CASSANDRA SUMMIT 2016

Artem Chebotko

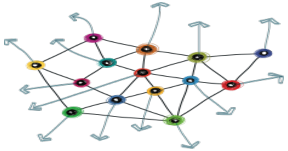
Graph Data Modeling in DataStax Enterprise

1	**DataStax Enterprise Graph**
2	Property Graph Data Model
3	Data Modeling Framework
4	Schema Optimizations

DSE Graph

- Real-time Graph DBMS
- Very large graphs
- Many concurrent users
- Proven technologies and standards
- OLTP and OLAP capabilities

DSE Graph Design

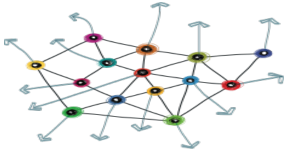


Graph Applications

DATASTAX
DSE Graph

CASSANDRA SUMMIT 2016

DSE Graph Design



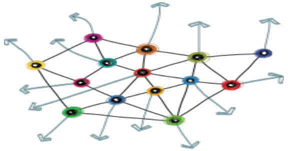
Graph Applications

Apache TinkerPop

Property Graph and Gremlin
DSE schema API

DATASTAX
DSE Graph

DSE Graph Design



Graph Applications

DSE Graph

Apache TinkerPop

Property Graph and Gremlin
DSE schema API

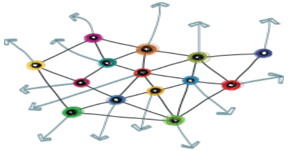
cassandra Solr

Fully integrated
backend technologies

Spark

CASSANDRA SUMMIT 2016

DSE Graph Design



Graph Applications

DATASTAX DSE Graph

Apache TinkerPop

Property Graph and Gremlin
DSE schema API

TOP SECRET

Schema, data, and query mappings
OLTP and OLAP engines

cassandra Solr

Fully integrated
backend technologies

Spark

CASSANDRA SUMMIT 2016

DSE Graph Use Cases

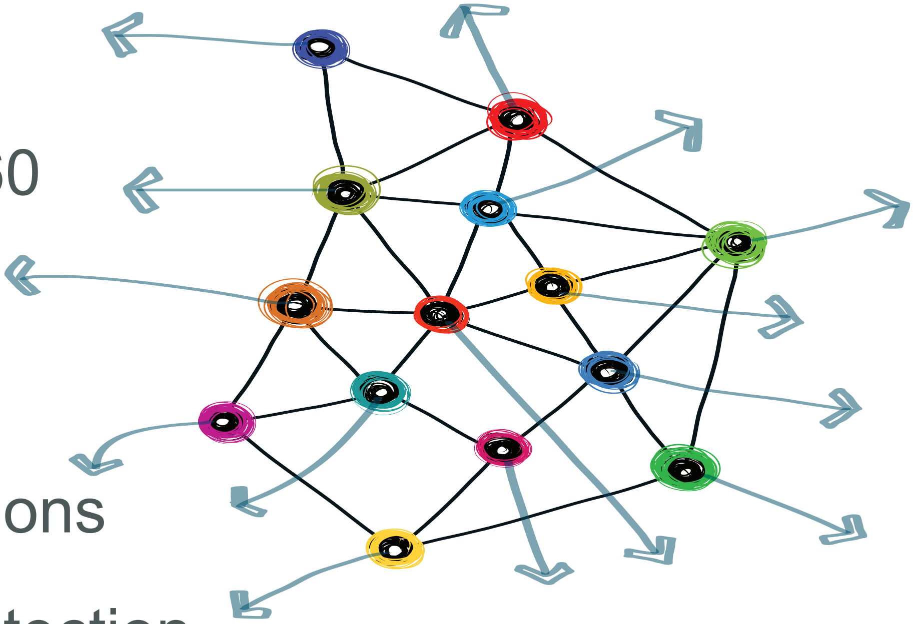
Internet of Things

Customer 360

Personalization

Recommendations

Fraud detection



CASSANDRA SUMMIT 2016

1 DataStax Enterprise Graph

2 **Property Graph Data Model**

3 Data Modeling Framework

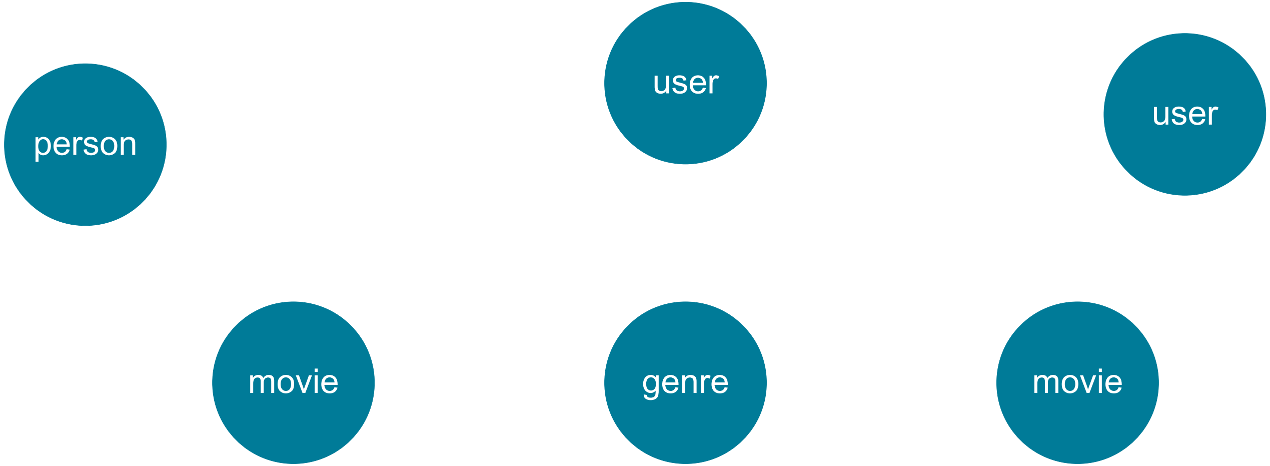
4 Schema Optimizations

CASSANDRA SUMMIT 2016

Property Graph Data Model

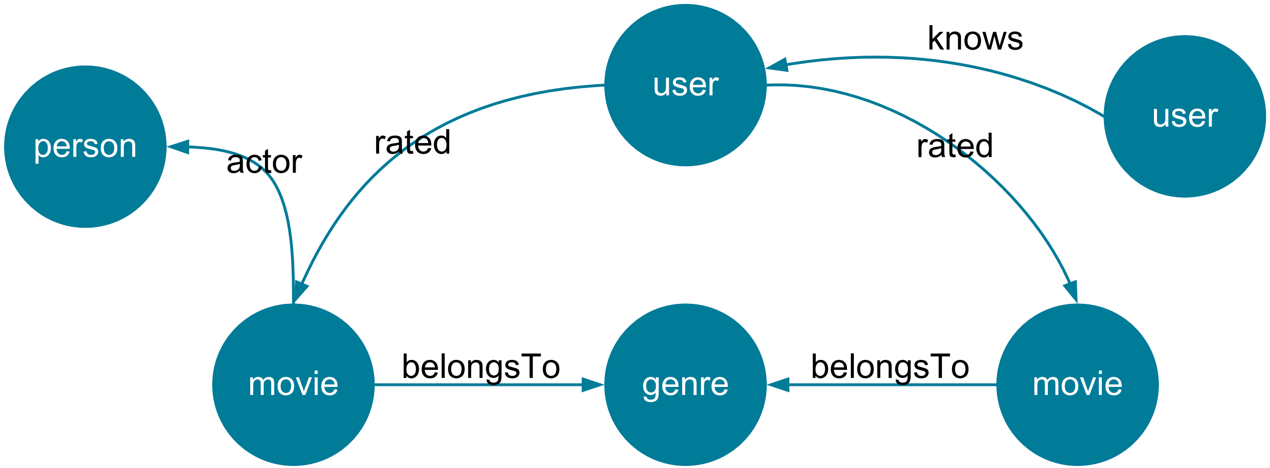
- Instance
  - Defined in *Apache TinkerPop*™
  - Vertices, edges, and properties
- Schema
  - Defined in *DataStax Enterprise*
  - Vertex labels, edge labels, and property keys

Vertices



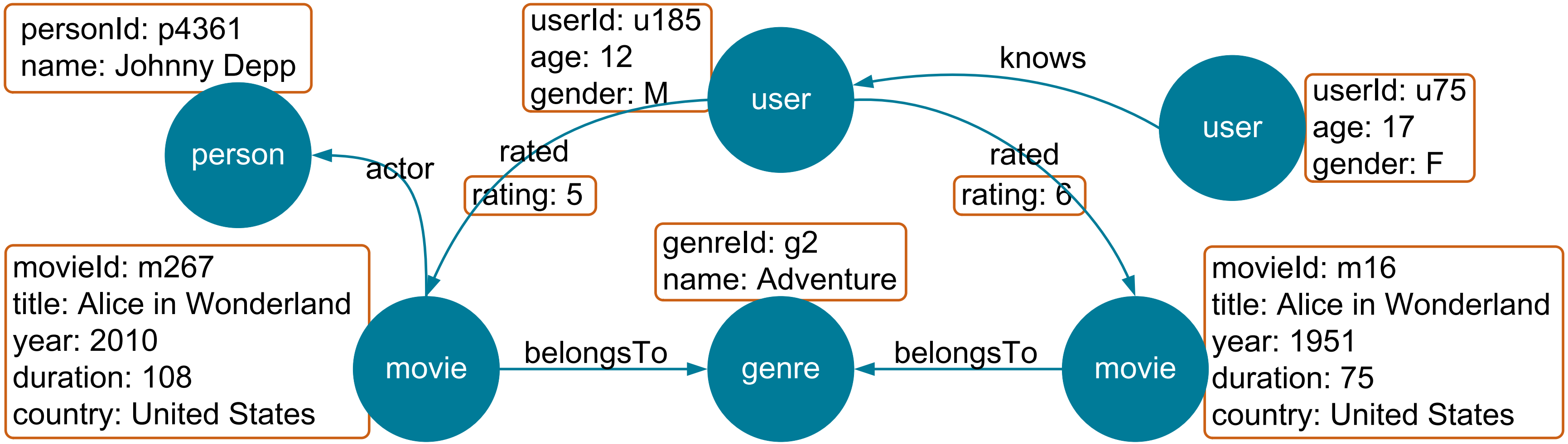
CASSANDRA SUMMIT 2016

Edges

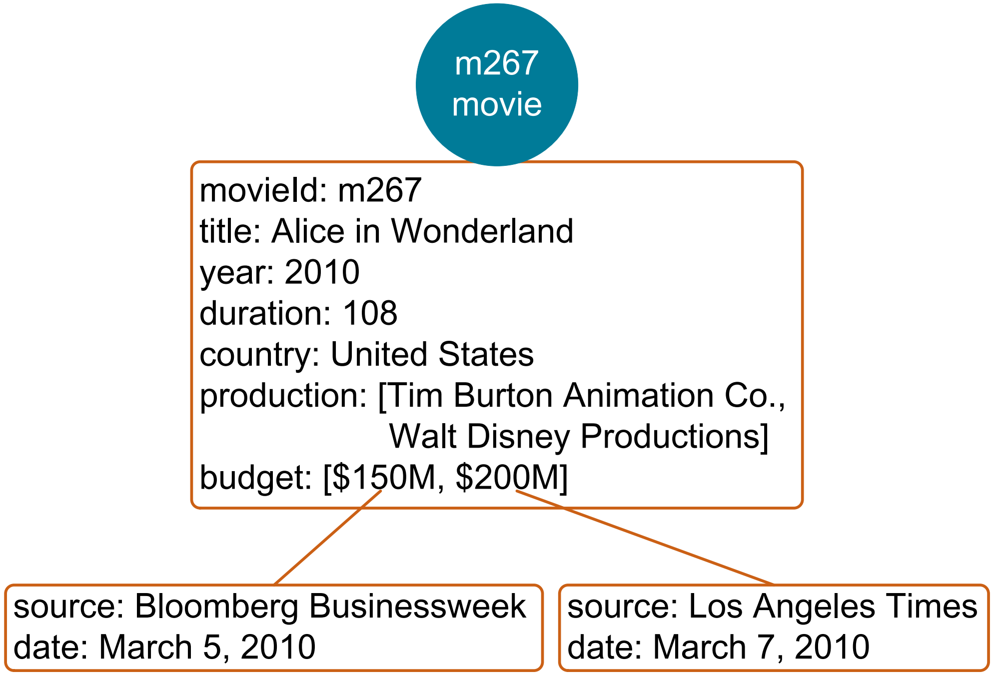


CASSANDRA SUMMIT 2016

Properties

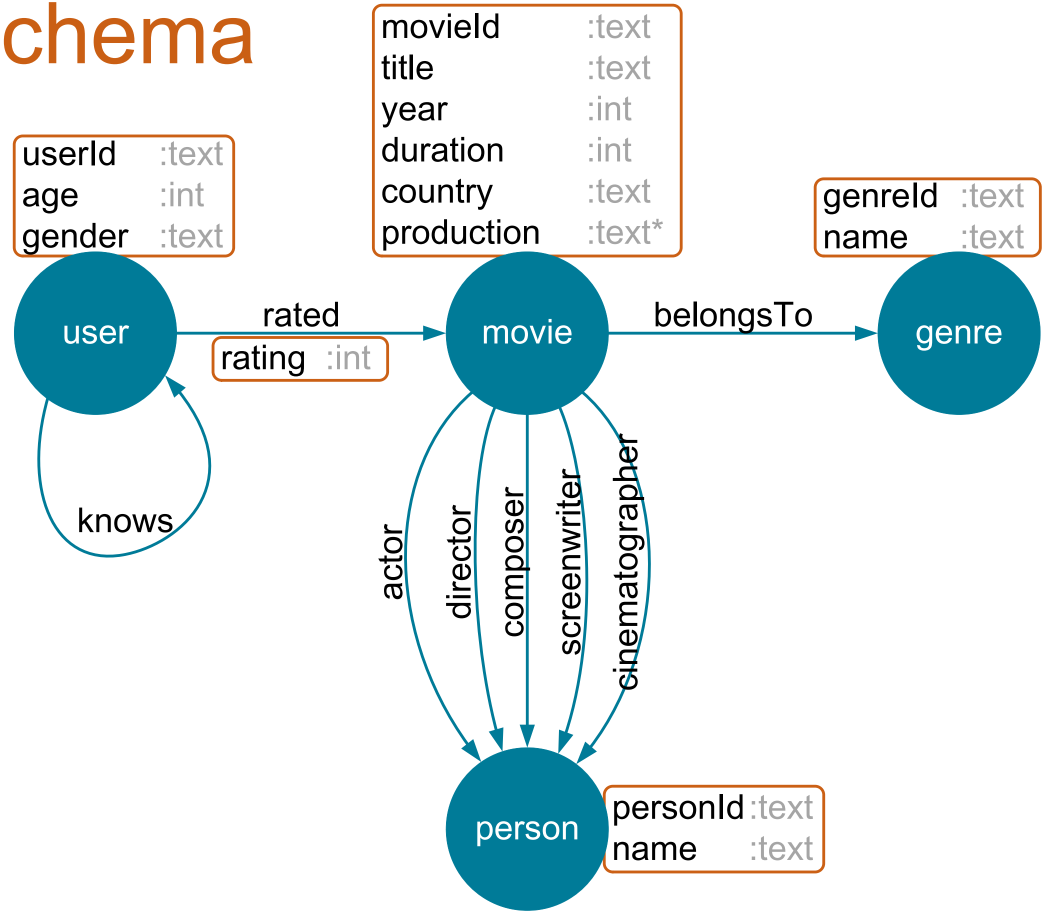

personId: p4361
name: Johnny Depp
person
actor
userId: u185
age: 12
gender: M
user
rated
rating: 5
knows
user
userId: u75
age: 17
gender: F
rated
rating: 6
movieId: m267
title: Alice in Wonderland
year: 2010
duration: 108
country: United States
movie
belongsTo
genreId: g2
name: Adventure
genre
belongsTo
movie
movieId: m16
title: Alice in Wonderland
year: 1951
duration: 75
country: United States

Multi- and Meta-Properties



CASSANDRA SUMMIT2016

Graph Schema

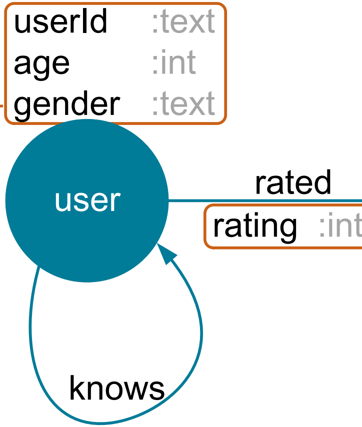


Importance of Graph Schema

- DSE needs a graph schema to generate a C* schema
  - Vertex labels → tables
  - Property keys → columns
  - Graph indexes → materialized views
secondary indexes
search indexes
- Additional data validation benefits

CASSANDRA SUMMIT 2016

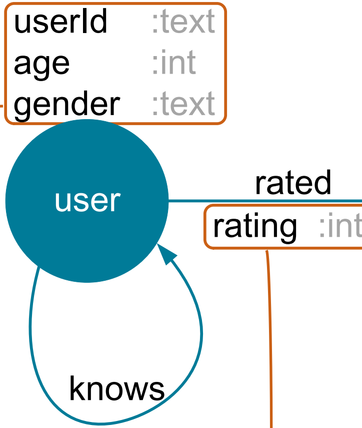
Schema Mapping Example



Property Table

```
CREATE TABLE user_p (
  community_id int,
  member_id bigint,
  "~~property_key_id" int,
  "~~property_id" uuid,
  age int,
  gender text,
  "userId" text,
  "~~vertex_exists" boolean,
PRIMARY KEY (community_id,
             member_id,
             "~~property_key_id",
             "~~property_id"))
```

Schema Mapping Example



Property Table

```
CREATE TABLE user_p (
  community_id int,
  member_id bigint,
  "~~property_key_id" int,
  "~~property_id" uuid,
  age int,
  gender text,
  "userId" text,
  "~~vertex_exists" boolean,
PRIMARY KEY (community_id,
             member_id,
             "~~property_key_id",
             "~~property_id"))
```

Adjacency Table

```
CREATE TABLE user_e (
  community_id int,
  member_id bigint,
  "~~edge_label_id" int,
  "~~adjacent_vertex_id" blob,
  "~~adjacent_label_id" smallint,
  "~~edge_id" uuid,
  "~rating" int,
  "~~edge_exists" boolean,
  "~~simple_edge_id" uuid,
PRIMARY KEY (community_id,
             member_id,
             "~~edge_label_id",
             "~~adjacent_vertex_id",
             "~~adjacent_label_id",
             "~~edge_id"))
```

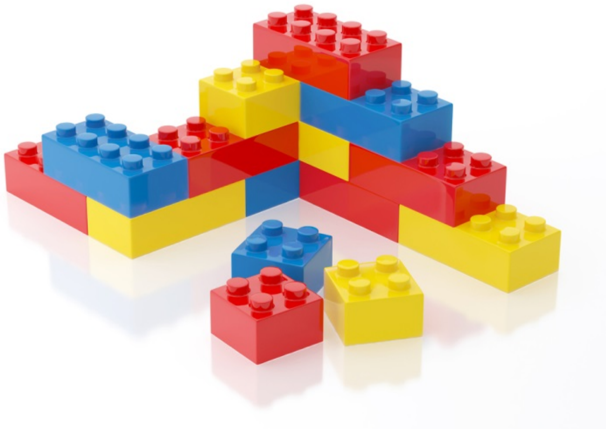

CASSANDRA SUMMIT 2016

1	DataStax Enterprise Graph
2	Property Graph Data Model
3	**Data Modeling Framework**
4	Schema Optimizations

CASSANDRA SUMMIT 2016

Data Modeling

- Process of organizing and structuring data
- Based on well-defined set of rules or methodology
- Results in a graph or database schema
- Affects data quality, data storage and data retrieval



CASSANDRA SUMMIT 2016

Traditional Schema Design

Data Model

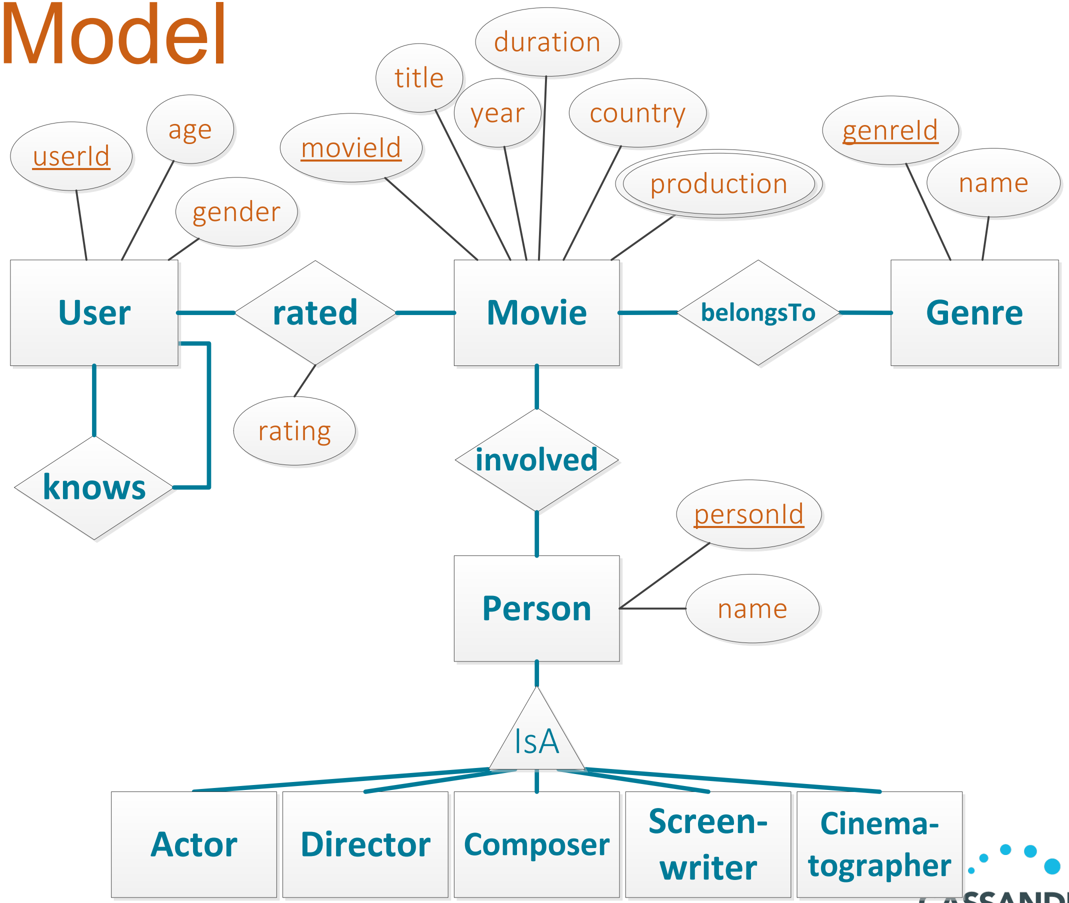
- Conceptual Data Model (CDM)
- Logical Data Model (LDM)
- Physical Data Model (PDM)

Purpose

- Understand data and its applications
- Sketch a graph data model
- Optimize physical design

Conceptual Data Model

- Entity types
- Relationship types
- Attribute types



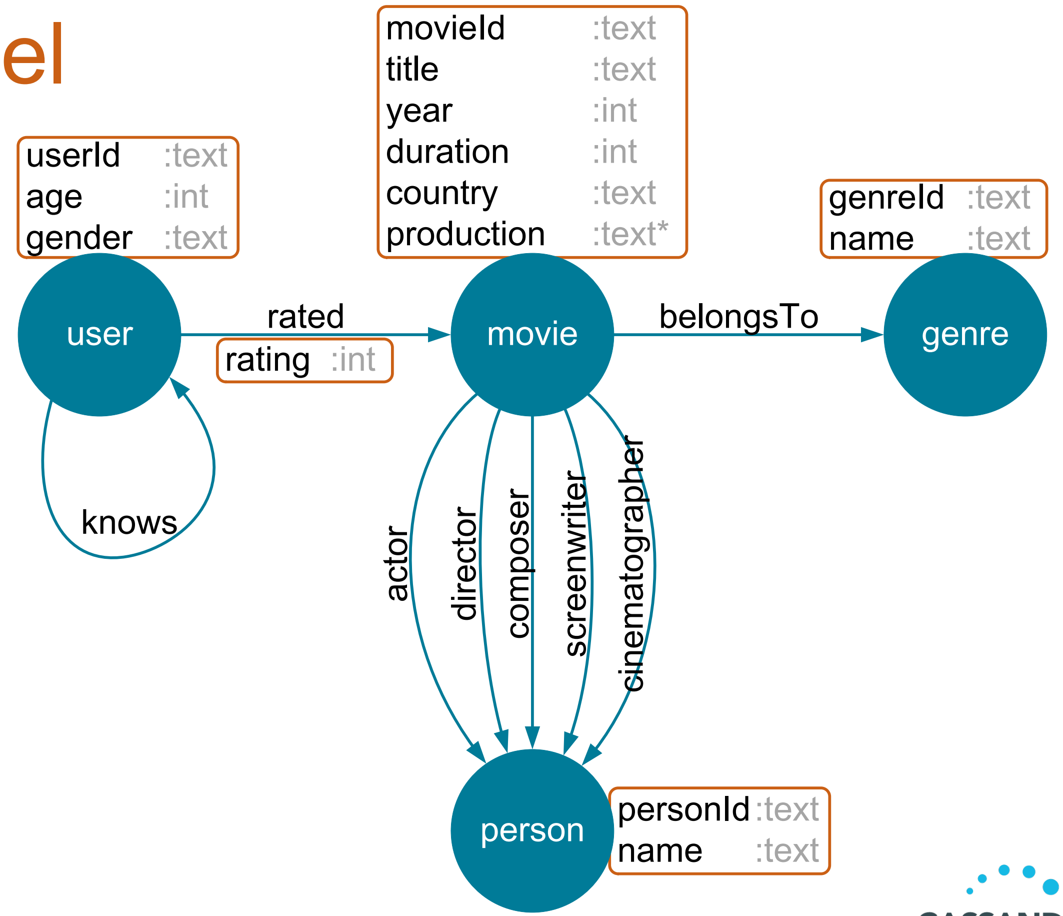
CASSANDRA SUMMIT 2016

Transition from CDM to LDM

- Both CDM and LDM are graphs
  - Entity types → Vertex labels
  - Relationship types → Edge labels
  - Attribute types → Property keys
- Mostly straightforward with a few nuances

Logical Data Model

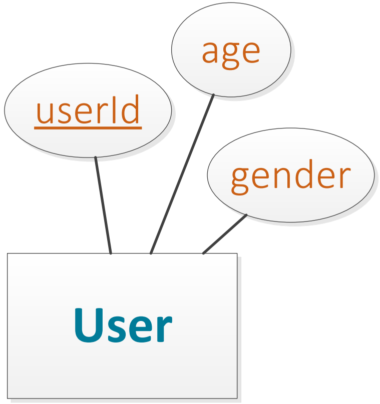
- Vertex labels
- Edge labels
- Property keys

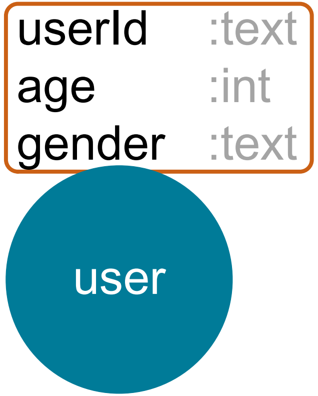


CASSANDRA SUMMIT 2016

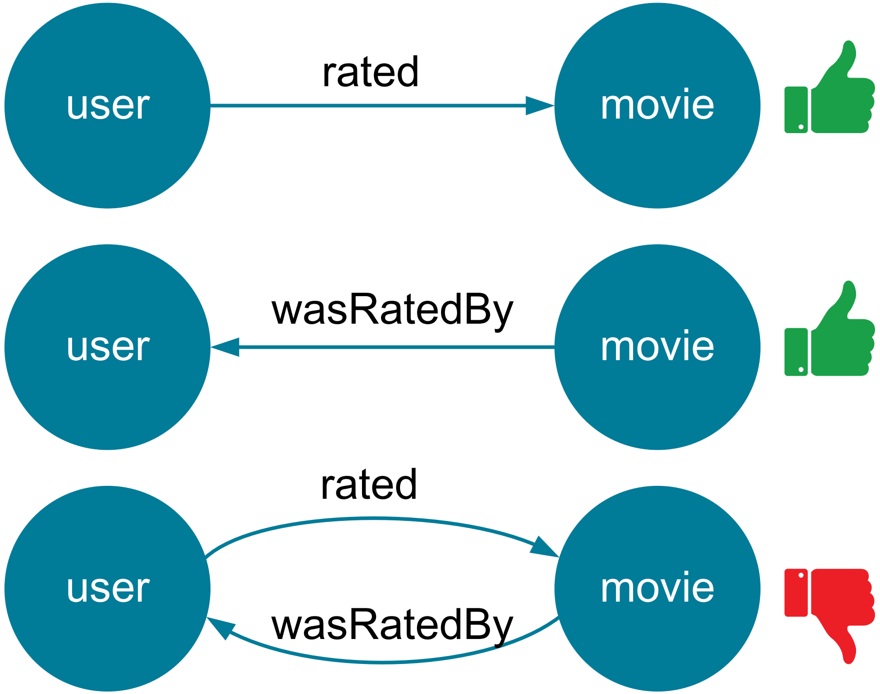
Keys

- Entity type keys → Property keys
  - Uniqueness is not enforced
  - Vertex IDs are auto-generated
- Entity type keys → Custom vertex IDs
  - Uniqueness is enforced
  - Overriding default partitioning
  - Advanced feature



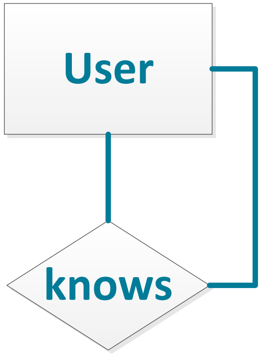


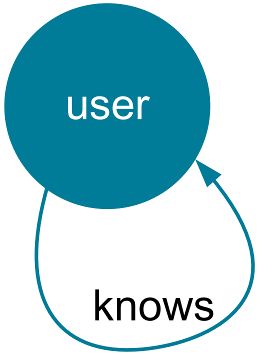
Symmetric Relationships

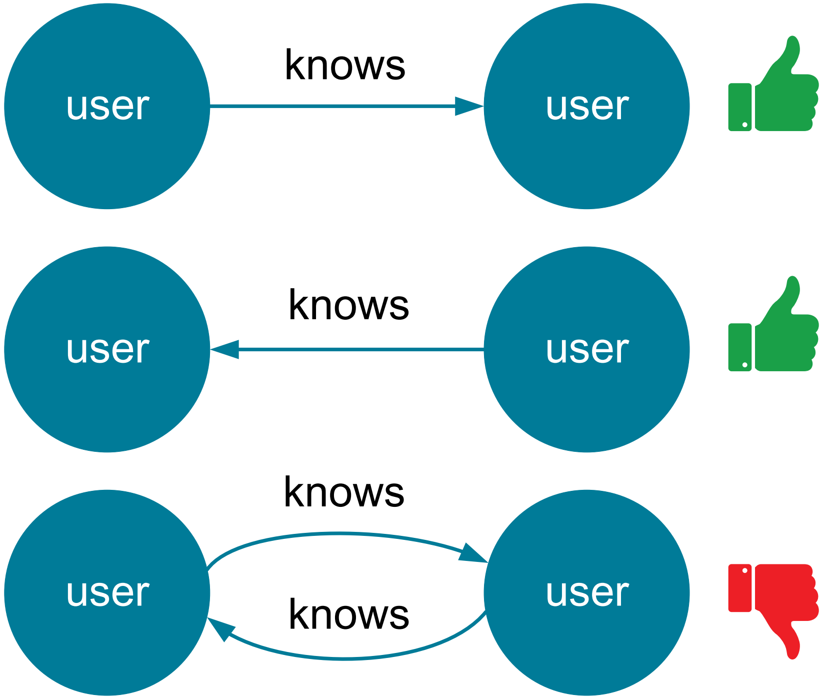


CASSANDRA SUMMIT 2016

Bi-Directional Relationships

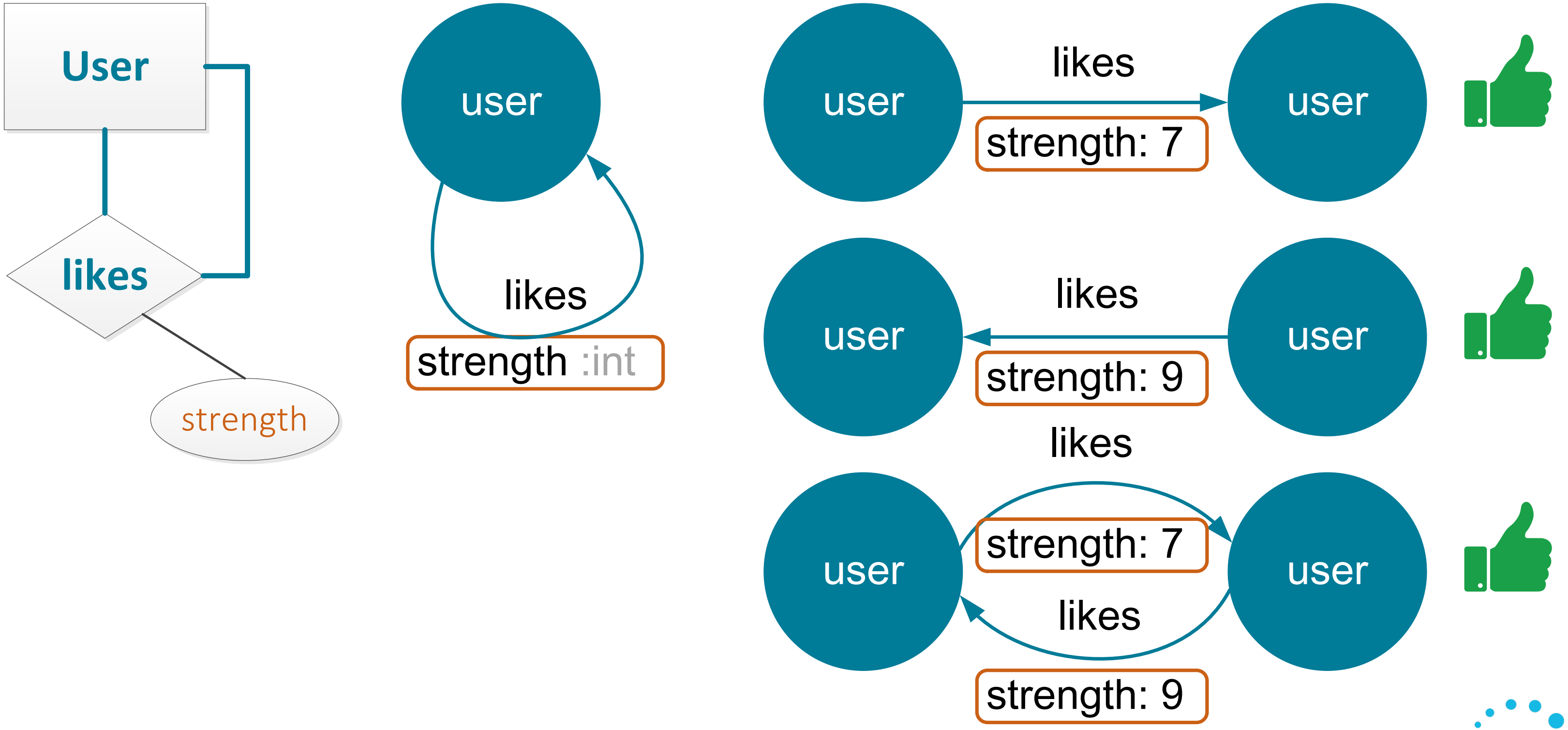






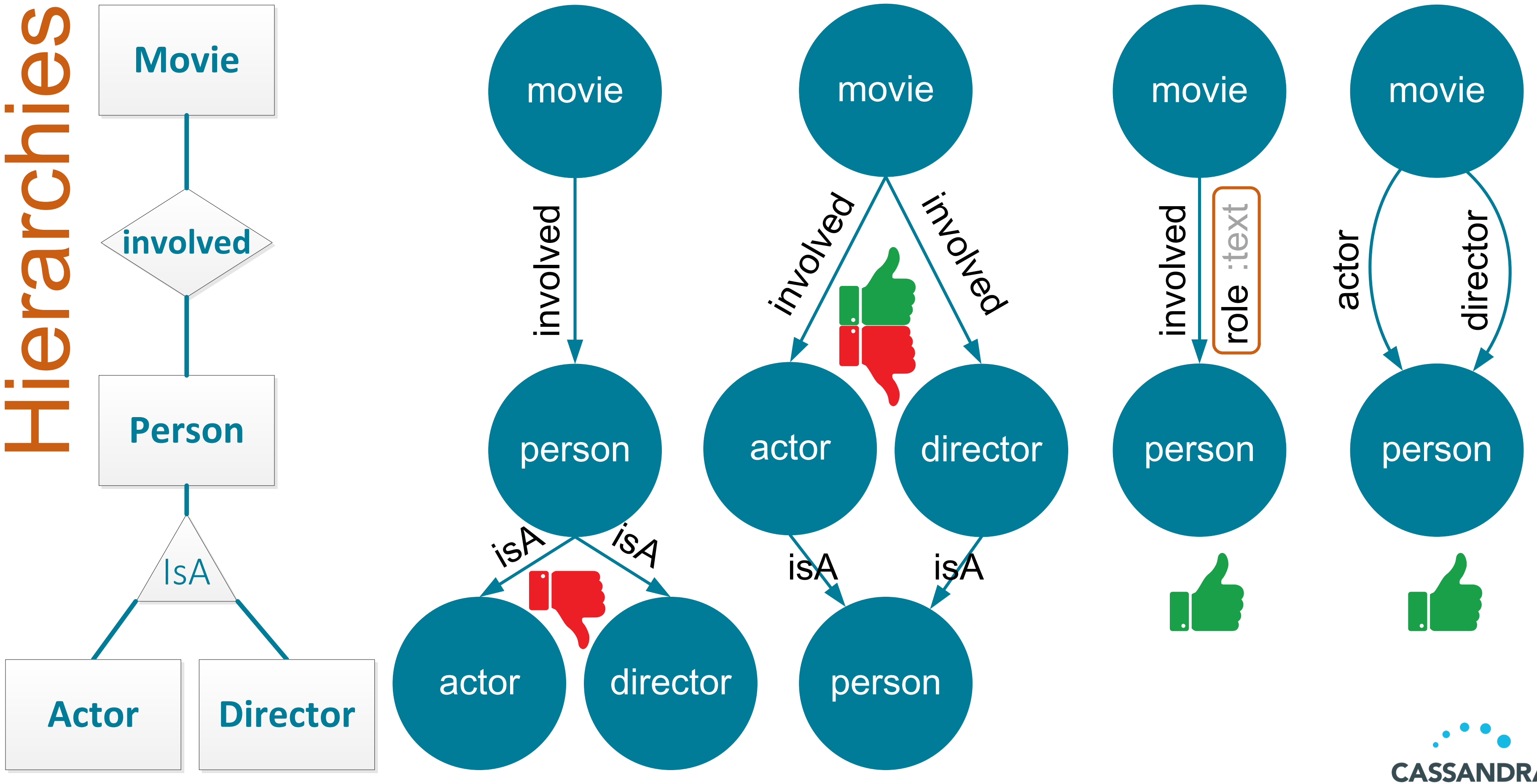
CASSANDRA SUMMIT 2016

Qualified Bi-Directional Relationships



CASSANDRA SUMMIT 2016

Hierarchies



Physical Data Model

```
schema.propertyKey("userId").Text().create()
schema.propertyKey("name").Text().create()
schema.propertyKey("age").Int().create()

schema.vertexLabel("user").properties("userId","age",…).create()
schema.vertexLabel("movie").properties("movieId",…).create()

schema.edgeLabel("knows").connection("user","user").create()
schema.edgeLabel("rated").single().properties("rating")
                              .connection("user","movie").create()
```

CASSANDRA SUMMIT 2016

1	DataStax Enterprise Graph
2	Property Graph Data Model
3	Data Modeling Framework
4	**Schema Optimizations**

CASSANDRA SUMMIT 2016

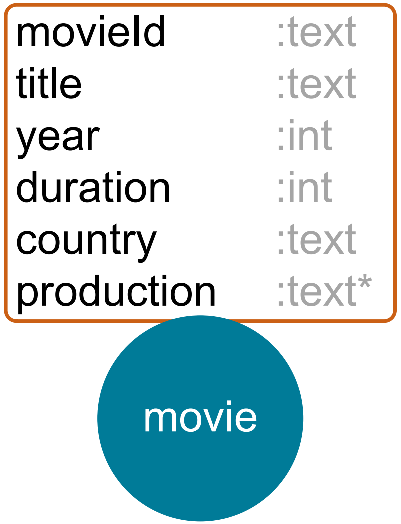
Optimizing PDM for Performance

- Indexing data
- Controlling partitioning
- Materializing aggregates and inferences
- Rewriting traversals

Vertex Indexes

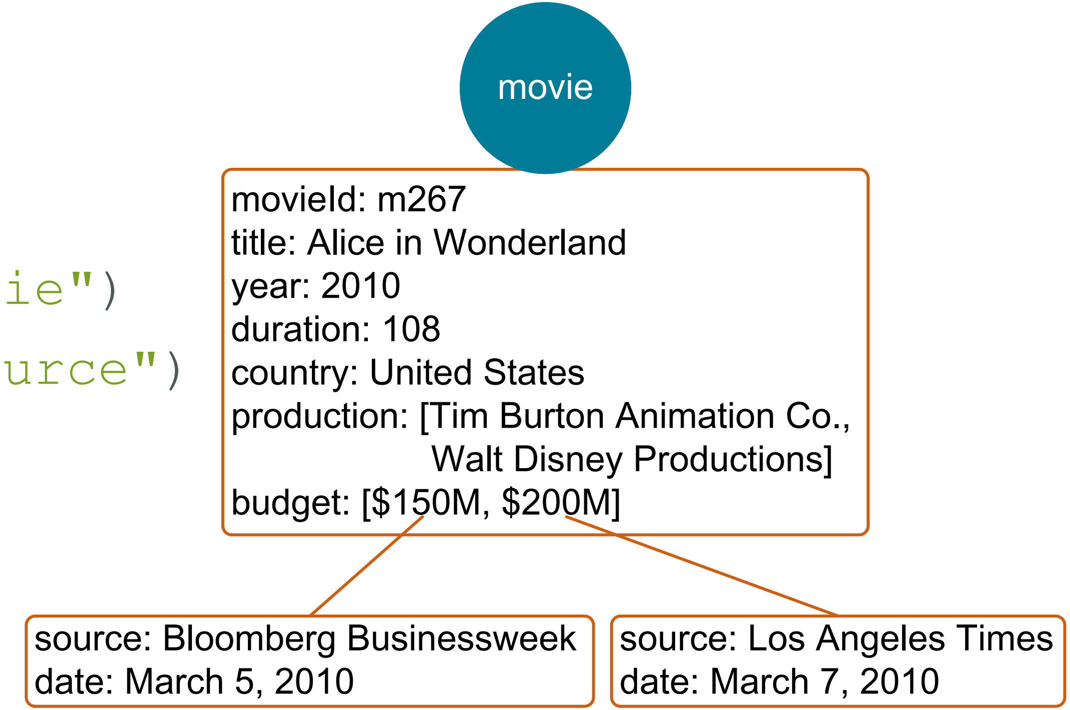
```
schema.vertexLabel("movie")
.index("moviesById")
.materialized()
.by("movieId")
.add()

g.V().has("movie","movieId","m267")
```



Property Indexes

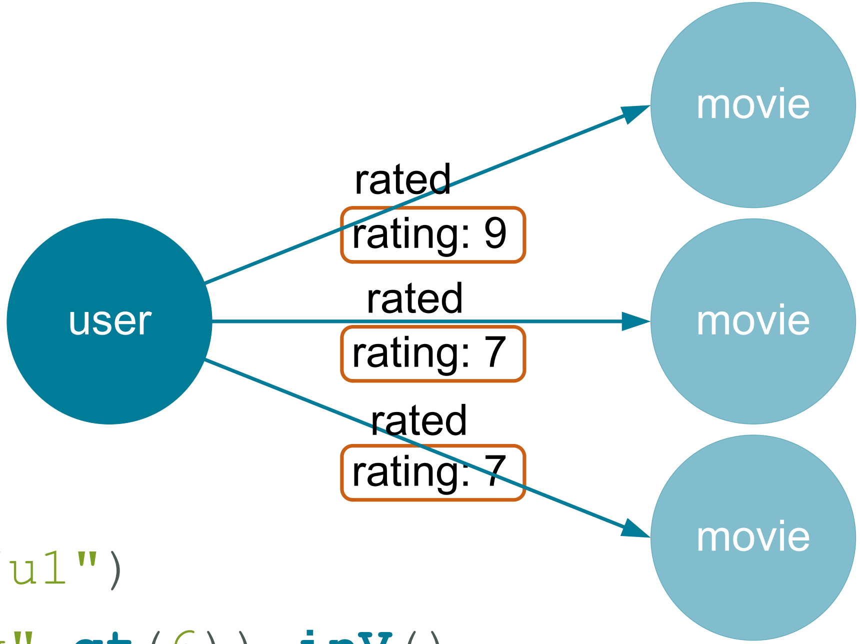
```
schema.vertexLabel("movie")
.index("movieBudgetBySource")
.property("budget")
.by("source")
.add()
```




```
g.V().has("movie","movieId","m267")
  .properties("budget")
  .has("source","Los Angeles Times").value()
```

Edge Indexes

```
schema.vertexLabel("user")
.index("toMoviesByRating")
.outE("rated")
.by("rating")
.add()
```




```
g.V().has("user","userId","u1")
  .outE("rated").has("rating",gt(6)).inV()
```


CASSANDRA SUMMIT 2016

Custom Partitioning

```
schema.vertexLabel("movie")
.partitionKey("year","country")
.clusteringKey("movieId")
.properties("title","duration")
.create()
```

movie_e	
year	K
country	K
movieId	C↑
~~edge_label_id	C↑
~~adjacent_vertex_id	C↑
~~adjacent_label_id	C↑
~~edge_id	C↑
~~edge_exists	
~~simple_edge_id	

movie_p	
year	K
country	K
movieId	C↑
~~property_key_id	C↑
~~property_id	C↑
duration	
title	
~~vertex_exists	

CASSANDRA SUMMIT 2016

Materializing Aggregates

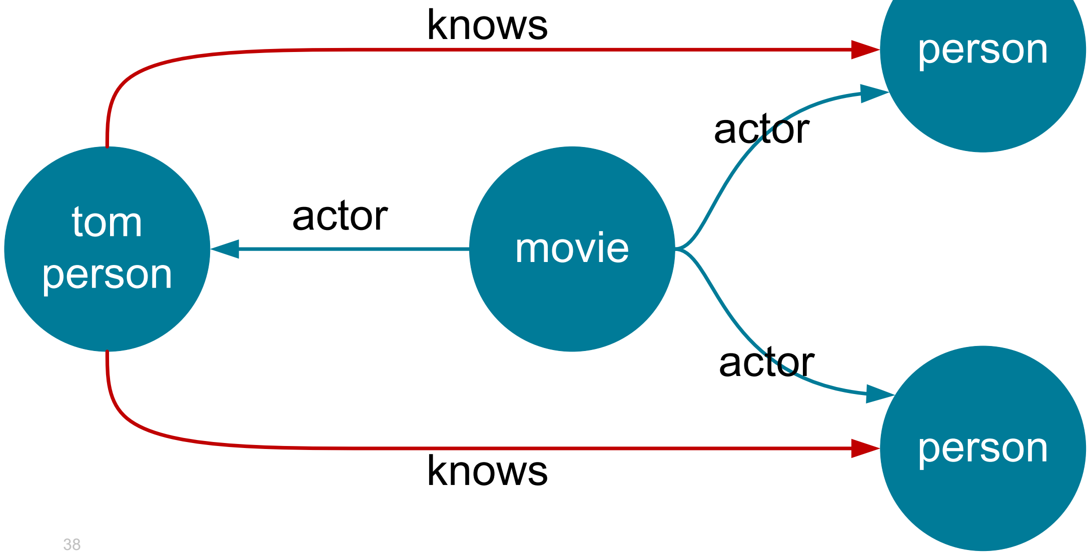
```
g.V().hasLabel("movie")
.property("avg",_.inE("rated")
                .values("rating")
                .mean())
```

movieId	:text
title	:text
year	:int
duration	:int
country	:text
production	:text*
avg	:float

movie

Materializing Inferences

```
g.V().has("person","name","Tom Hanks").as("tom")
.in("actor").out("actor").where(neq("tom")).dedup()
.addE("knows").from("tom")
```



Rewriting Traversals

- Equivalent results
- Different execution plans
- Different response times

```
g.V().has("movie","year",2010).out("actor")
     .has("name","Johnny Depp").count()

g.V().has("person","name","Johnny Depp").in("actor")
     .has("year",2010).count()
```


CASSANDRA SUMMIT 2016

Profiling Traversals

```
gremlin> g.V().has("person","name","Johnny Depp").in("actor").has("year",2010).count().profile()
==>Traversal Metrics
Step                                                      Count  Traversers       Time (ms)    % Dur
=============================================================================================================
DsegGraphStep([~label.eq(person), name.eq(Johnn...            1           1           0.838    20.73
  query-optimizer                                                                     0.145
  query-setup                                                                         0.051
  index-query                                                                         0.210
DsegVertexStep(IN,[actor],vertex)                            14          14           0.611    15.12
  query-optimizer                                                                     0.031
  query-setup                                                                         0.000
  vertex-query                                                                        0.193
HasStep([year.eq(2010)])                                      1           1           2.542    62.83
CountGlobalStep                                               1           1           0.053     1.32
                                            >TOTAL            -           -           4.046        -
```


CASSANDRA SUMMIT 2016

Thank You

Artem Chebotko
achebotko@datastax.com
www.linkedin.com/in/artemchebotko

LEARN MORE
academy.datastax.com

CASSANDRA SUMMIT 2016

The End