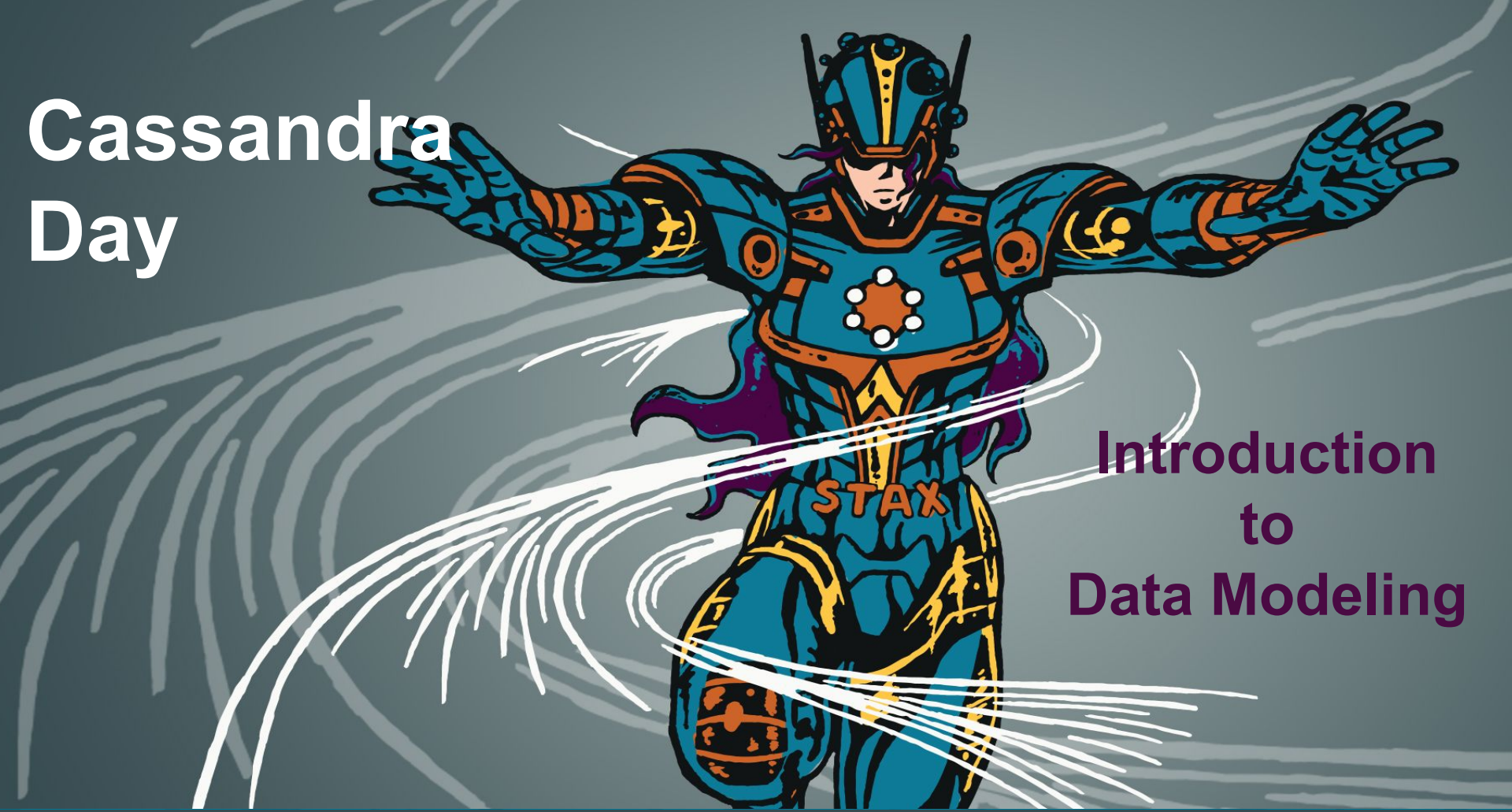
Cassandra Day



Introduction to Data Modeling

DATASTAX

DATASTAX ACADEMY

SQL (relational) vs CQL (NoSQL Cassandra)

Structuring Your Database

Normalization: To reduce data redundancy and increase data integrity.

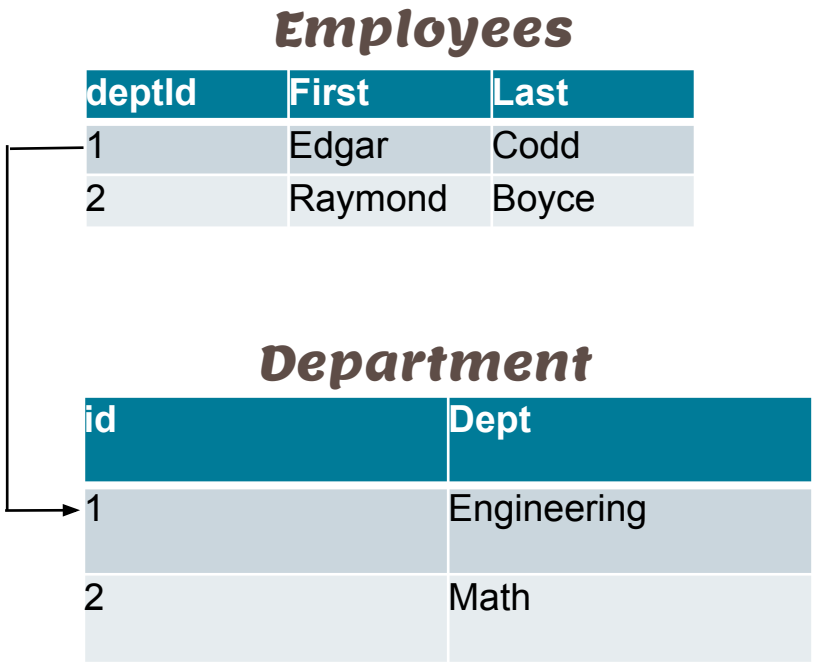
Denormalization: Must be done in read heavy workloads to increase performance

Normalization

The process of **structuring** a relational database in accordance with a series of **normal forms** in order **to reduce data redundancy** and **increase data integrity.**

Relational Data Models

- Multiple normal forms
  - most do not go beyond 3NF
- Foreign Keys
- Joins



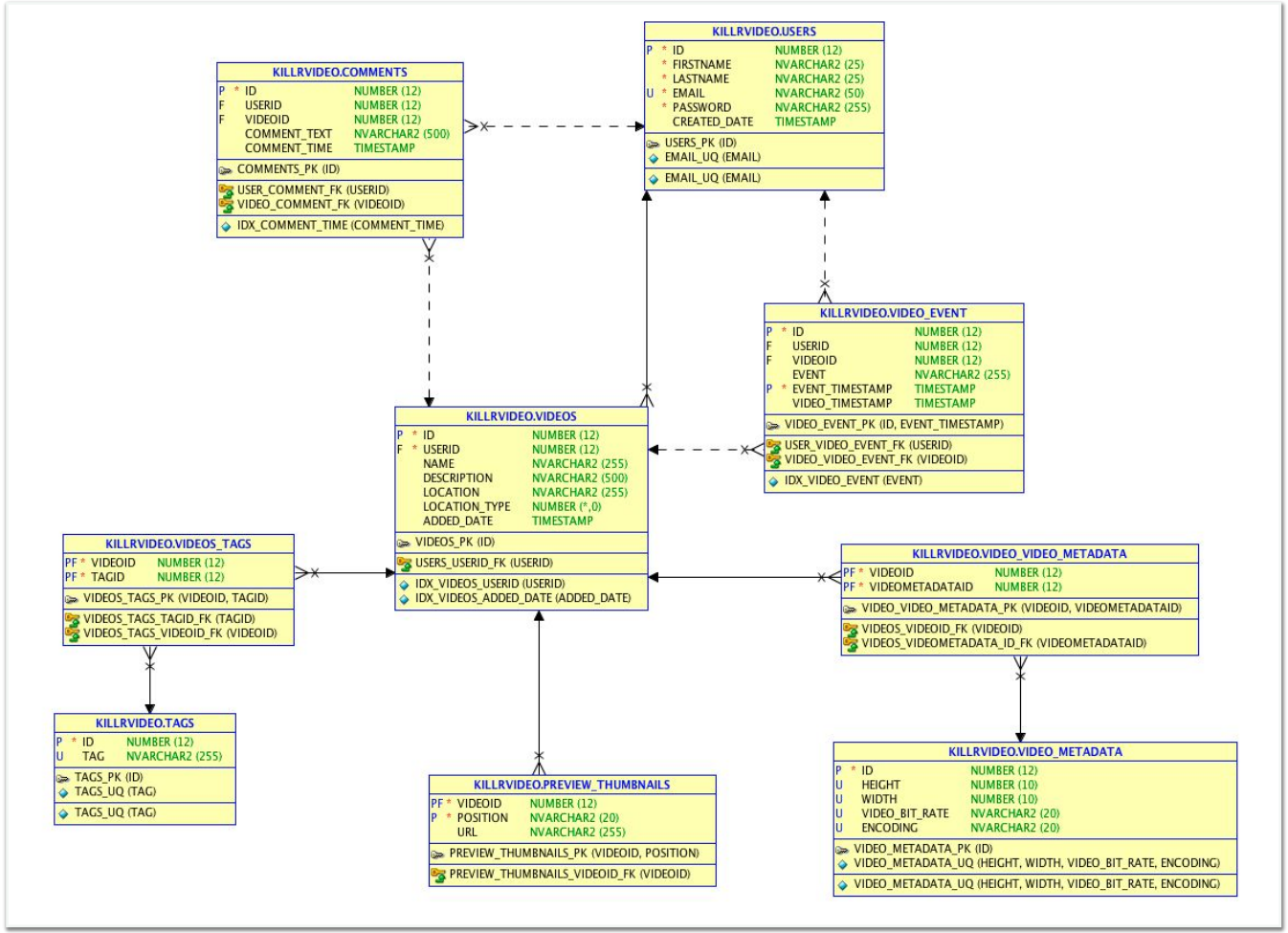
Relational Modeling

- Create entity table
- Add constraints
- Index fields
- Foreign Key relationships

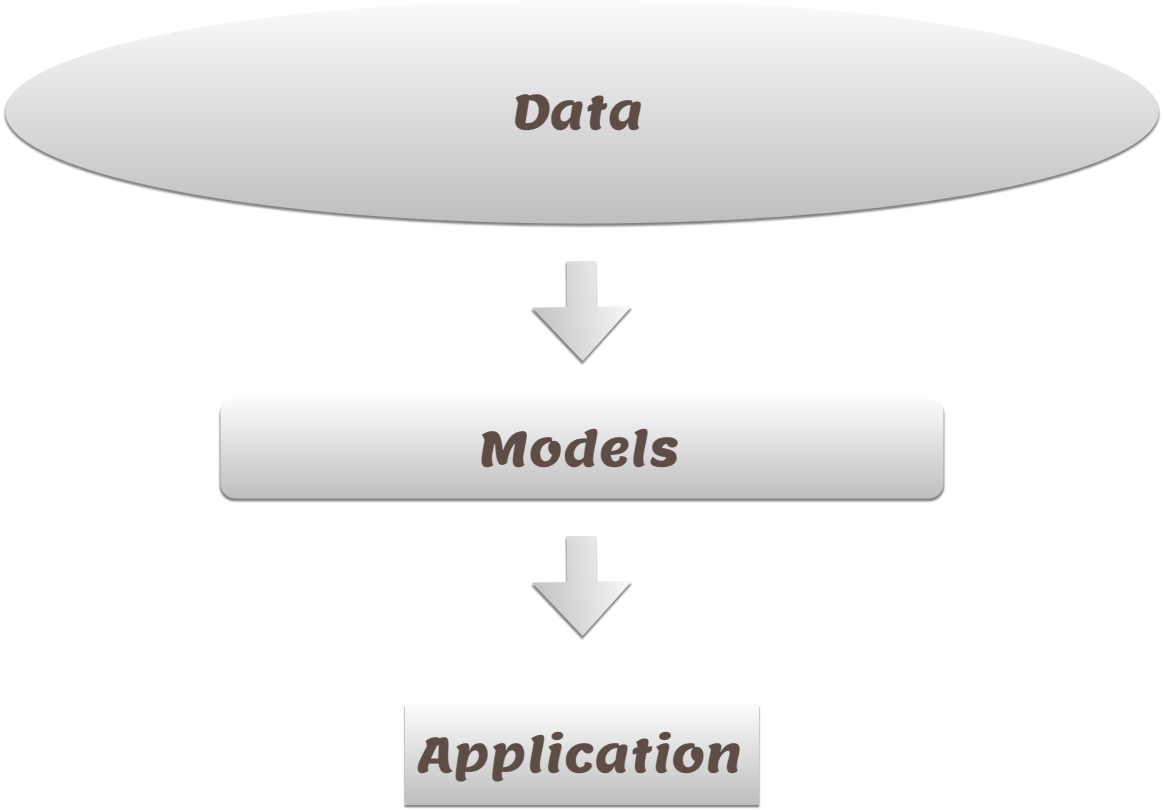
```
CREATE TABLE users (
  id          number(12) NOT NULL ,
  firstname    nvarchar2(25) NOT NULL ,
  lastname     nvarchar2(25) NOT NULL,
  email       nvarchar2(50) NOT NULL,
  password     nvarchar2(255) NOT NULL,
  created_date timestamp(6),
  PRIMARY KEY (id),
  CONSTRAINT email_uq UNIQUE (email)
);

-- Users by email address index
CREATE INDEX idx_users_email ON users (email);
```

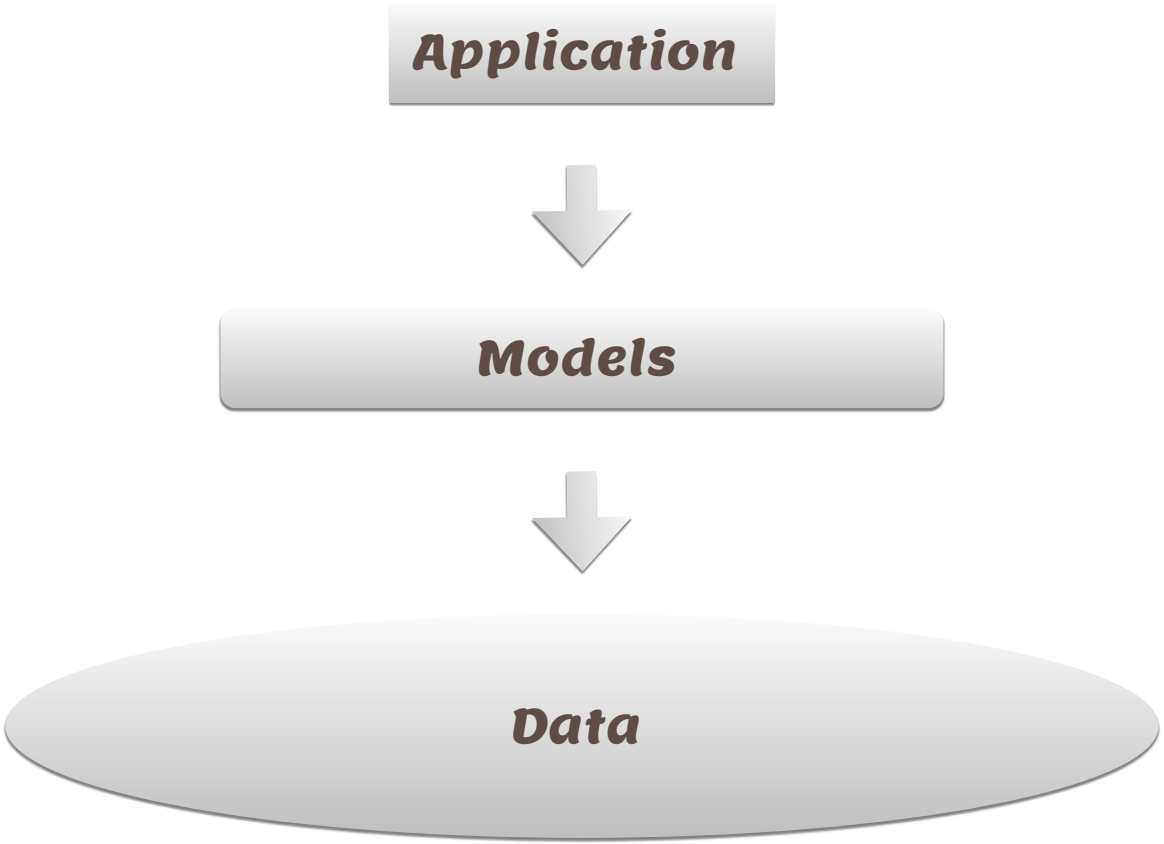
```
CREATE TABLE videos (
  id number(12),
  userid number(12) NOT NULL,
  name nvarchar2(255),
  description nvarchar2(500),
  location nvarchar2(255),
  location_type int,
  added_date timestamp,
  CONSTRAINT users_userid_fk
    FOREIGN KEY (userid)
    REFERENCES users (Id) ON DELETE CASCADE,
  PRIMARY KEY (id)
);
```



Relational Modeling



Cassandra Modeling



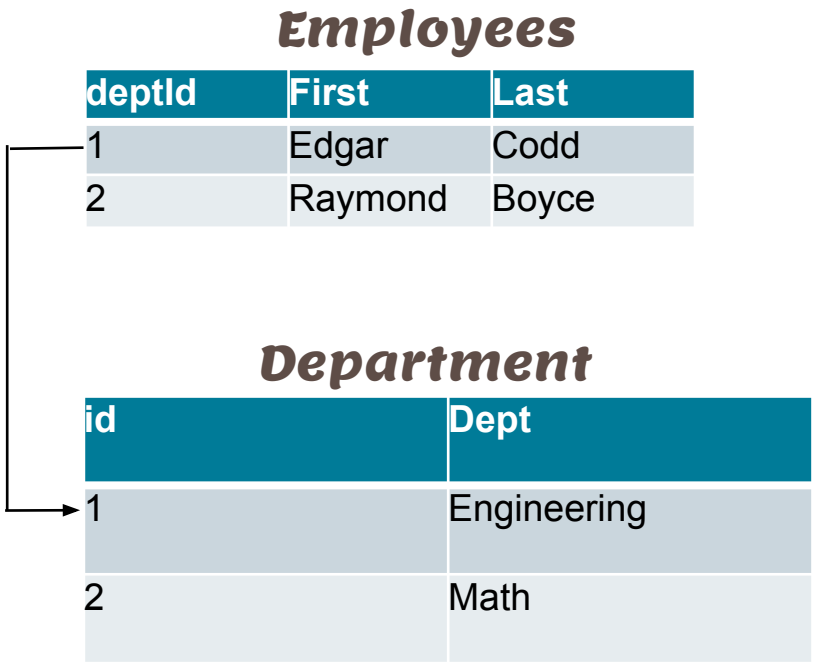
Denormalization

“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.”

CQL vs SQL

- No joins
- Limited aggregations

```
SELECT e.First, e.Last, d.Dept
FROM Department d, Employees e
WHERE 'Codd' = e.Last
AND e.deptId = d.id
```



Denormalization

- Combine table columns into a single view
- Eliminate the need for joins
- Queries are concise and easy to understand

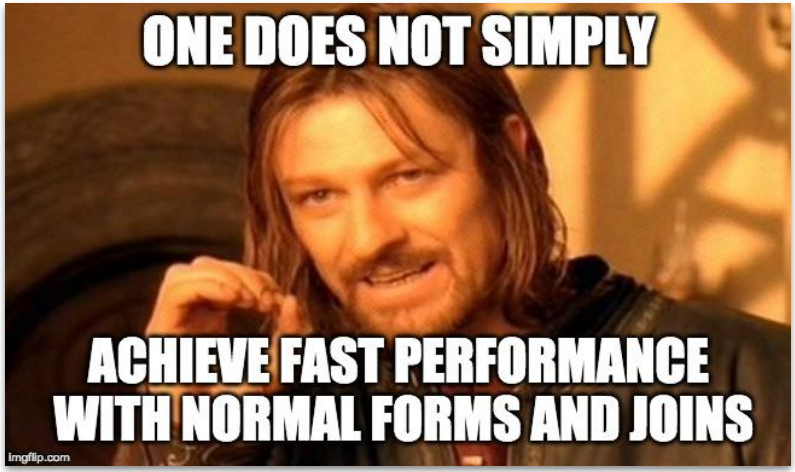
Employees

id	First	Last	Dept
1	Edgar	Codd	Engineering
2	Raymond	Boyce	Math

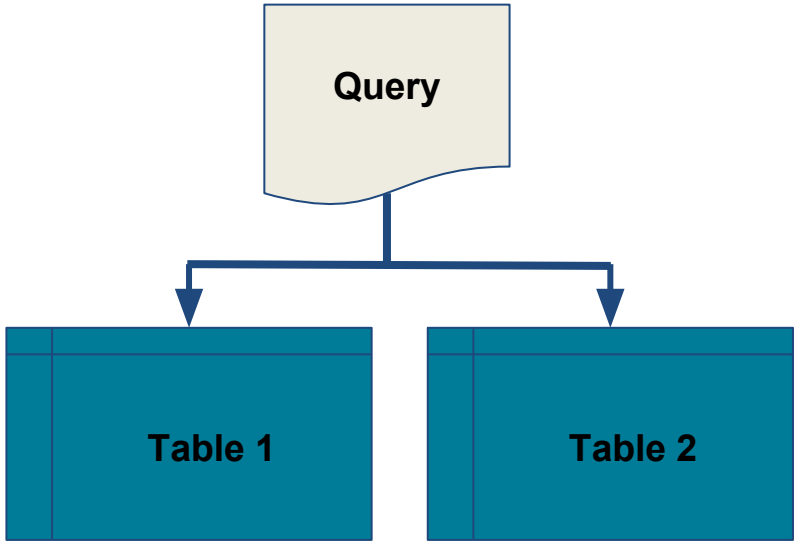
```
SELECT First, Last, Dept
FROM employees
WHERE id = '1'
```

Denormalization in Apache Cassandra

Denormalization of tables in Apache Cassandra is **absolutely critical**. The biggest take away is to think about your **queries** first. There are no `JOINS` in Apache Cassandra.

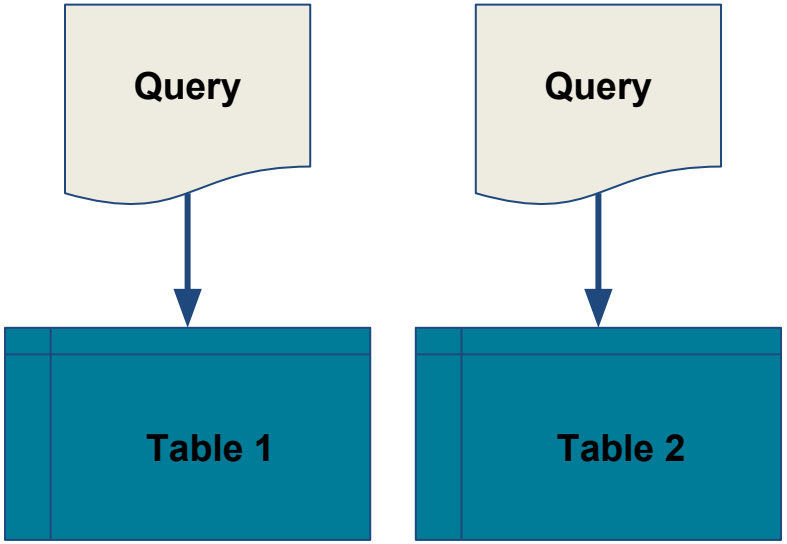


Relational Databases



In a relational database, one query can access and join data from multiple tables

NoSQL Databases

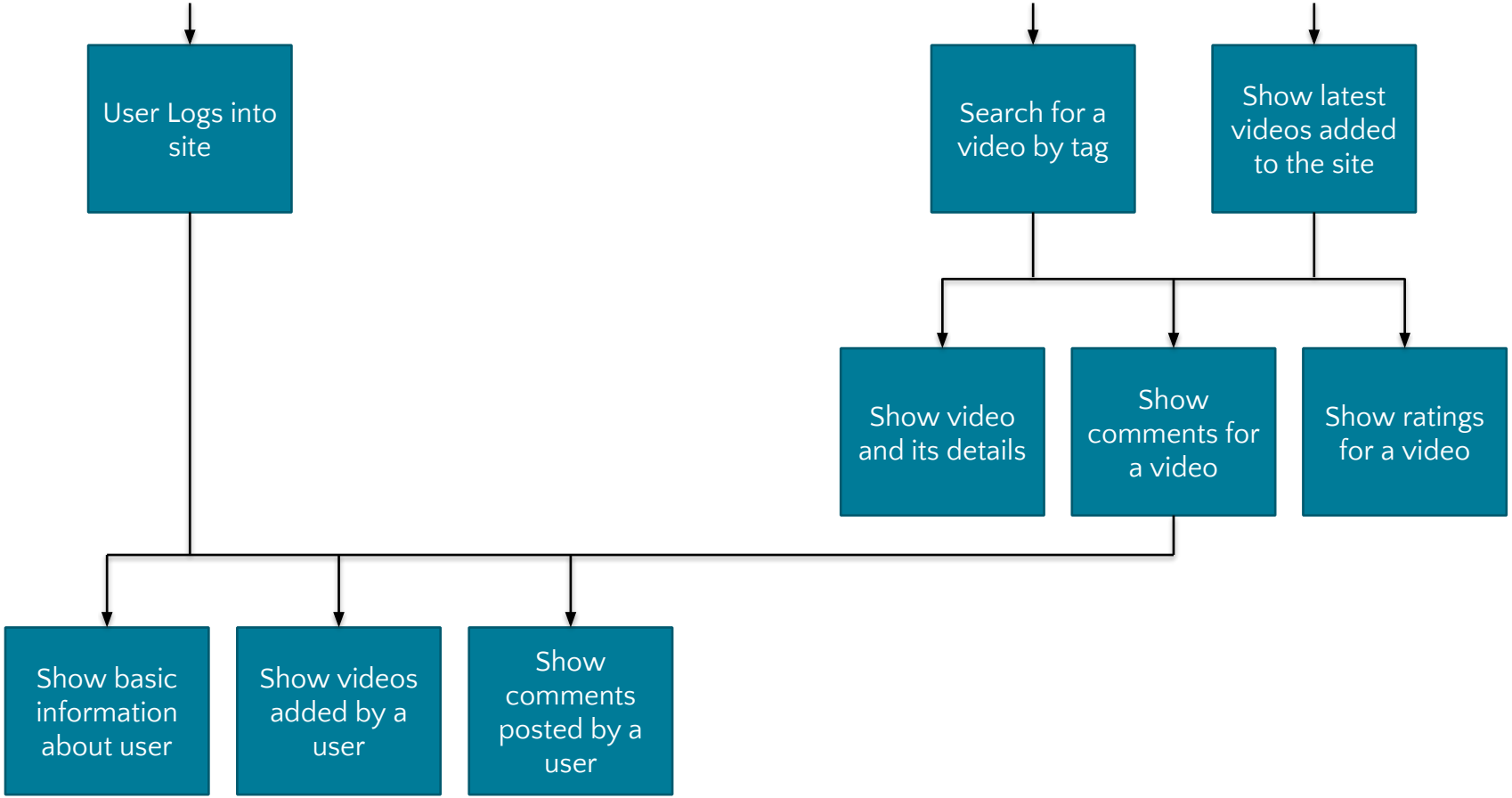


In Apache Cassandra, you cannot join data, queries can only access data from one table

Modeling Queries

- What are your application's workflows?
- Knowing your queries in advance is **CRITICAL**
- Different from RDBMS because I can't just JOIN or create a new indexes to support new queries
- **One** table per **one** query

Some Application Workflows in KillrVideo



Some Queries in KillrVideo to Support Workflows

Users

User Logs into site

Find user by email address

Show basic information about user

Find user by id

Comments

Show comments for a video

Find comments by video (latest first)

Show comments posted by a user

Find comments by user (latest first)

Ratings

Show ratings for a video

Find ratings by video

DATASTAX ACADEMY

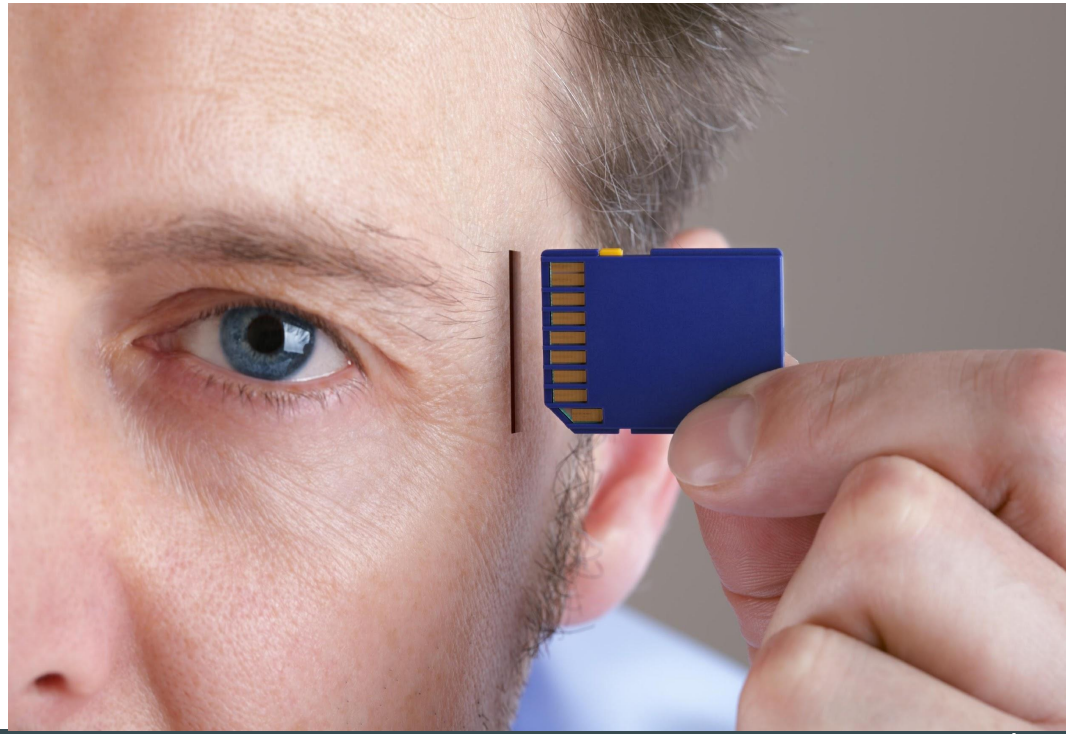
Cassandra data modeling

DATASTAX ACADEMY

The Cassandra Mind-shift

Change the way you think about data modeling

- Two Main concepts for successful modeling:
  - Begin with the query in mind
  - Denormalization is good



Cassandra Data Modeling

Partition – the fundamental unit of access

- A Partition has
  - A partition key
  - Associated rows with
    - Clustering columns
    - Data columns

Last Name	First Name	Address	Email
Flintstone	Dino	3 Stone St	dino@gmail.com
Flintstone	Fred	3 Stone St	fred@gmail.com
Flintstone	Wilma	3 Stone St	wilm@gmail.com
Rubble	Barney	4 Rock Cir	brub@gmail.com
Rubble	Betty	4 Rock Cir	betr@gmail.com

Partition Key:
"Bedrock"

Hash Function

42

Cassandra Data Modeling

Tables hold many partitions

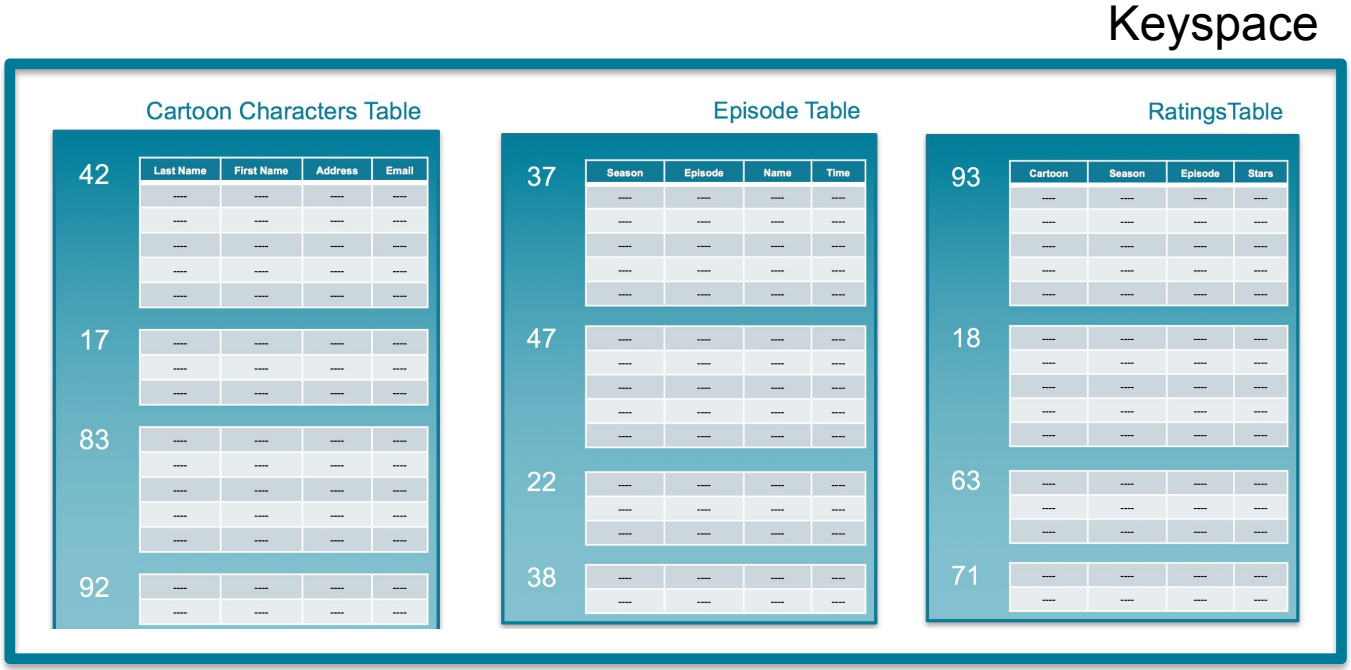
"Fro[illegible]" Partition

Last Name	First Name	Address	Email
Moose	Bullwinkle	1 Tree St	moo@gmail.com
Squirrel	Rocky	3 Sky Blvd	fly@gmail.com
Jetson	Judy	2 StarSt	judy@gmail.com
Rubble	Barney	4 Rock Cir	brub@gmail.com
Rubble	Betty	4 Rock Cir	betr@gmail.com

Cartoon Characters by City Table

Cassandra Data Modeling

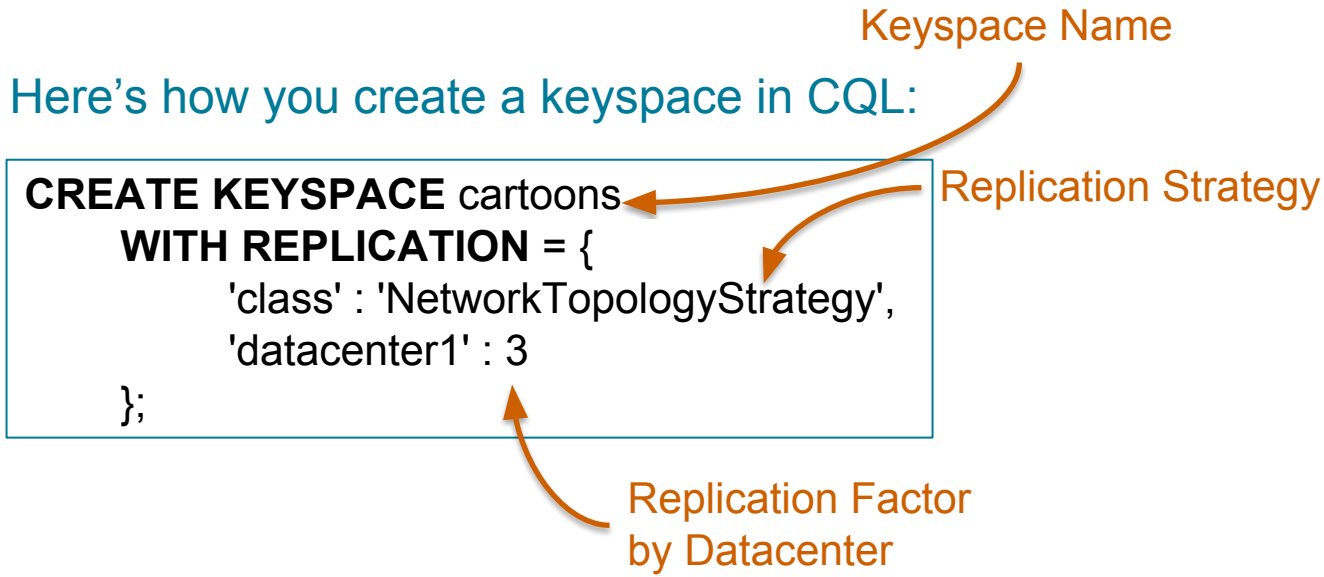
Keyspaces contain many tables



Cassandra Data Modeling

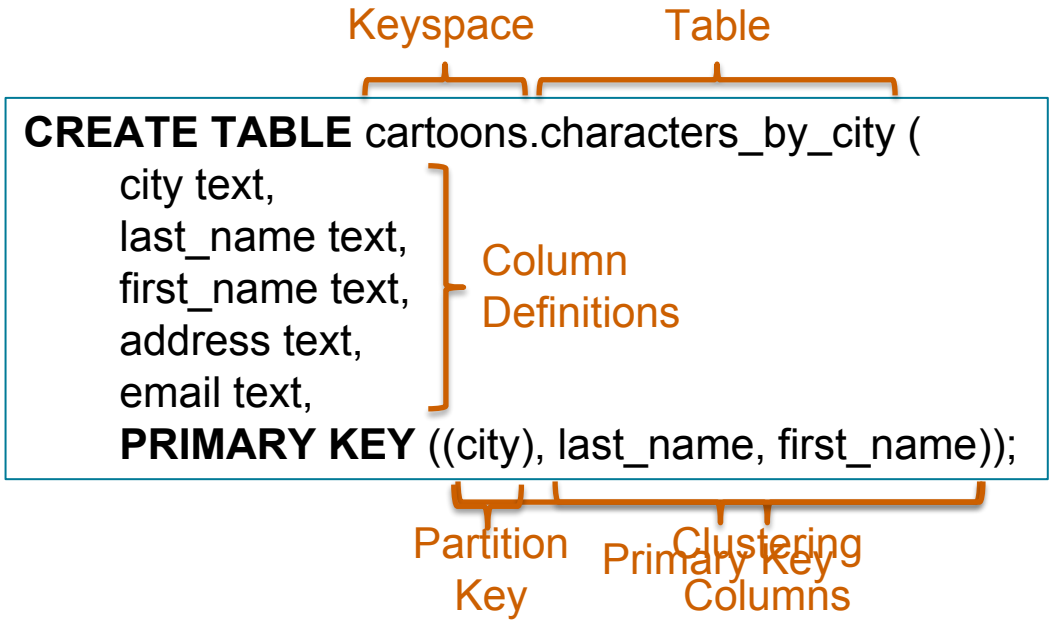
CQL – Cassandra's Query Language

Here's how you create a keyspace in CQL:



Cassandra Data Modeling

CQL – Create table example



Cassandra Data Modeling

CQLSH – Investigating our schema

To list your keyspaces:

```
DESCRIBE KEYSPACES;
```

To look at a specific keyspace definition:

```
DESCRIBE KEYSPACE cartoon;
```

To look at a table definition:

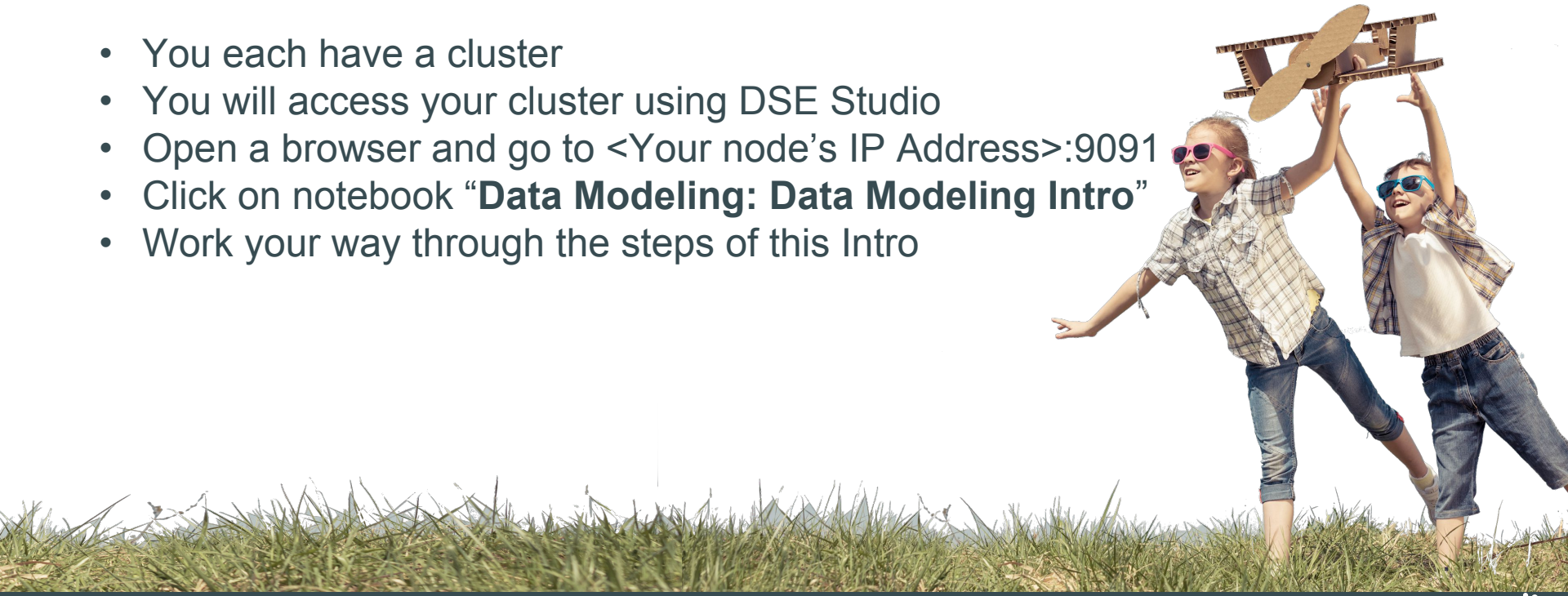
```
DESCRIBE TABLE cartoon.characters_by_city;
```

DATASTAX ACADEMY

Enough Talk Already!!!!

It's time to play!

- You each have a cluster
- You will access your cluster using DSE Studio
- Open a browser and go to <Your node's IP Address>:9091
- Click on notebook "**Data Modeling: Data Modeling Intro**"
- Work your way through the steps of this Intro



DATASTAX ACADEMY

Data Modeling Intro

Quick review

- Key concepts:
  - Keyspace – contains tables
  - Table – contains partitions
  - Row – has a primary key and data columns
  - Partition – basic unit of storage/retrieval
    - Identified by partition key embedded within primary key
    - Contains one or more rows
  - Primary key – intra-table row identifier
    - Consists of partition key and clustering columns
  - Partition key – partition identifier, hashes to partition token
  - Clustering column – intra-partition key for sorting rows within partition

DATASTAX ACADEMY

Let’s do a Data Modeling Project Together!

Welcome to Cassandra-Land

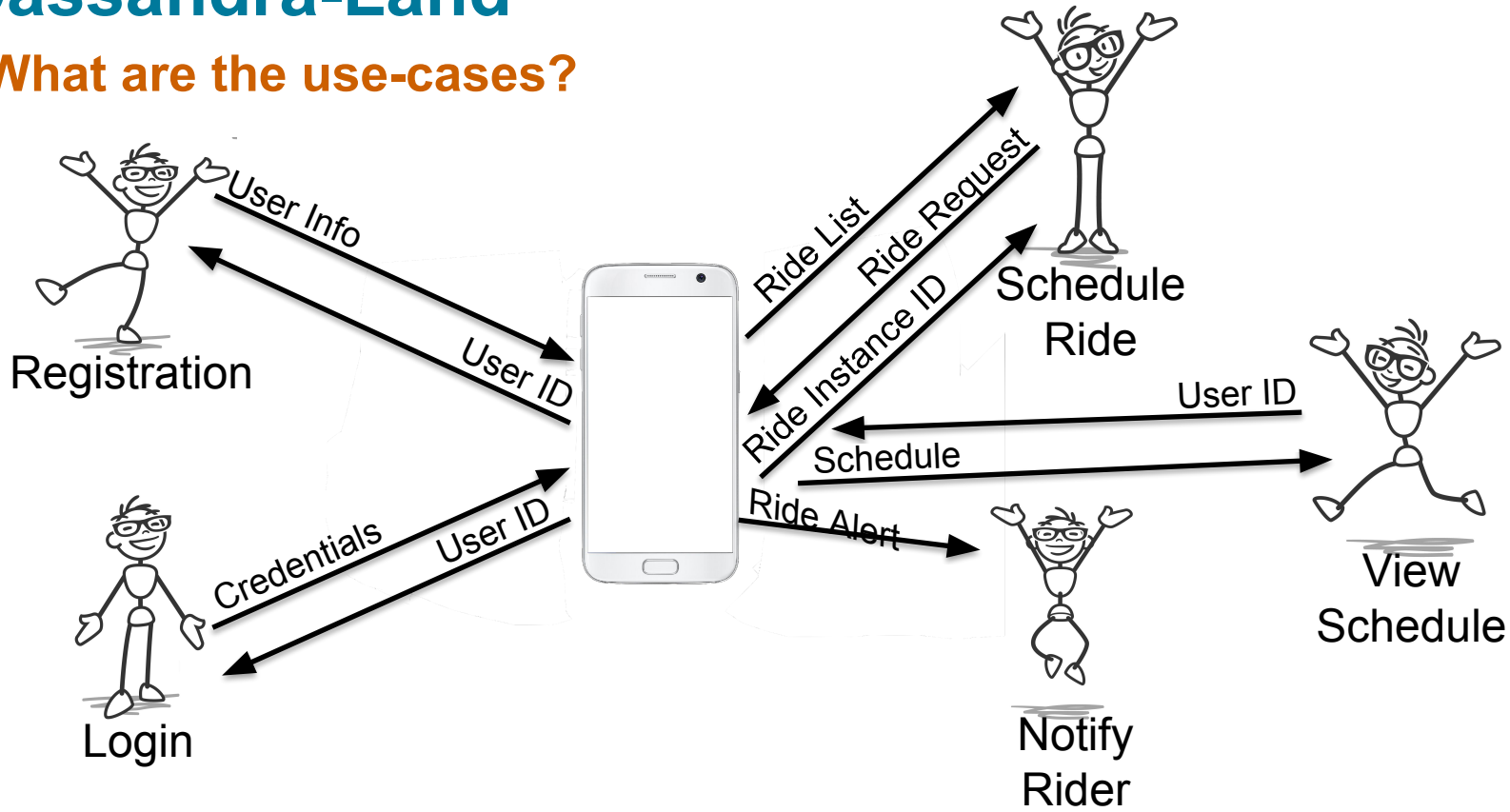
The Theme Park Where You Can Find…

- **Distributed & Fault-Tolerant Rides**
- **Amazing Throughput**
- **And Fast Response Times**

But We Need an App!

Cassandra-Land

What are the use-cases?



Registration Use-Case

What you need to know – CREATE KEYSPACE

```
CREATE KEYSPACE <keyspace name> WITH REPLICATION = {
       'class' : <replication strategy>,
      <datacenter name> : <replication factor>,
      // Specify addition datacenters/replication factors here...
};
```

For example:

```
CREATE KEYSPACE park WITH REPLICATION = {
       'class' : NetworkTopologyStrategy,
      'USWestDC': 3,
      'USEastDC': 2
};
```

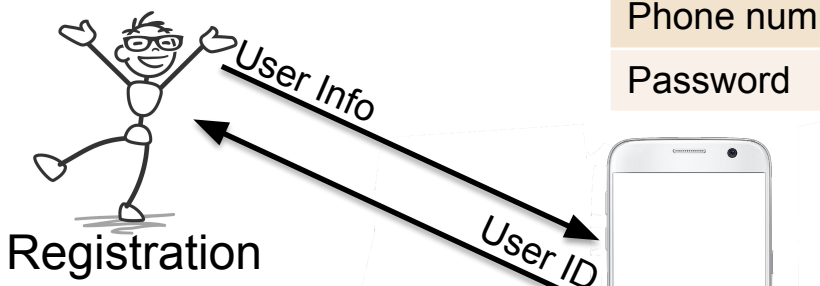
Registration Use-Case

What you need to know – DESCRIBE KEYSPACE

DESCRIBE KEYSPACE <keyspace name>;

Cassandra-Land

Registration Use-Case



User Info

Field	Type
Phone number	text
Password	text

User ID

Field	Type
User ID	UUID

users_by_phone_number

Field	Type
phone_number	text
user_id	UUID
password	text

Registration Use-Case

What you need to know – CREATE TABLE

```
CREATE TABLE <keyspace name>.<table name> (
        <field name> <field type>,
        // Add additional field descriptions here
        PRIMARY KEY(<primary key descriptor>)
);
```

Registration Use-Case

What you need to know – INSERT

```
INSERT INTO <keyspace name>.<table name>
      (<column list>)
      VALUES(<column values>);
```

Registration Use-Case

What you need to know – SELECT *

SELECT * FROM <keyspace name>.<table name>;

DATASTAX ACADEMY

Registration Use-Case

What you need to know – Upserts

- Cassandra does **NOT** read before writing
- Inserting a row with the same primary key causes an update called an "upsert"
- Similarly, updates to non-existent rows cause an insert
- INSERT INTO keyspace.table IF NOT EXISTS …
  - Lightweight transaction to prevent upserts as they do read before writing

DATASTAX ACADEMY

Notebook Data Modeling: Cassandra-Land Project

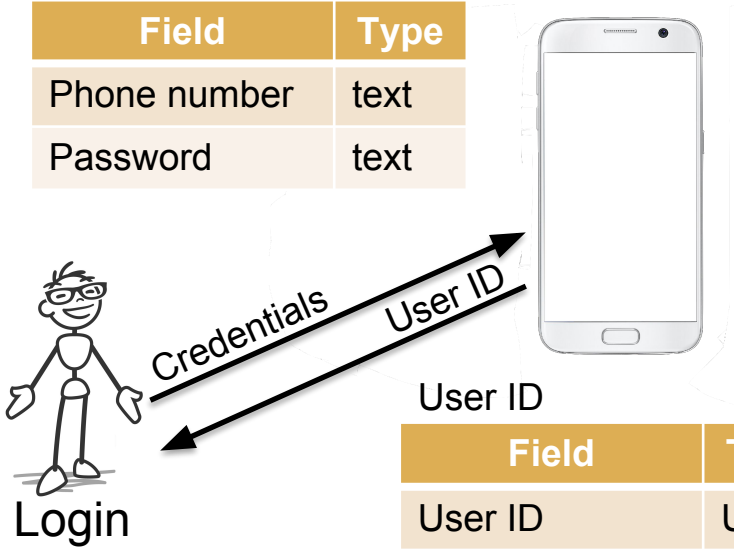
Do Part 1

Cassandra-Land

Login Use-Case

Credentials

Field	Type
Phone number	text
Password	text



User ID

Field	Type
User ID	UUID

users_by_phone_number

Field	Type
phone_number	text
user_id	UUID
password	text

Registration Use-Case

What you need to know – SELECT

```
SELECT * FROM <keyspace name>.<table name>
      WHERE <query constraints>;
```

- Must include full partition key
- Partition keys do NOT support inequalities
- Not all clustering columns need be specified, but…
- Any preceding clustering columns *MUST* be specified

DATASTAX ACADEMY

Notebook Data Modeling: Cassandra-Land Project

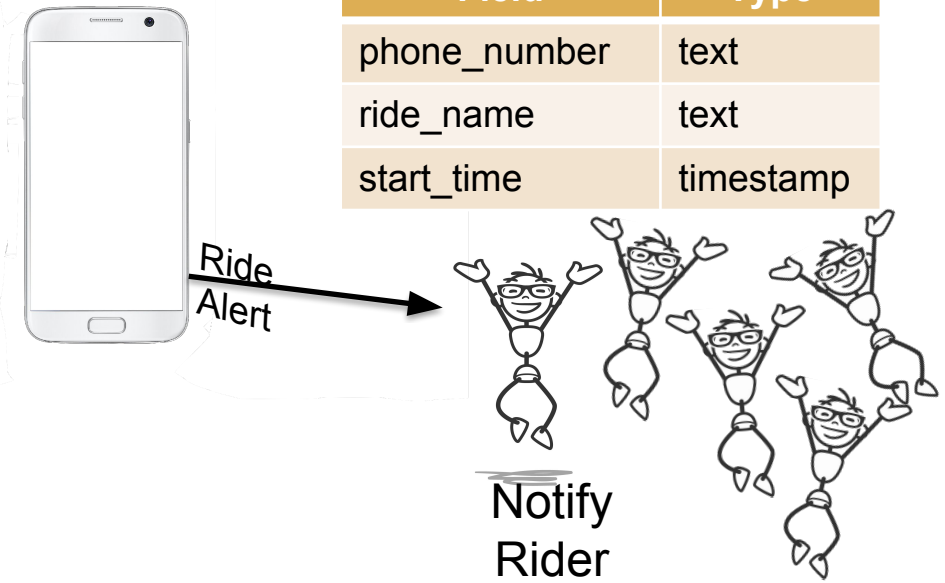
Do Part 2

Cassandra-Land

Ride Alert Use-Case

ride_instances_by_start_time

Field	Type
start_time	timestamp
ride_id	UUID↑
ride_name	text
user_id	UUID↑
phone_number	text



Ride Alert

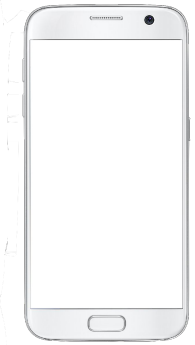
Field	Type
phone_number	text
ride_name	text
start_time	timestamp

Cassandra-Land

View Schedule Use-Case

ride_instances_by_user_id

Field	Type
user_id	UUID
start_time	timestamp↑
ride_id	UUID
ride_name	text

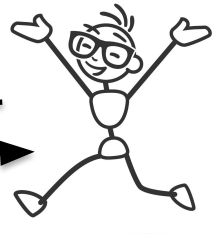


User ID

Field	Type
User ID	UUID

User ID

Schedule



View Schedule

Schedule

Field	Type
start_time	timestamp
ride_name	text

Cassandra-Land

Schedule Ride Use-Case

ride_list_by_location

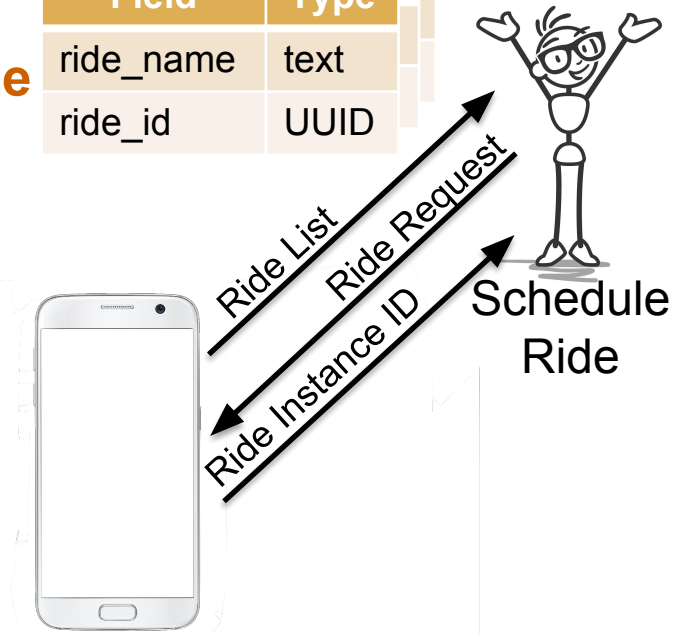
Field	Type
location	text
ride_id	UUID↑
ride_name	text
capacity	int

rider_count_by_time_and_ride

Field	Type
start_time	timestamp
ride_id	UUID
rider_count	int

Ride List

Field	Type
ride_name	text
ride_id	UUID



Ride Request

Field	Type
user_id	UUID
ride_id	UUID
start_time	timestamp

Ride Instance ID

Field	Type
ride_instance_id	UUID

Cassandra-Land

Table Summary

users_by_phone_number

Field	Type
phone_number	text
user_id	UUID
password	text

rider_count_by_time_and_ride

Field	Type
start_time	timestamp
ride_id	UUID
rider_count	int

ride_instances_by_start_time

Field	Type
start_time	timestamp
ride_id	UUID↑
ride_name	text
user_id	UUID↑
phone_number	text

ride_list_by_location

Field	Type
location	text
ride_id	UUID↑
ride_name	text
capacity	int

ride_instances_by_user_id

Field	Type
user_id	UUID
start_time	timestamp↑
ride_id	UUID
ride_name	text

Last Three Use-Cases

What you need to know – PRIMARY KEY()

PRIMARY KEY((<partition key column>,…), <clustering column>,...)

- Must have one or more partition key columns
- May have zero or more clustering columns

Last Three Use-Cases

What you need to know – Timestamps

Format is ‘YYYY-MM-DDTHH:MM:SS’

- Notice the quotes

Last Three Use-Cases

What you need to know – UPDATE

```
UPDATE <keyspace name>.<table name>
      SET <assignment>
      WHERE <row specification>
IF <condition>
```

- Can have multiple <assignment>
- IF is optional – causes a lightweight transaction

Last Three Use-Cases

What you need to know – BATCH

```
BEGIN BATCH
      INSERT statement
      INSERT statement
      …
APPLY BATCH
```

- Causes all operations to complete
- Use for inserting into multiple tables

DATASTAX ACADEMY

Notebook Data Modeling: Cassandra-Land Project

Do Part 3

DATASTAX ACADEMY

Cassandra-Land

What you learned!

- How to analyze use-cases to derive a data model
- How to denormalize to maintain performance
- How to use lightweight transactions
- How to leverage batch operations