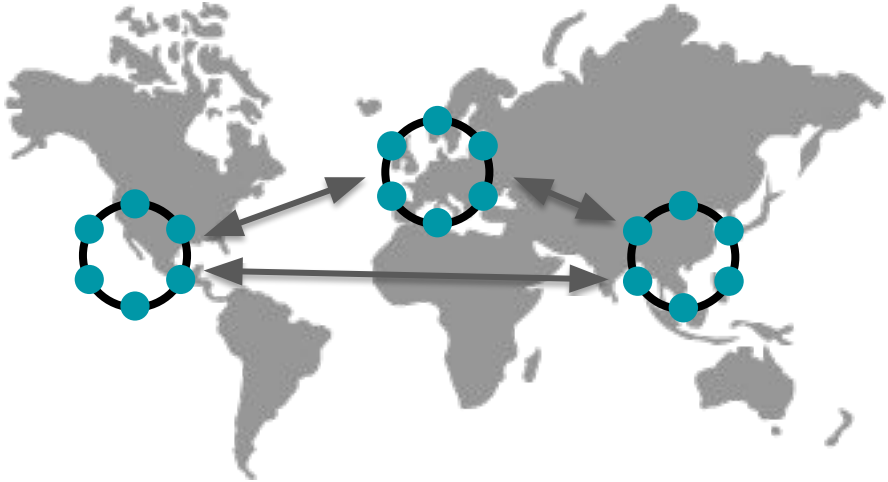
RF = 3

Replication Factor 3 means that every partition is stored on 3 nodes

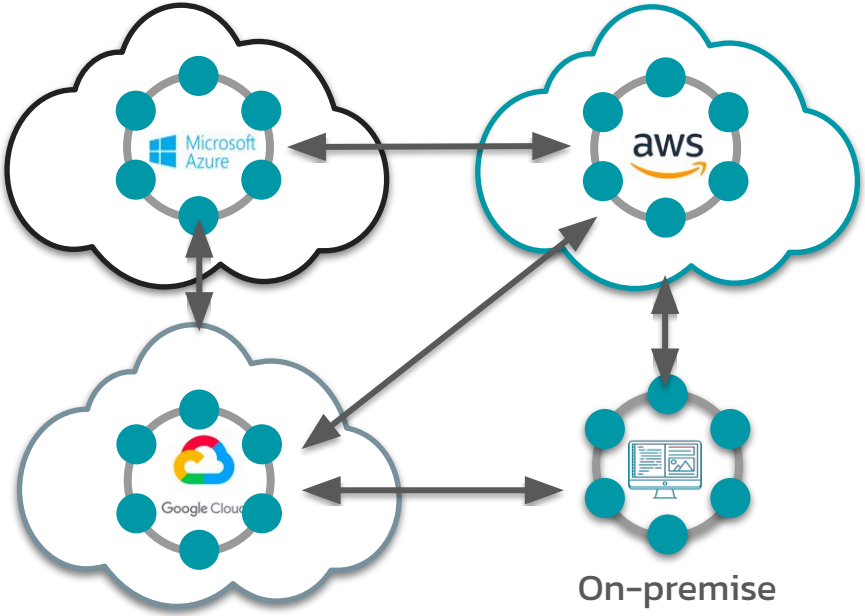


Data is Replicated

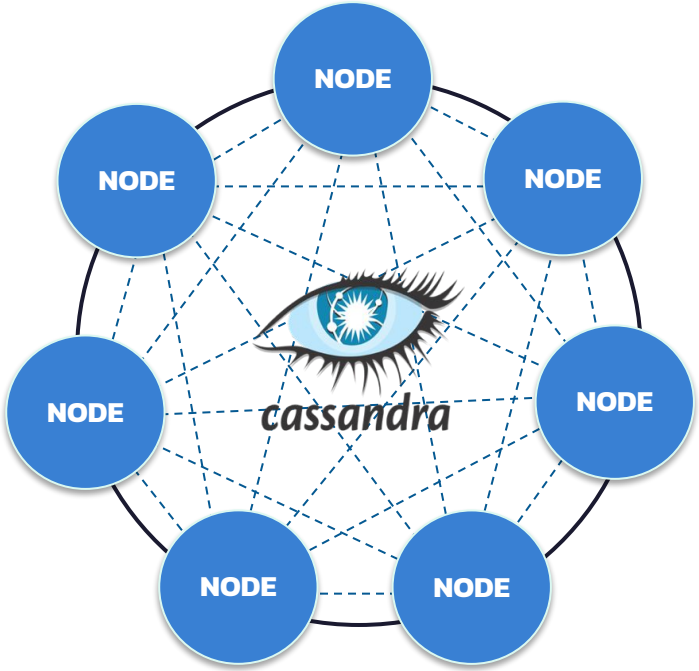
Geographical Distribution



Hybrid-Cloud and Multi-Cloud



Data is globally distributed



1. **NO** Single Point of Failure
2. Scales for writes and reads
3. Application can contact any node
   (in case of failure - just contact next one)

Master-less (Peer-to-Peer) Architecture

Cassandra’s scalability and performance is a shared responsibility. It gives you enough tools to build an incredibly performant and reliable database but you have to learn how to do it right!

01

Intro to Bootcamp 2022
HouseKeeping

02

Week I Refresh
Apache Cassandra in 2 words

03

Keys, Keys, Keys
How do you define keys

04

Partitioning in details
Rules of Partitioning

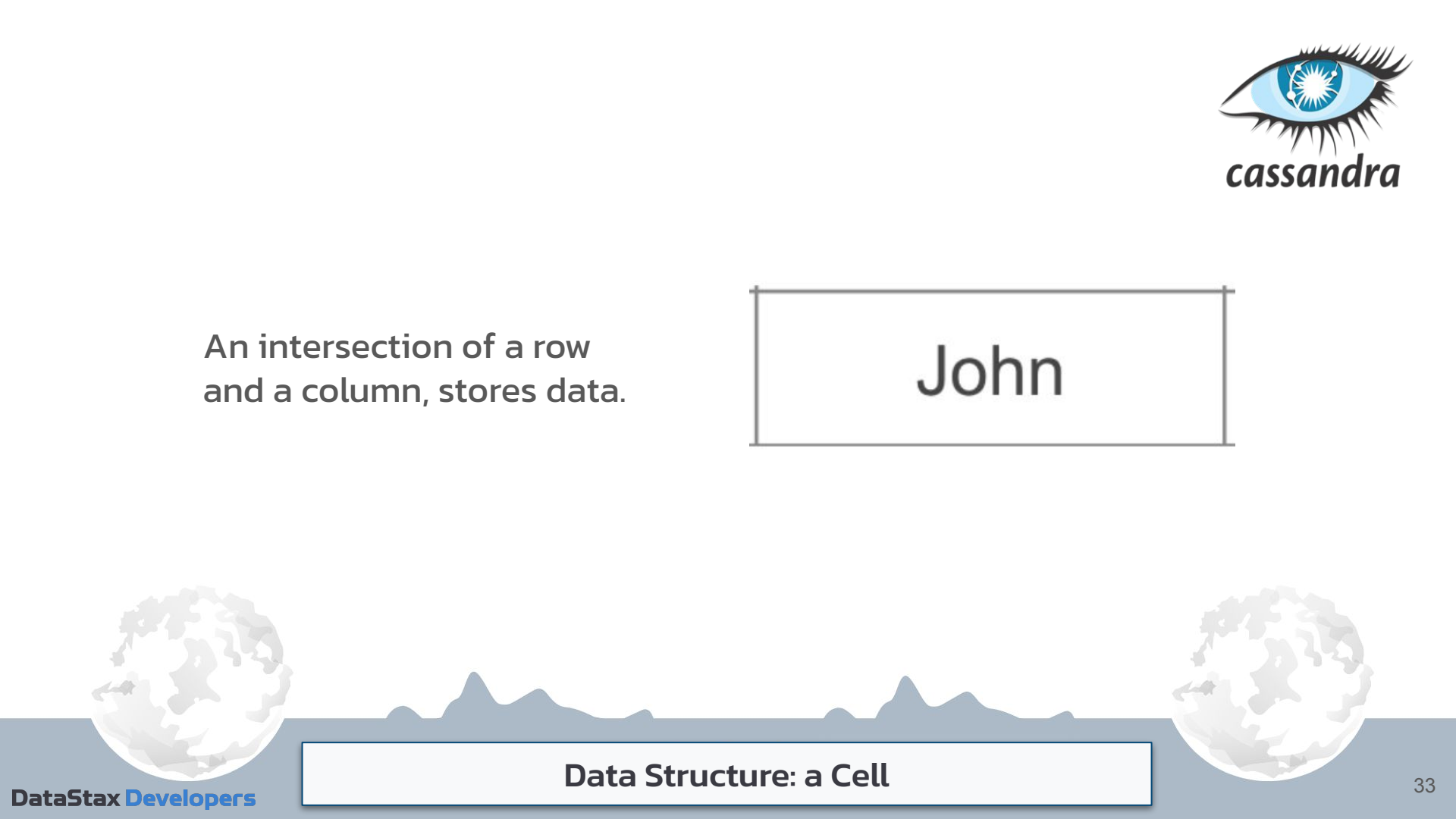
05

Modelling Methodology
Building efficient Data Model

06

What's next?
Quiz, Homework, Next week

Agenda



An intersection of a row and a column, stores data.

John

Data Structure: a Cell

A single, structured data item in a table.

1	John	Doe	Wizardry

Data Structure: a Row

A group of rows having the same partition token, a base unit of access in Cassandra.

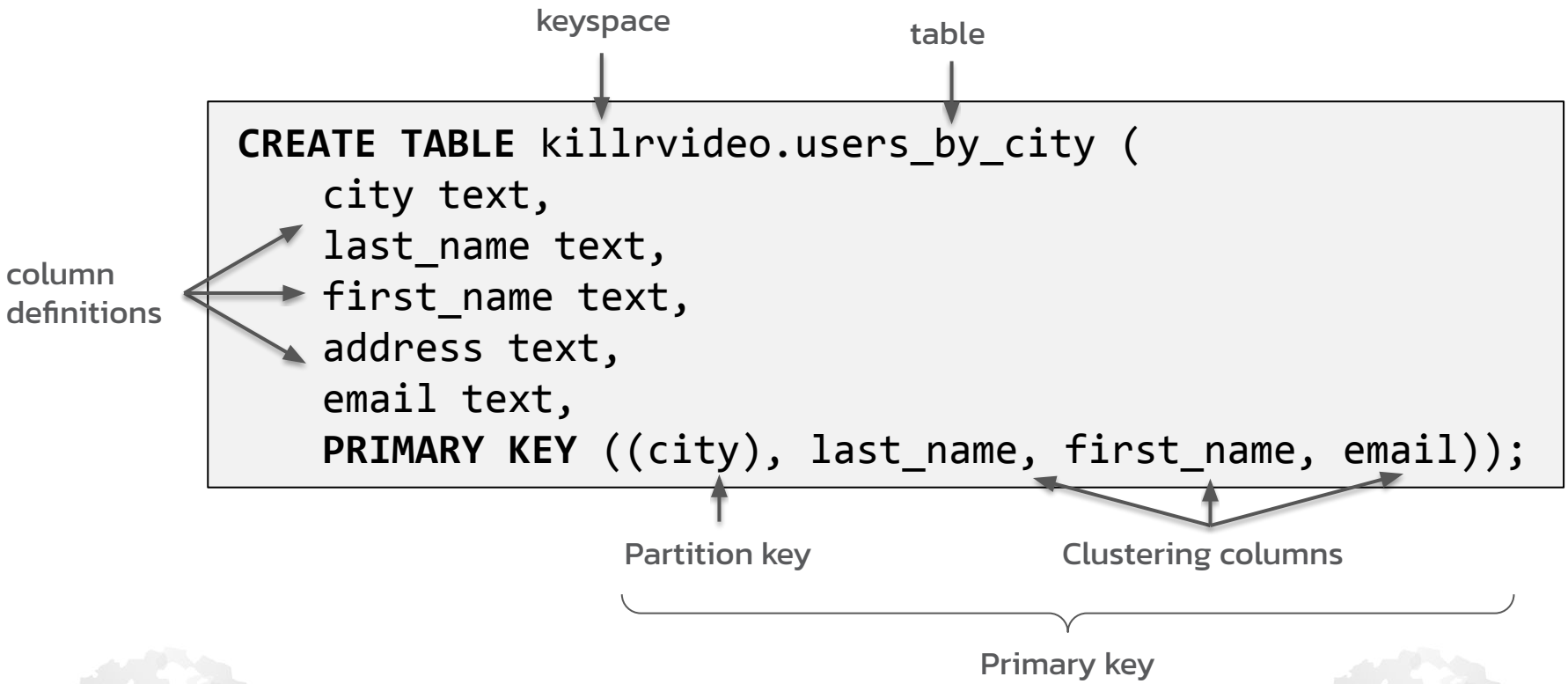
IMPORTANT: stored together, all the rows are guaranteed to be neighbors.

ID	First Name	Last Name	Department
1	John	Doe	Wizardry
399	Marisha	Chapez	Wizardry
415	Maximus	Flavius	Wizardry

A group of columns and rows storing partitions.

ID	First Name	Last Name	Department
1	John	Doe	Wizardry
2	Mary	Smith	Dark Magic
3	Patrick	McFadin	DevRel

Data Structure: a Table



Creating a table

An identifier for a partition.
Consists of at least one column, may have more if needed

PARTITIONS ROWS.

```
CREATE TABLE killrvideo.users_by_city (
      city text,
      last_name text,
      first_name text,
      address text,
      email text,
      PRIMARY KEY ((city), last_name, first_name, email));
```

Partition key

Examples:

```
PRIMARY KEY (user_id);
```

```
PRIMARY KEY ((video_id), comment_id);
```

```
PRIMARY KEY ((country, city), email);
```

Partition Key

Used to **ensure uniqueness and sorting order.** Optional.

```
CREATE TABLE killrvideo.users_by_city (
      city text,
      last_name text,
      first_name text,
      address text,
      email text,
      PRIMARY KEY ((city), last_name, first_name, email));
```

Clustering columns

```
PRIMARY KEY ((city), last_name, first_name);
```
Not Unique

```
PRIMARY KEY ((city), last_name, first_name, email);
```

```
PRIMARY KEY ((video_id), comment_id);
```
Not Sorted

```
PRIMARY KEY ((video_id), created_at, comment_id);
```

Clustering Columns

An identifier for a row. Consists of at least one Partition Key and zero or more Clustering Columns.

MUST ENSURE UNIQUENESS.

```
CREATE TABLE killrvideo.users_by_city (
      city text,
      last_name text,
      first_name text,
      address text,
      email text,
      PRIMARY KEY ((city), last_name, first_name, email));
```

Primary key

Examples:

```
PRIMARY KEY ((city), last_name, first_name, email);
```

```
PRIMARY KEY (user_id);
```

Primary Key

```
SELECT * FROM  users_by_city ...
```

```
    WHERE last_name = ?;
```

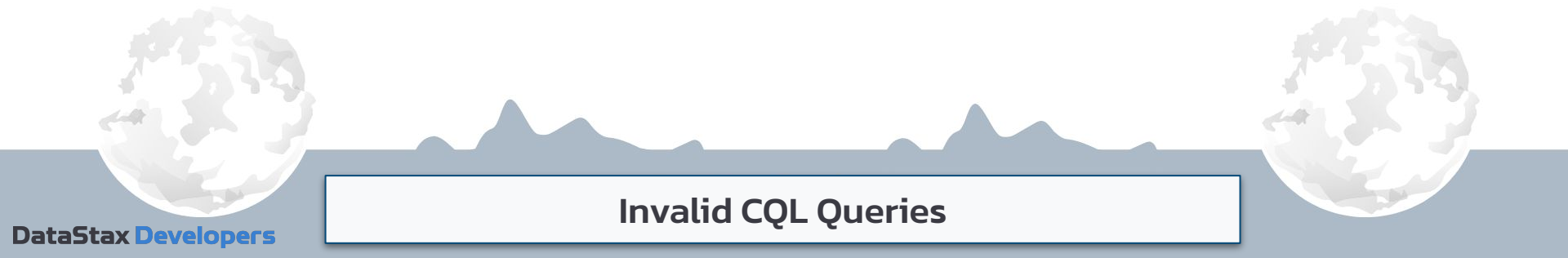
```
    WHERE city > ?;
```

```
    WHERE city > ? AND city < ?;
```

```
    WHERE city = ? AND first_name = ? AND email = ?;
```

```
    WHERE city = ? AND last_name > ? AND first_name = ?;
```

```
PRIMARY KEY ((city), last_name, first_name, email));
```



Invalid CQL Queries

QUESTION

Why don’t we use sequential IDs in Cassandra?

[My]SQL

```
CREATE TABLE table_name (
  column1 integer NOT NULL AUTO_INCREMENT,
  …
);
```

```
mysql> INSERT INTO table_name …
mysql> SELECT LAST_INSERT_ID();

Result: 99 (last value + 1)
```

Usually ID generated by a database

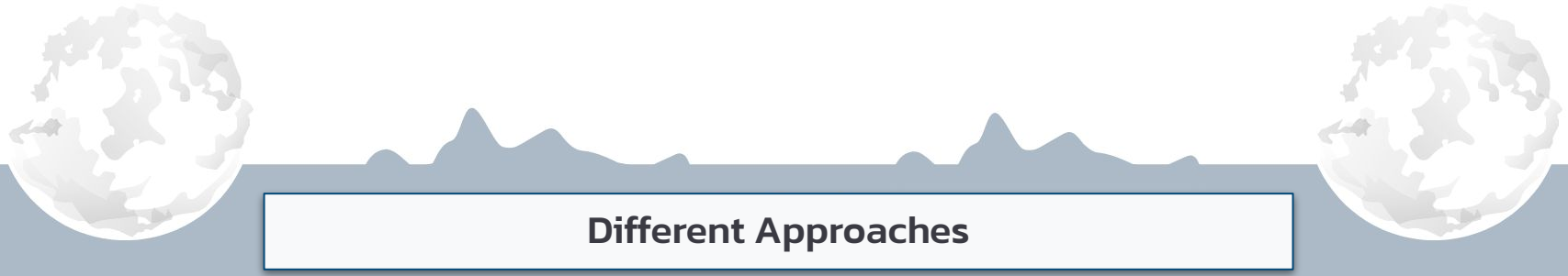
Cassandra

```
CREATE TABLE todoitems (
   item_id UUID,
   …
);
```

```
cql> INSERT INTO todoitems ( item_id )
       VALUES ( UUID() );
cql> SELECT item_id FROM todoitems …

Result: 123e4567-e89b-12d3-a456-426614174000
```

Usually UUID generated in an application code



Different Approaches

01

Intro to Bootcamp 2022
HouseKeeping

02

Week I Refresh
Apache Cassandra in 2 words

03

Keys, Keys, Keys
How do you define keys

04

Partitioning in details
Rules of Partitioning

05

Modelling Methodology
Building efficient Data Model

06

What's next?
Quiz, Homework, Next week

Agenda

The Slide of the Year Award!

- **Store together what you retrieve together**
- Avoid big partitions
- Avoid hot partitions

Example: open a video? Get the comments in a single query!

```
PRIMARY KEY ((video_id), created_at, comment_id);
```

```
PRIMARY KEY ((comment_id), created_at);
```

- Store together what you retrieve together
- **Avoid big partitions**
- Avoid hot partitions

The Slide of the Year Award!

```
PRIMARY KEY ((video_id), created_at, comment_id);
```

```
PRIMARY KEY ((country), user_id);
```

- Up to 2 billion cells per partition
- Up to ~100k rows in a partition
- Up to ~100MB in a Partition

Rules of a Good Partition

The Slide of the Year Award!

- Store together what you retrieve together
- Avoid big partitions
- **Avoid hot partitions**

```
PRIMARY KEY (user_id);
```

```
PRIMARY KEY ((video_id), created_at, comment_id);
```

```
PRIMARY KEY ((country), user_id);
```

Rules of a Good Partition

QUESTION

Danger is not always obvious.
What's wrong with the next model?

The Slide of the Year Award!

- Store together what you retrieve together
- Avoid big partitions
- Avoid hot partitions

CASE: a huge IoT infrastructure, hardware all over the world, different sensors reporting their state every 10 seconds. Every sensor reports its UUID, timestamp of the report, sensor's value.

- Sensor ID: UUID
- Timestamp: Timestamp
- Value: float

```
PRIMARY KEY ((sensor_id), reported_at);
```

Rules of a Good Partition

The Slide of the Year Award!

- Store together what you retrieve together
- Avoid big partitions
- Avoid hot partitions

CASE: a huge IoT infrastructure, hardware all over the world, different sensors reporting their state **every 10 seconds**. Every sensor reports its UUID, timestamp of the report, sensor's value.

- Sensor ID: UUID
- Timestamp: Timestamp
- Value: float

```
PRIMARY KEY ((sensor_id), reported_at);
```

Rules of a Good Partition

The Slide of the Year Award!

- Store together what you retrieve together
- **Avoid big and constantly growing partitions!**
- Avoid hot partitions

Example: a huge IoT infrastructure, hardware all over the world, different sensors reporting their state every 10 seconds. Every sensor reports its UUID, timestamp of the report, sensor's value.

```
PRIMARY KEY ((sensor_id), reported_at);
```

```
PRIMARY KEY ((sensor_id, month_year), reported_at);
```

BUCKETING

- Sensor ID: UUID
- **MonthYear**: Integer or String
- Timestamp: Timestamp
- Value: float

01

Intro to Bootcamp 2022
HouseKeeping

02

Week I Refresh
Apache Cassandra in 2 words

03

Keys, Keys, Keys
How do you define keys

04

Partitioning in details
Rules of Partitioning

05

Modelling Methodology
Building efficient Data Model

06

What's next?
Quiz, Homework, Next week

Agenda

"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

Normalization

"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

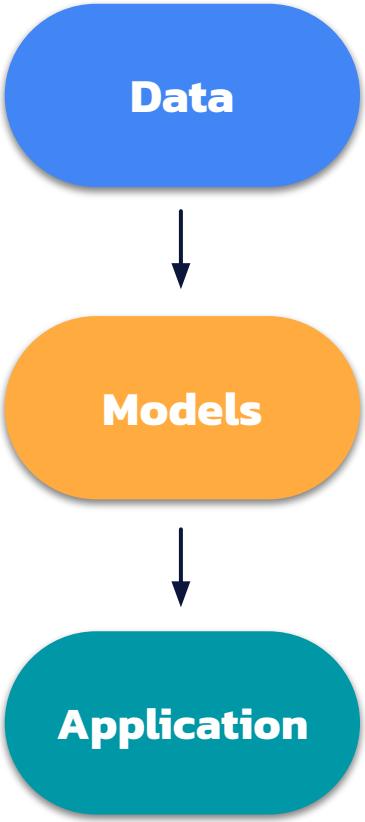
departmentId	department
1	Engineering
2	Math

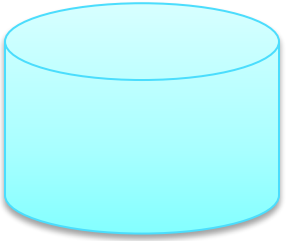
1. Analyze raw data

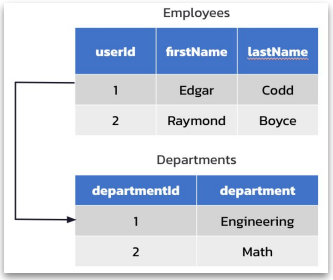
2. Identify entities, their properties and relations

3. Design tables, using **normalization** and foreign keys.

4. Use JOIN when doing queries to join normalized data from multiple tables

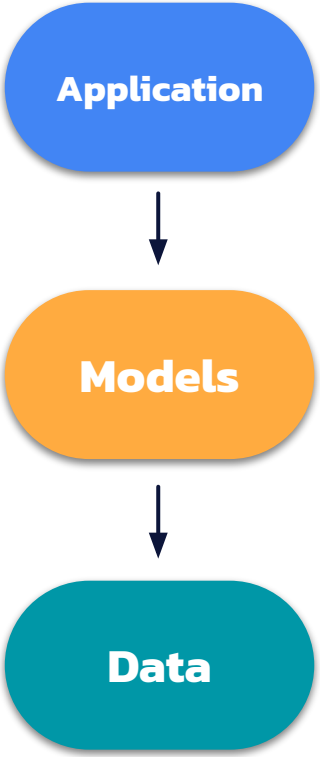






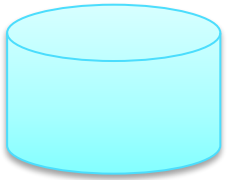
Relational Data Modelling

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. Use BATCH when inserting or updating denormalized data of multiple tables



Employees

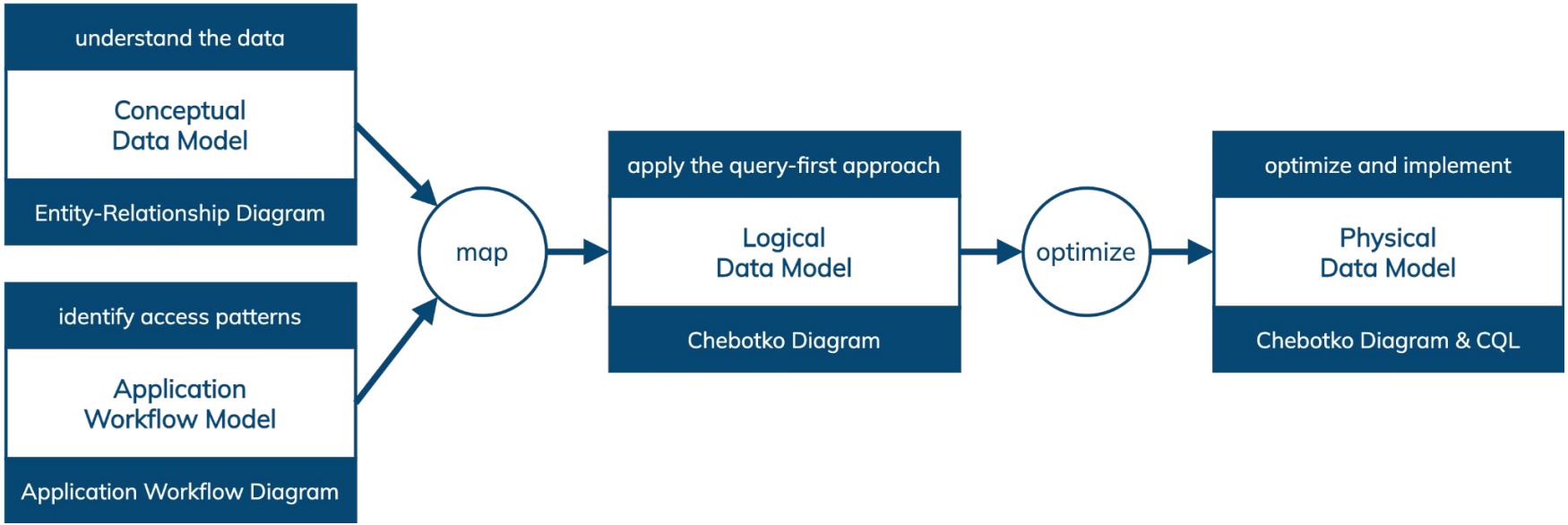
userId	firstName	lastName	department
1	Edgar	Codd	Engineering
2	Raymond	Boyce	Math
3	Sage	Lahja	Math
4	Juniper	Jones	Botany



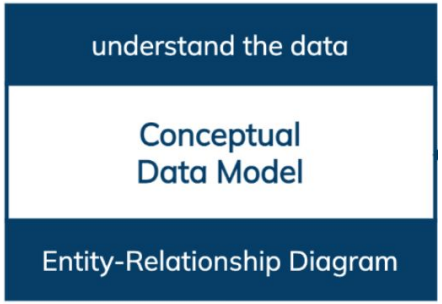
Data Modelling Methodology

by Dr. Artem Chebotko

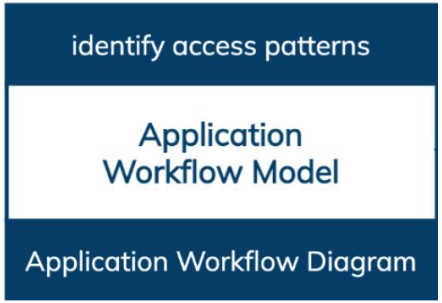
Modelling process
Step by Step



Data Modelling Methodology

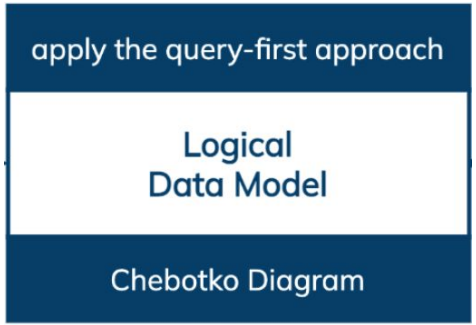


Analyse the Domain



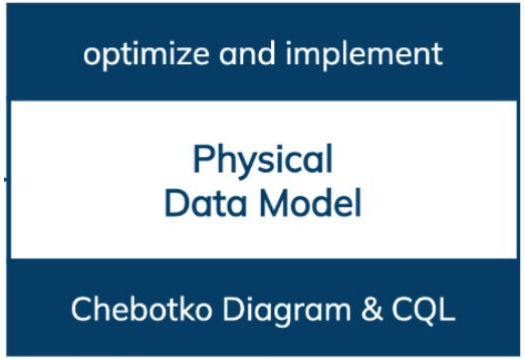
Analyse Customer Workflows

Data Modelling Methodology: Step I



Design queries and build
tables based on the queries

Data Modelling Methodology: Step II

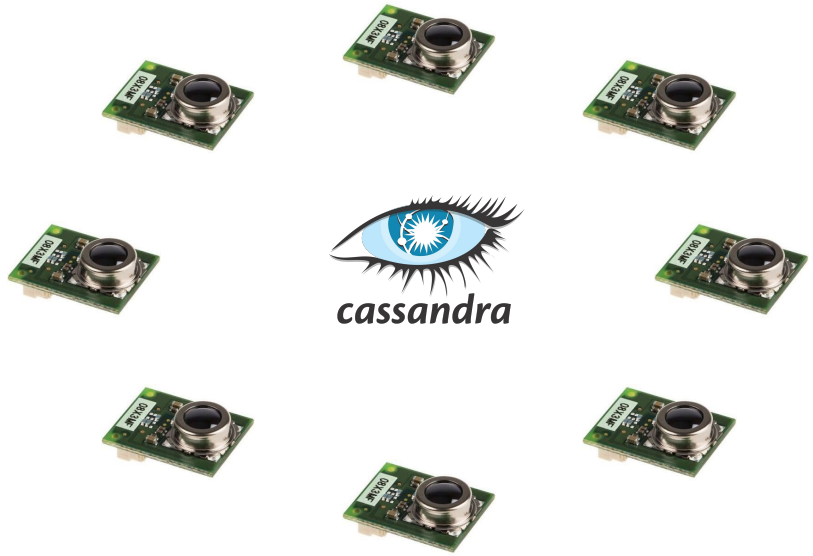


Optimize and Create

Data Modelling Methodology: Step III

Learn by a real use-case! In this one, we learn how to create an efficient data model for global temperature monitoring sensor networks.

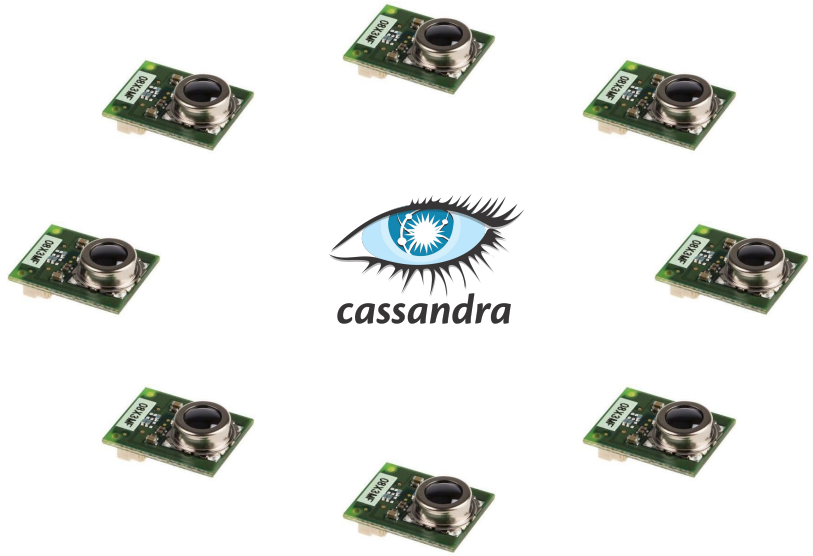
We cover a conceptual data model, application workflow, logical data model, physical data model, final CQL schema and query design.

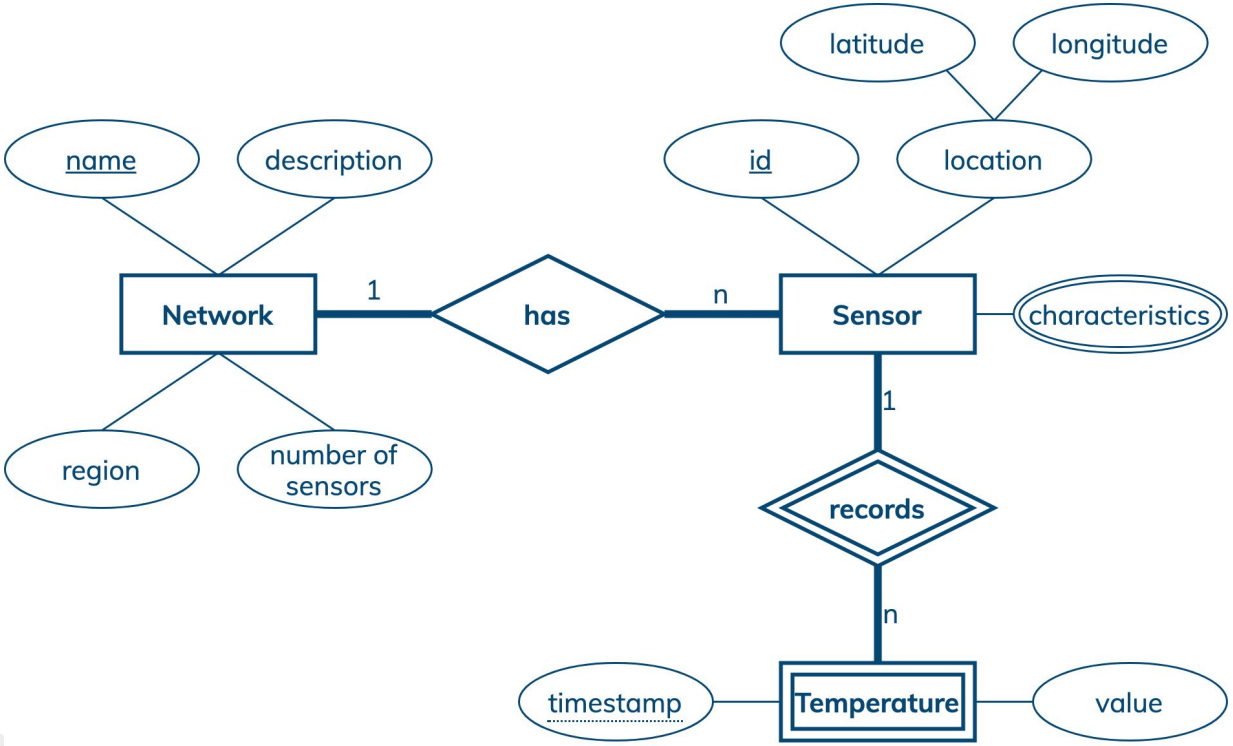


IoT Sensor Scenario features **sensor networks**, **sensors**, and **temperature measurements**.

Each **network** has a unique name, description, region, and number of sensors. A **sensor** is described by a unique id, location (latitude and longitude), multiple sensor characteristics. A temperature **measurement** has a timestamp and value, and belongs to one sensor id.

While a network can have many sensors, each sensor can only belong to one network.





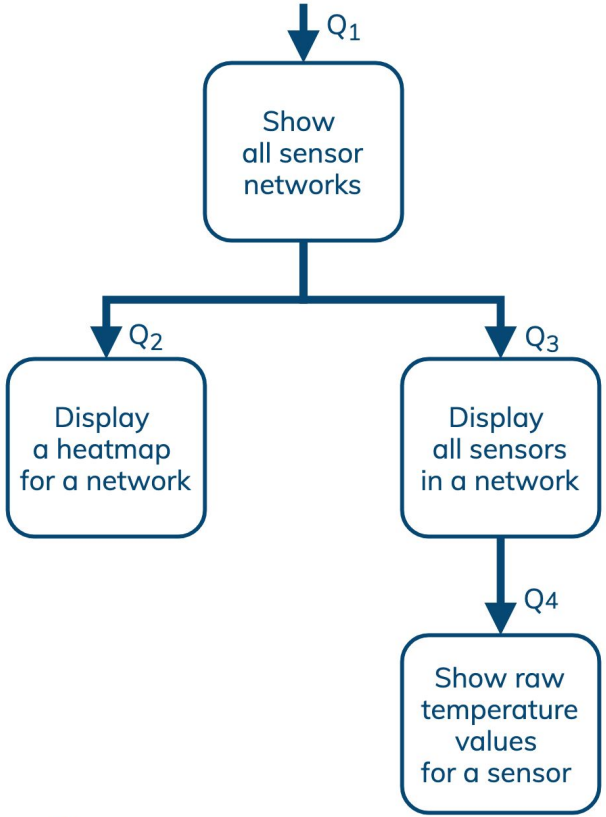
Entity-Relationship Diagram

The application workflow has an entry-point task that **shows all sensor networks**. This task requires querying a database to find information about all networks and arrange the results in ascending order of network names.

Next, an application can either display a **heatmap** for a selected network, or **display all sensors in a selected network**.

Finally, the latter can lead to the task of **showing raw temperature values** for a given sensor.

Q_1: Find information about all networks; order by name (asc)
Q_2: Find hourly average temperatures for every sensor in a specified network for a given date range; order by date (desc) and hour (desc)
Q_3: Find information about all sensors in a specified network
Q_4: Find raw measurements for a particular sensor on a specified date; order by timestamp (desc)



Data access patterns

Q_1: Find information about all networks; order by name (asc)
Q_2: Find hourly average temperatures for every sensor in a specified network for a given date range; order by date (desc) and hour (desc)
Q_3: Find information about all sensors in a specified network
Q_4: Find raw measurements for a particular sensor on a specified date; order by timestamp (desc)

Application Workflow Diagram

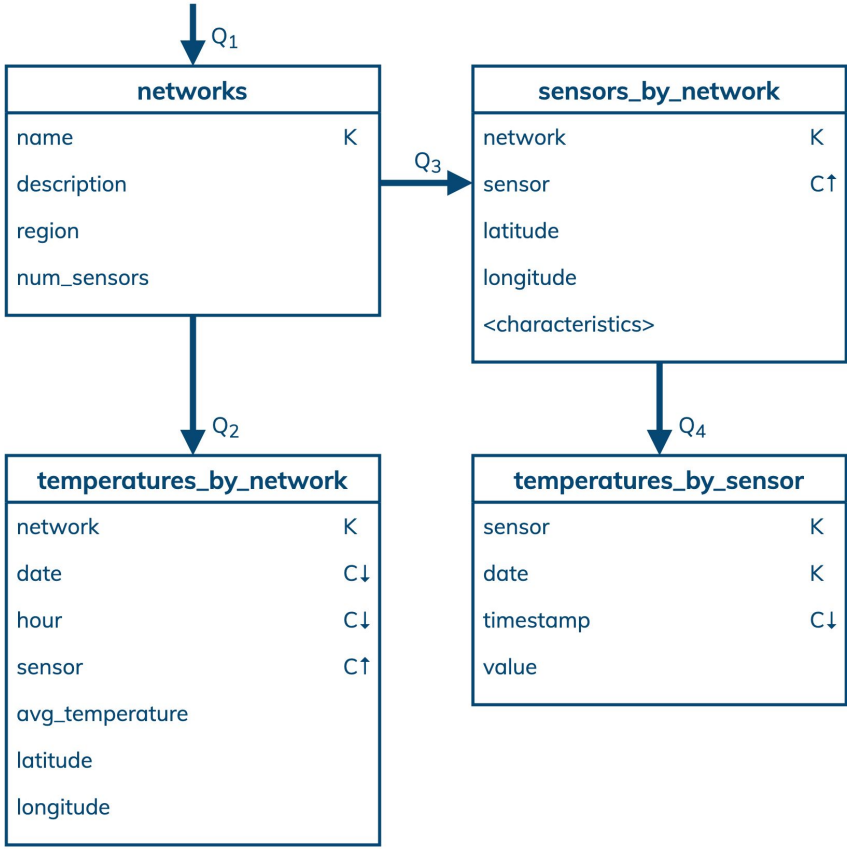
```
SELECT name, description,
       region, num_sensors
  FROM networks
 WHERE bucket = 'all';
```

Q1

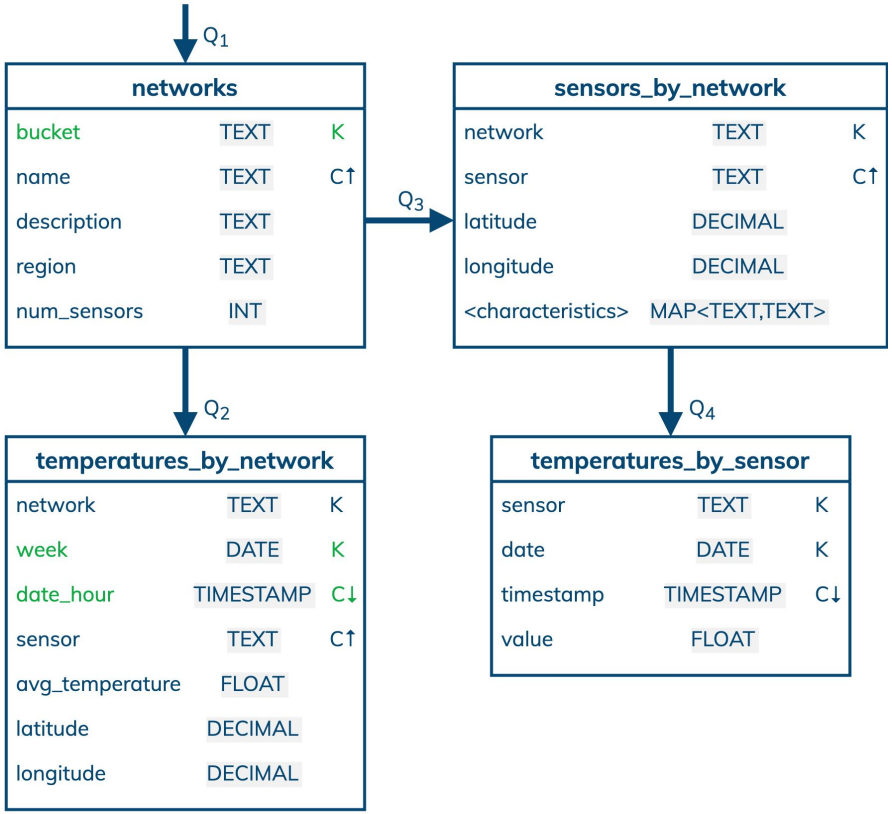
```
SELECT date_hour, avg_temperature,
            latitude, longitude, sensor
FROM temperatures_by_network
WHERE network    = 'forest-net'
    AND week       = '2020-07-05'
    AND date_hour >= '2020-07-05'
    AND date_hour  < '2020-07-07';
```

Q2

Queries First!



Logical Data Model: Chebotko Diagram



Physical Data Model: Chebotko Diagram

```
CREATE TABLE networks (
    bucket TEXT,
    name TEXT,
    description TEXT,
    region TEXT,
    num_sensors INT,
    PRIMARY KEY ((bucket),name)
);
```

DDL Statements: Networks

```
CREATE TABLE sensors_by_network (
    network TEXT,
    sensor TEXT,
    latitude DECIMAL,
    longitude DECIMAL,
    characteristics MAP<TEXT,TEXT>,
    PRIMARY KEY ((network),sensor)
);
```

DDL Statements: Sensors

```
CREATE TABLE temperatures_by_sensor (
    sensor TEXT,
    date DATE,
    timestamp TIMESTAMP,
    value FLOAT,
    PRIMARY KEY ((sensor,date), timestamp)
) WITH CLUSTERING ORDER BY (timestamp DESC);
```


DDL Statements: Networks

QUESTION

Imagine we design together a data model for a YouTube!

Build a Netflix clone with GraphQL, React and a NoSQL DB

42,071 views • Dec 8, 2021 • Learn how to manage big data in a real-time recommenda Show more

153

DataStax Developers
24.5K subscribers

SUBSCRIBED

Comments
12

I got this v

12 Comments

SORT BY

Add a public comment...

ffff 3 weeks ago

I got this video as an ad! It was still quite informative and not annoying like other ads 👍

14 REPLY

David Arcos 2 weeks ago

Very cool! I am glad I followed the ad link and learned more about GraphQL I always wanted to take a look at it. Thank you.

2 REPLY

Alejo Franco 3 weeks ago

I lov u man please keep doing it, you are really helping me to improve my coders skills, so thank you very much.

REPLY

YouTube Comments Case

Use-Case I User opens a Video Page

Workflow I: Find comments related to target video using its identifier, most recent first

Use-Case II User opens his own Profile
Use-Case III Moderator opens a User Profile

Workflow I: Find comments related to target user using its identifier, get most recent first

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

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

Question: Do we have to apply denormalisation? How many tables we will need to serve those requirements?

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

comments_by_user

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

comments_by_video

01

Intro to Bootcamp 2022
HouseKeeping

02

Week I Refresh
Apache Cassandra in 2 words

03

Keys, Keys, Keys
How do you define keys

04

Partitioning in details
Rules of Partitioning

05

Modelling Methodology
Building efficient Data Model

06

What's next?
Quiz, Homework, Next week

Agenda

menti.com


Apache
CASSANDRA
25
DS COMICS
DATASTAX

Go to www.menti.com and use the code 3491 9972
Inequality predicates are allowed on ...
4
3
15
1
All table columns
Partition key columns
clustering key columns
No inequality predicates are allowed
2:10:19 / 2:26:05

Go to www.menti.com and use the code 3491 9972
Leaderboard
4821 p spanda
4820 p Agent X9
fastest
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

SWAG WINNERS 🎉

Congratulations to 1st, 2nd and 3rd place on the Menti quiz!

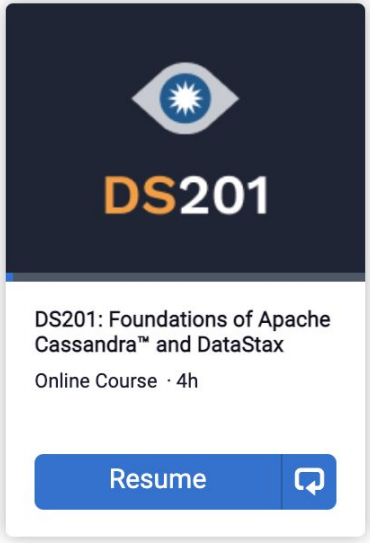
To claim your prize, please send an email to:

gary.harvey@datastax.com

** Include a screenshot of your Menti screen

Homework

!homework



JAN
19

Bootcamp - Build a Full-Stack Backend with a NoSQL database

by DataStax

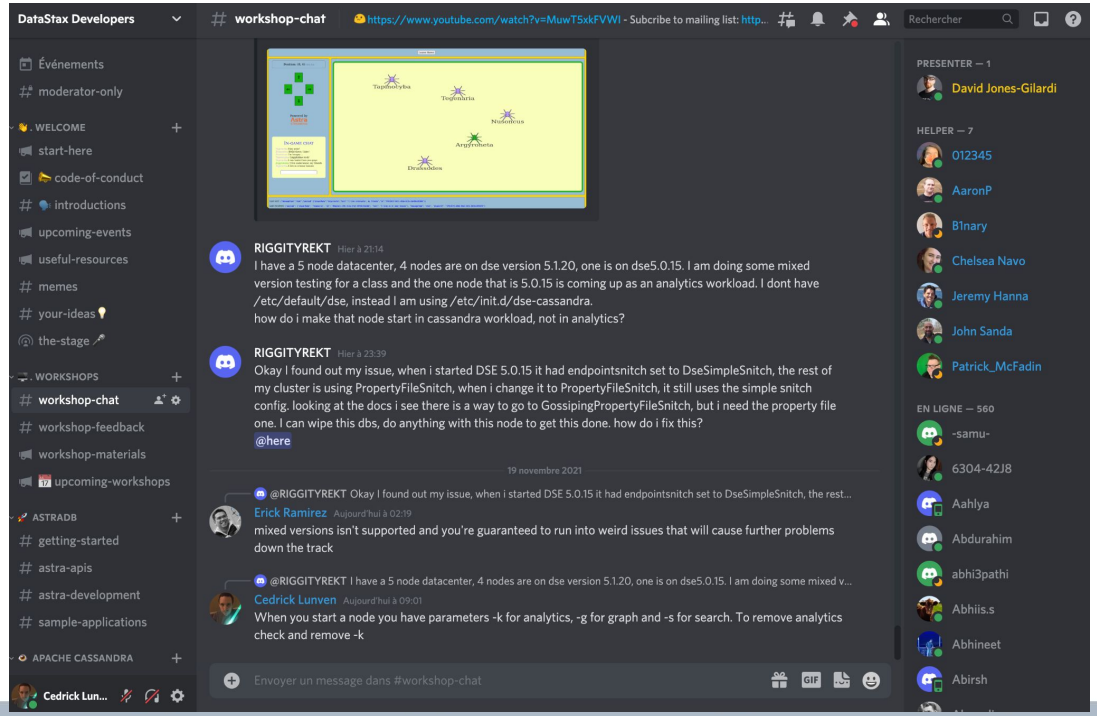
46597 followers Follow

Free

Register

Next Week

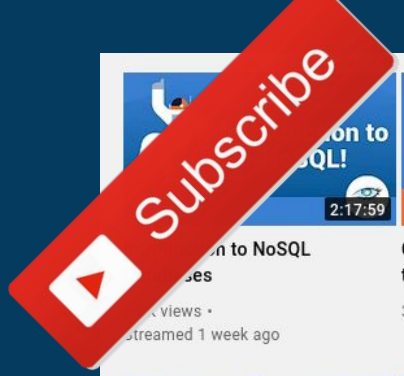
Join our 18k Discord Community
DataStax Developers

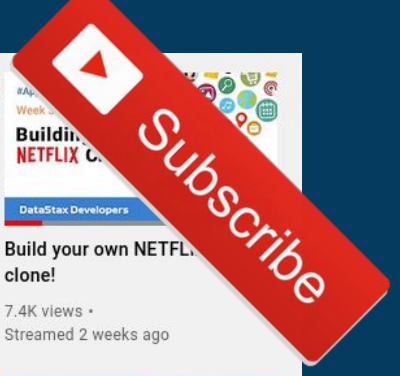


!discord

dtsx.io/discord

Discord Community





…n to NoSQL …ses

…k views •
…Streamed 1 week ago

Crash Course | Introduction to Cassandra for Developers

331 views • 1 week ago

Introduction to NoSQL Databases

3.5K views •
Streamed 1 week ago

Introduction to NoSQL Databases

10K views •
Streamed 1 week ago

Build your own NETFLIX clone!

4K views •
Streamed 2 weeks ago

Build your own NETFL… clone!

7.4K views •
Streamed 2 weeks ago

Astra Streaming Demo

177 views • 2 weeks ago

Kubernetes Ingress Management with Traefik…

496 views •
Streamed 2 weeks ago

Build your own TikTok clone!

1.9K views •
Streamed 3 weeks ago

Build your own TikTok Clone!

4K views •
Streamed 3 weeks ago

How to use the Connect Driver in Astra DB

113 views • 4 weeks ago

How to use the CQL Console in Astra DB

39 views • 4 weeks ago

How to create an Authentication Token in…

37 views • 4 weeks ago

How to use the Data Loader in Astra DB

62 views • 4 weeks ago

Astra DB Sample App Gallery

36 views • 4 weeks ago

How to use Secure Connect in Astra DB

42 views • 4 weeks ago

Cassandra Day India: CL Room (Workshops)

2.4K views •
Streamed 4 weeks ago

Cassandra Day India: RF Room (Talks)

1.3K views •
Streamed 1 month ago

Thank You!