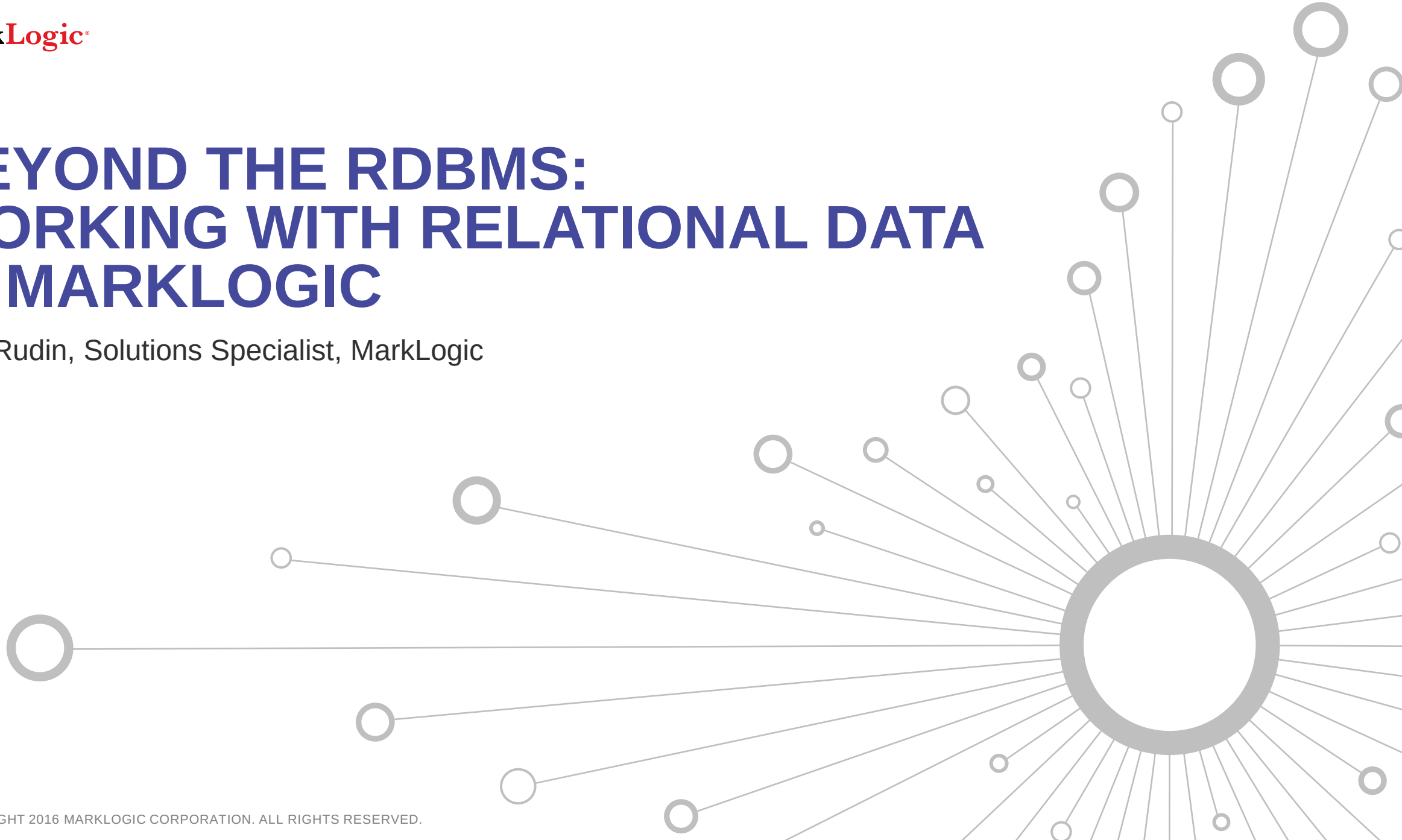
MarkLogic

BEYOND THE RDBMS: WORKING WITH RELATIONAL DATA IN MARKLOGIC

Rob Rudin, Solutions Specialist, MarkLogic



MarkLogic

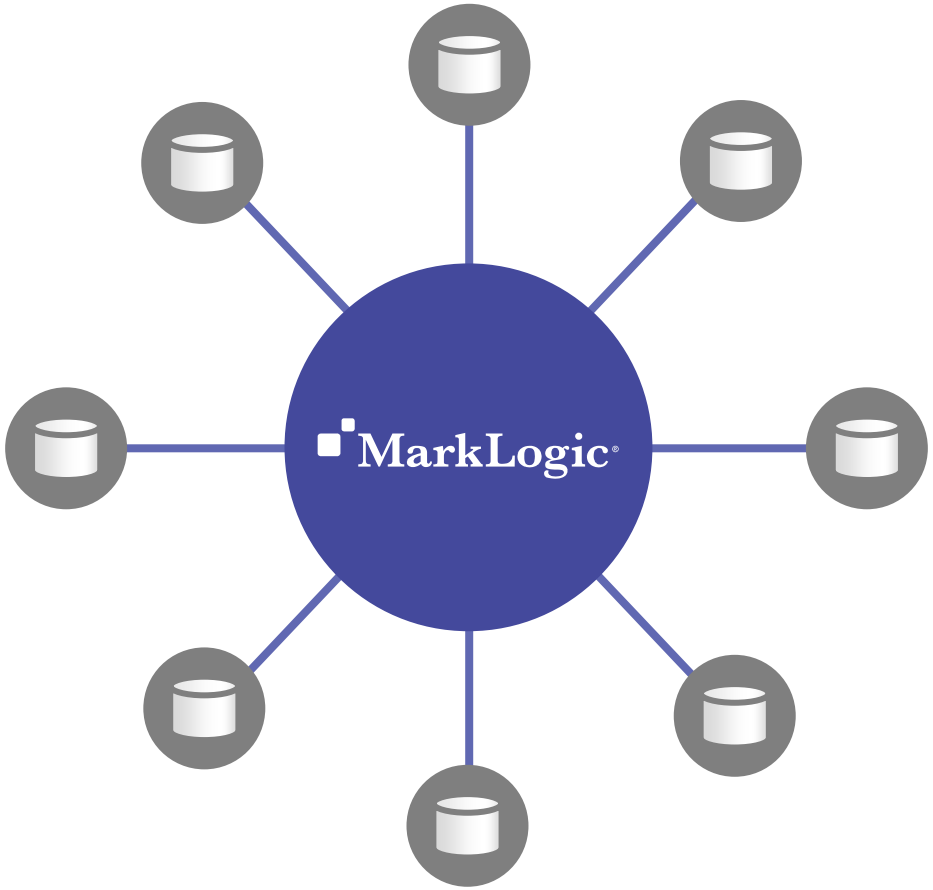
Agenda

- Introduction
- The problem – getting relational data into MarkLogic
- Demo how to do this

MarkLogic

Introduction

- Rob Rudin
- Solutions Architect at MarkLogic, 2+ years
- Former MarkLogic customer, 5+ years
- https://github.com/rjrudin



THE DESIRED SOLUTION

A Database That Integrates Data Better, Faster, with Less Cost

MarkLogic

The MarkLogic Alternative

An *Operational and Transactional* Enterprise NoSQL Database

EASY TO GET DATA IN

Flexible Data Model

- Data ingested as is (no ETL)
- Structured and unstructured data
- Data and metadata together
- Adapts to changing data and changing data structures

EASY TO GET DATA OUT

Ask Anything Universal Index

- Index once and query endlessly
- Real-time and lightning fast
- Query across JSON, XML, text, geospatial, and semantic triples in one database

100% TRUSTED

Enterprise Ready

- Reliable data and transactions (100% ACID compliant)
- Out-of-the-box automatic failover, replication, and backup/recovery
- Enterprise-grade security and Common Criteria certified

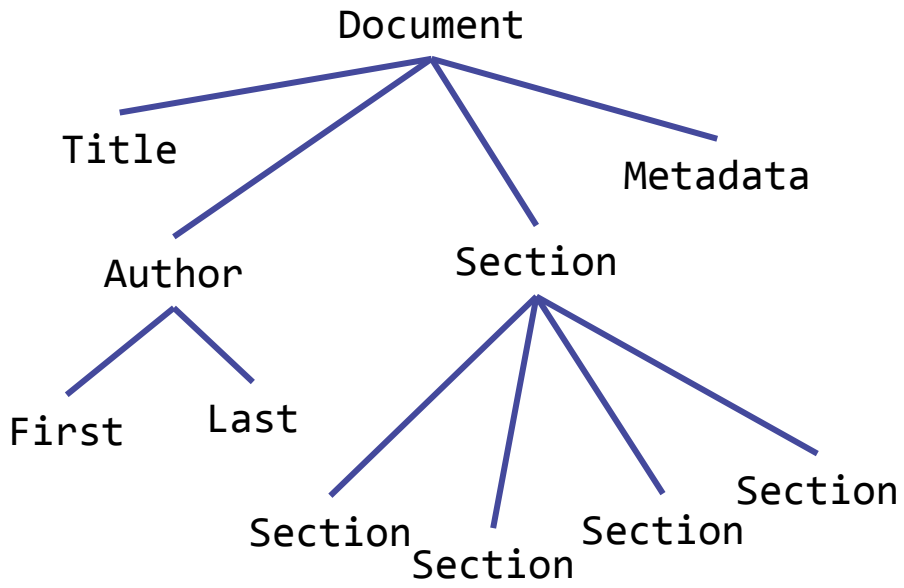
MarkLogic

Easy To Get Data In

- Structured and unstructured data Yes
- Adapts to changing data and changing data structures Absolutely
- Data ingested as-is Let's see...

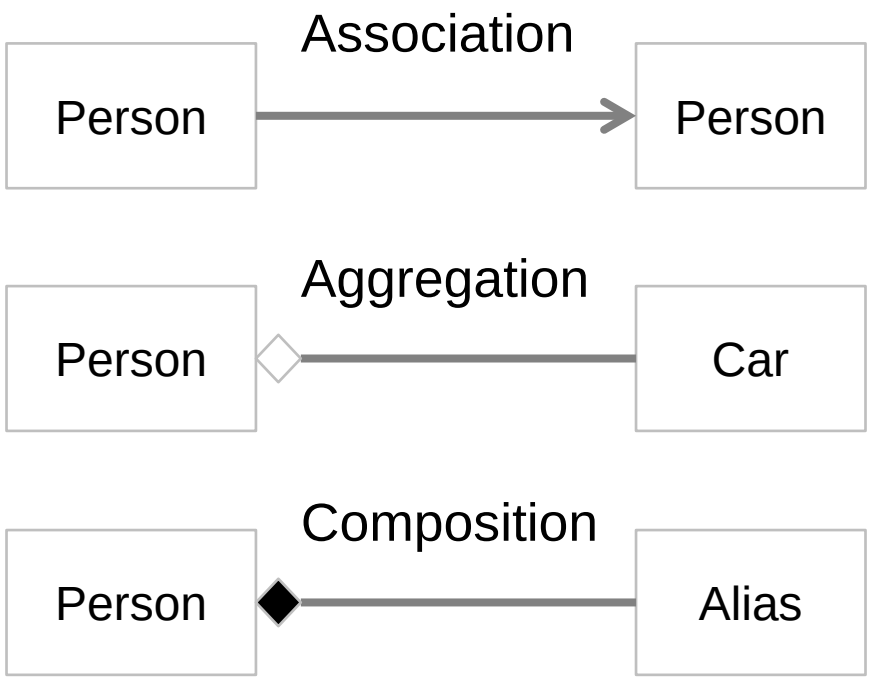
Tables and Documents

- Relational = 2-dimensional tables
- Documents = hierarchical with nesting
- We can load tables as-is; each row becomes a document
- But we're missing out on the benefits of documents



Why Documents?

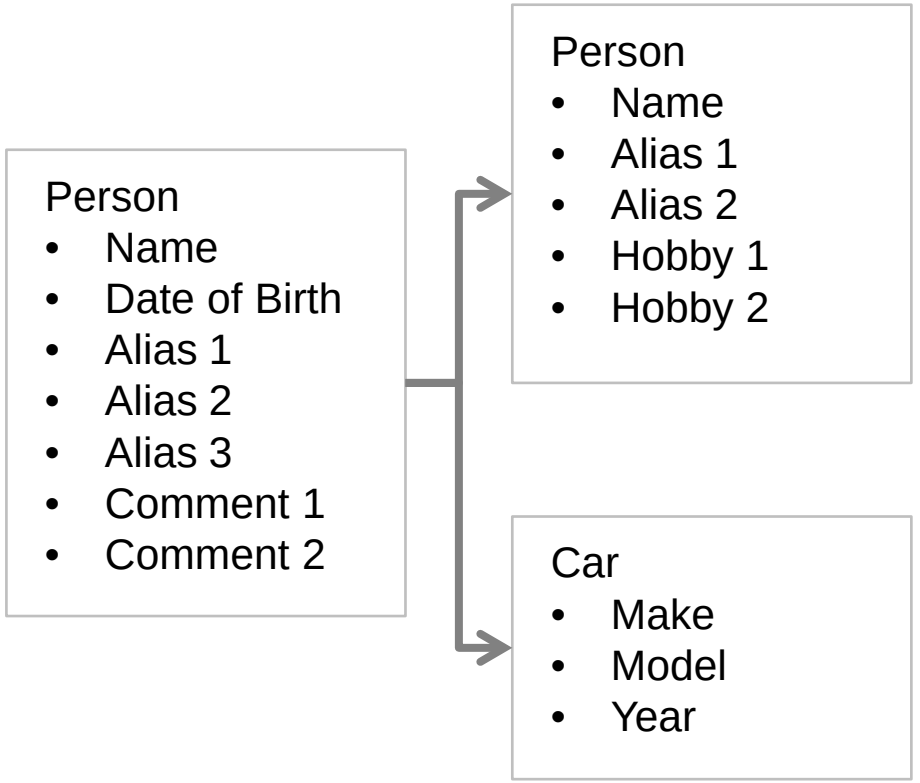
- Our applications have conceptual models
- A conceptual model has entities with different kinds of relationships
- Relational forces us to break up our model into separate tables for every 1:many relation
  - Sometimes makes sense, but not always
  - We never have a choice



MarkLogic

Documents Can Mirror Our Conceptual Model

- We can model hierarchical entities in our business model
- And then preserve them in our physical model
- We still have relationships
- But we have choices now for modeling them



MarkLogic

Documents in MarkLogic Are More Flexible

- Can query the entire database, not limited to a single table
- Can group documents into multiple collections
- Can insert/update/delete many different documents in a single transaction
- Can ingest different datasets without modeling them the same way
  - Key aspect of data integration

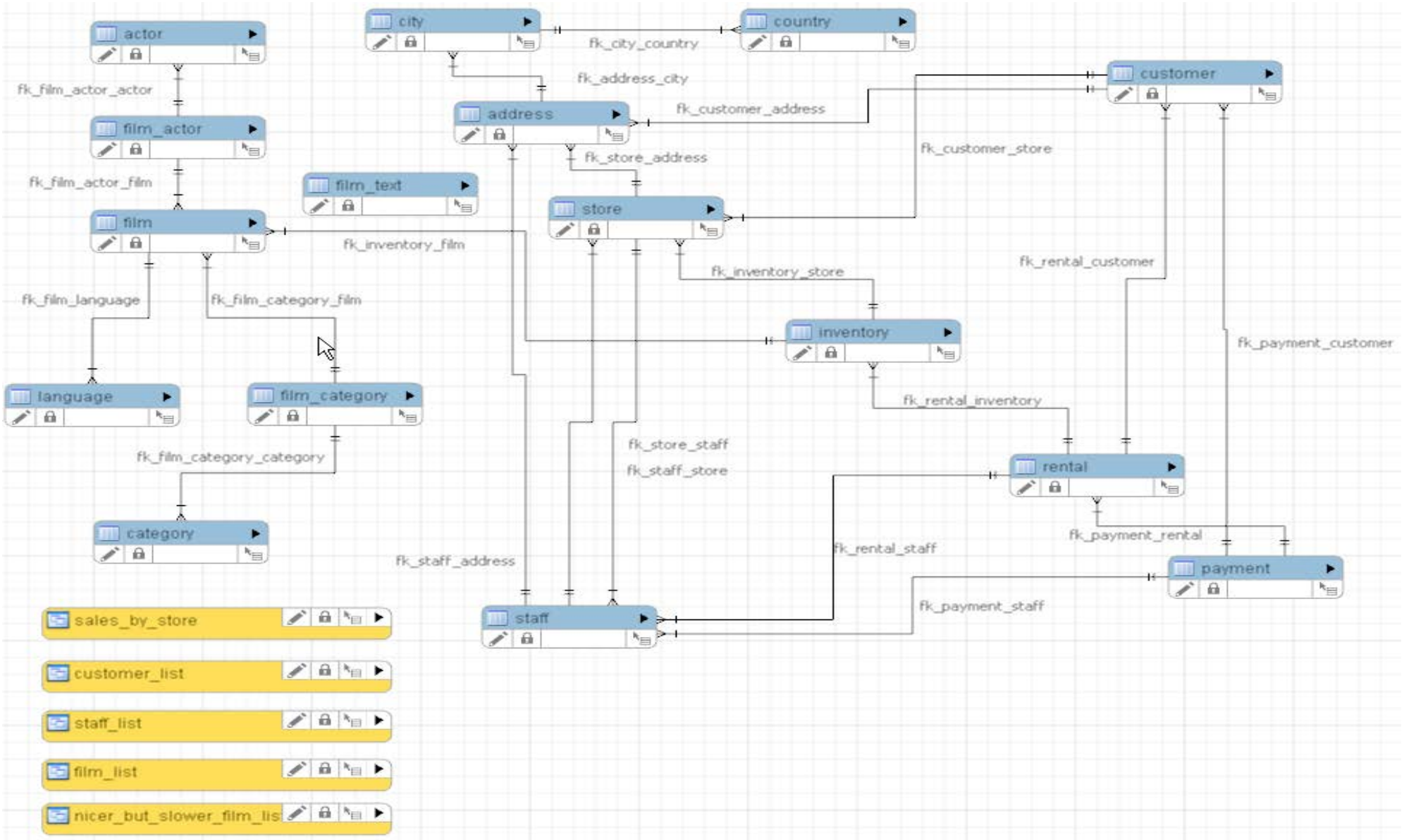
Problem Summary

- We want a unified, actionable 360º view of data
- So we need to integrate our data silos
- Our data silos are mostly relational
- MarkLogic holds documents
- We see the benefits of MarkLogic and documents
- But how do we actually go from tables to documents?

MarkLogic

Our Sample Data

- MySQL Sakila dataset
  - https://dev.mysql.com/doc/sakila/en/
  - Represents a DVD rental store
  - Used for testing MySQL-specific functionality and new features
  - Pre-loaded in MySQL install
- Perfect for testing out tables ➞ documents



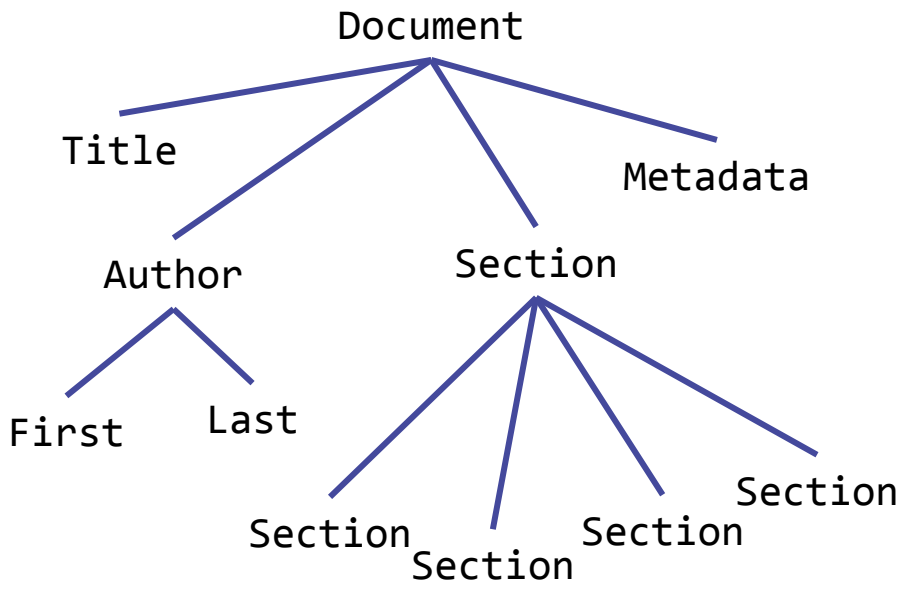
MarkLogic

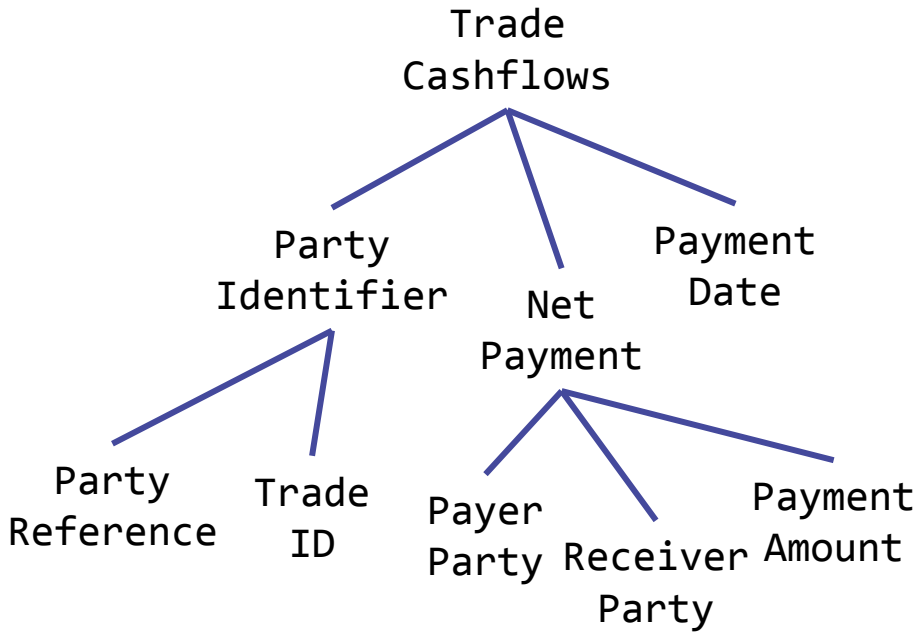
Moving From Tables to Documents

- Need to understand data modeling options in MarkLogic
- Need to identify entities
  - What are the high level nouns in our system?
  - Those typically drive the types of documents we'll have

MarkLogic Data Modeling 101

- MarkLogic is a document-oriented database
  - Supports any-structured data via hierarchical data model
  - Stores compressed trees





MarkLogic

MarkLogic Data Modeling 101

- Documents have URIs
  - Can be any string, they just need to be unique
  - Analogous to a primary key, but scoped to the entire database as opposed to a table

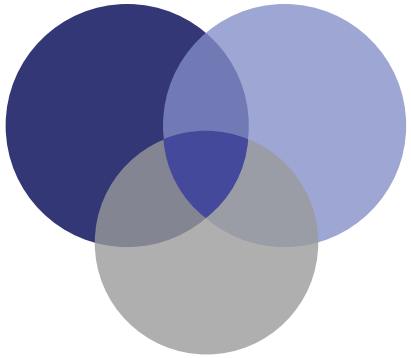
MarkLogic

MarkLogic Data Modeling 101

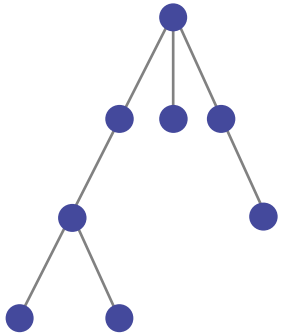
- Prefer UUIDs over sequence numbers
  - Can use `sem:uuid-string()`
  - Avoids namespace problems; i.e. ID of `14` from different databases
- Often looks like a path, but not required to be
  - Example - `/film/123.xml` or just `123.xml` or just `123`

MarkLogic

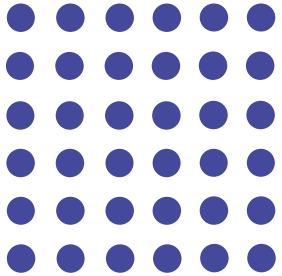
MarkLogic Data Modeling 101



COLLECTIONS
Set-based, n:m relationship



DIRECTORIES
Exclusive, hierarchical, analogous to file system
Based on URI



SECURITY
Roles and document-level permissions invisible to your app

MarkLogic

Identifying Entities

- What are the main nouns that users talk and ask questions about?
- For our DVD rental store:
  - As a customer, I want to search for films with certain actors
  - As a customer, I want to find a store with a particular film
  - As a staff member, I want to find overdue rentals
- Nouns – films, actors, stores, rentals, customers, staff members
- Those are the collections of documents we'll have

MarkLogic

Let's Start Migrating Data

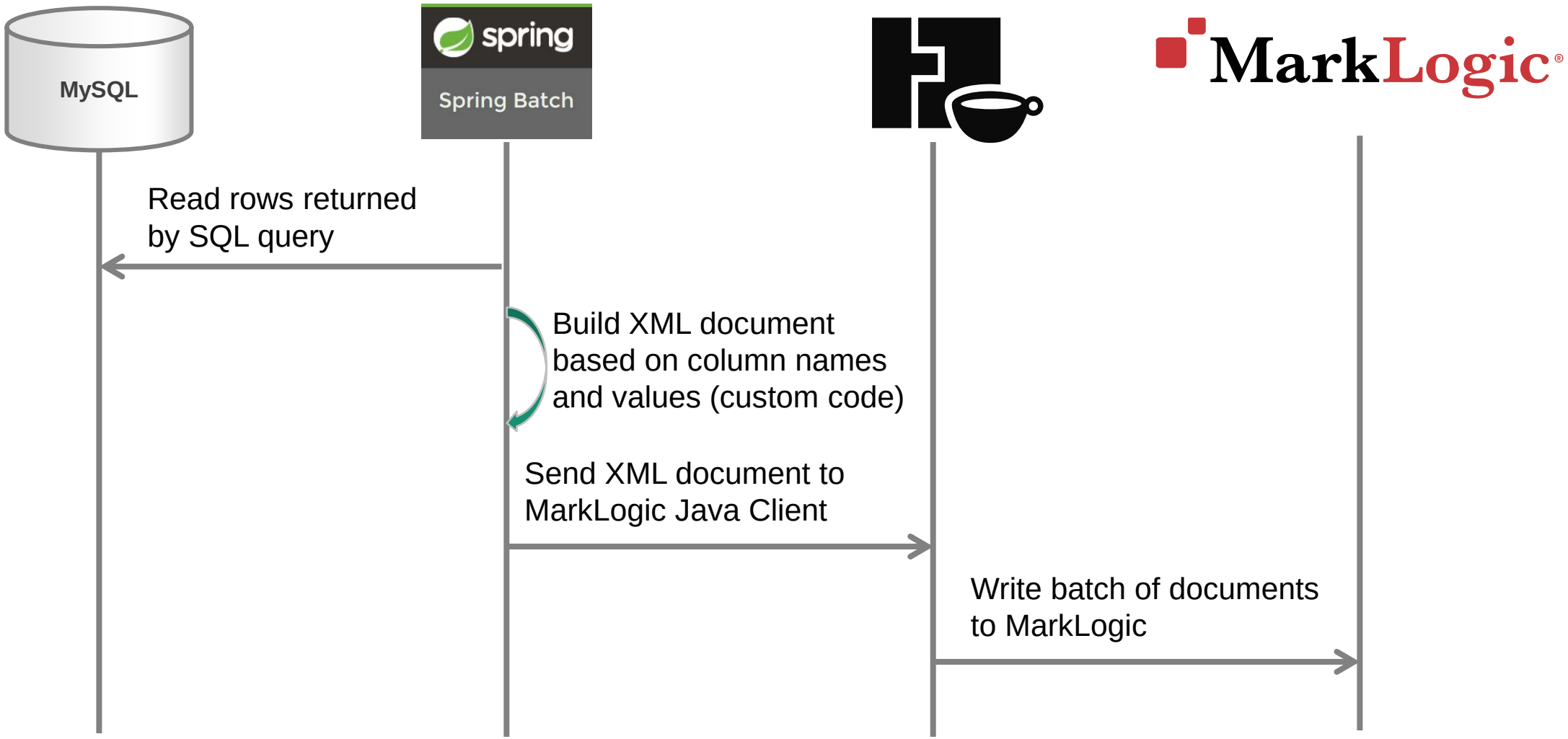
- Demo created using a slush generator
  - https://github.com/rjrudin/slush-marklogic-spring-boot
  - Angular / Spring Boot / MarkLogic
- Spring Boot middle tier uses Spring Batch to migrate data
- We'll start with actors, then bring in films

MarkLogic

Spring Batch for Bulk Migrations

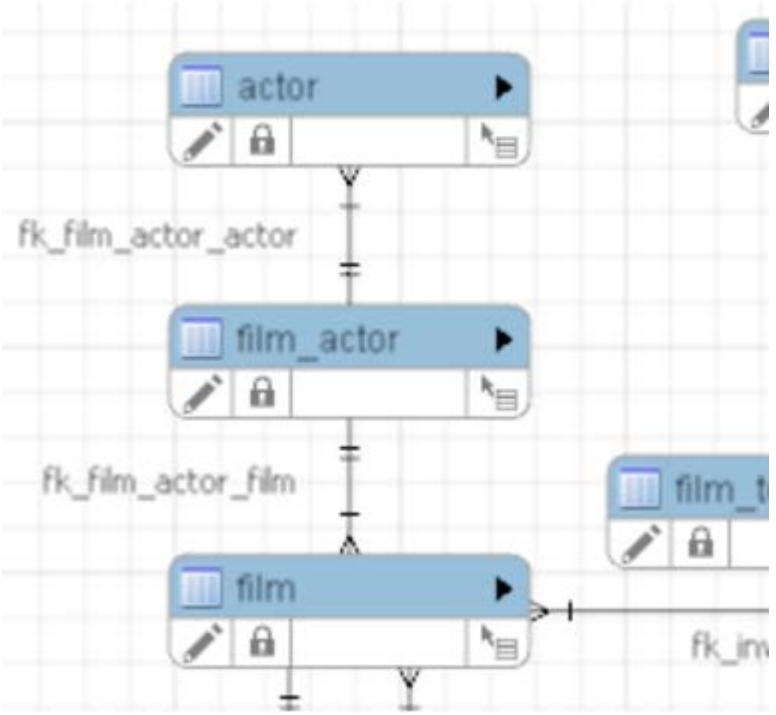
- Many commercial and open source tools exist to help out
- We'll use Spring Batch for a demo here
- It's Java, so it integrates easily with MarkLogic tools
  - MarkLogic Java API
  - mlcp

Demo Architecture

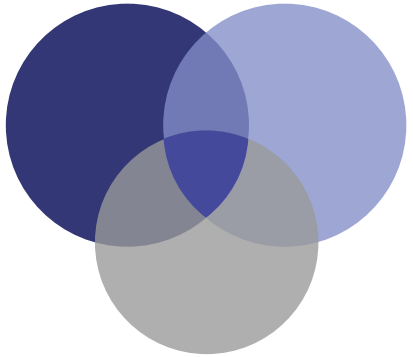


Analyzing Actors

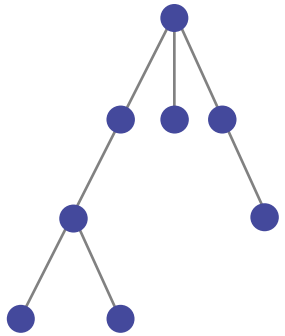
- Actor is an entity
  - So create an “actor” document
- SQL query:
  - `SELECT * FROM actor`



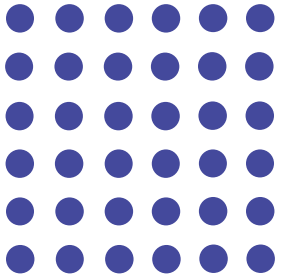
Modeling Actors



COLLECTIONS
actor, sakila



DIRECTORIES
/actor/(uuid).xml



SECURITY
dvd-store-reader/read and
dvd-store-writer/update

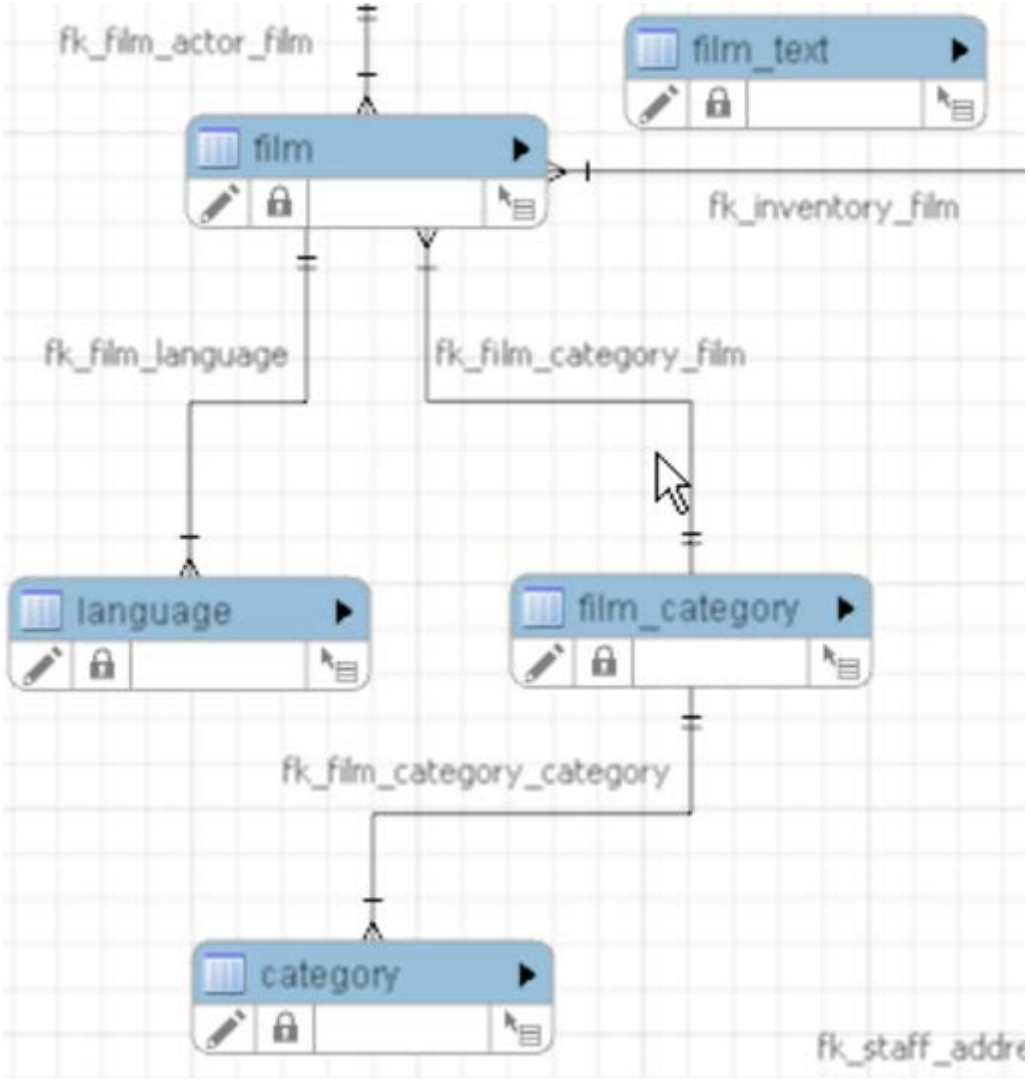
MarkLogic

More About Our Actor Documents

- Each column in a row becomes an element name
  - Simple approach, often a good starting point
- Can analyze it and determine what more to do
- URI and collections allow for future “actor” datasets
  - Example – pull in IMDB set of actors with collections of `actor` and `imdb`
  - Could have completely different structure

Analyzing Films

- A film is an entity
- So create a “film” document
- Tables – `films`, `film_text`, `language`, `film_category`, `category`
  - Conceptually, we feel all of those belong together

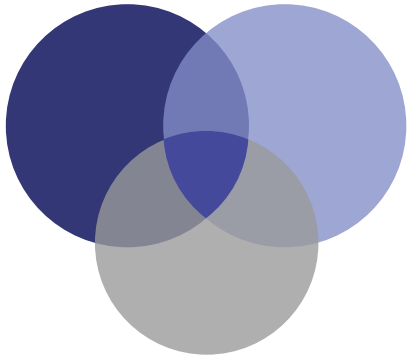


MarkLogic

Migrate Films

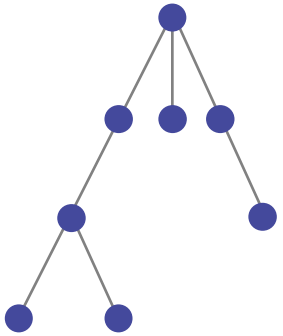
```
SELECT film.*, film_text.description as filmText,
  category.name as category

FROM film

LEFT JOIN film_category ON film.film_id = film_category.film_id

LEFT JOIN category
  ON film_category.category_id = category.category_id

LEFT JOIN film_text ON film.film_id = film_text.film_id

LEFT JOIN language ON film.language_id = language.language_id

ORDER BY film.film_id
```

Modeling Films



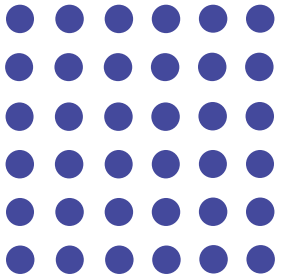
COLLECTIONS

film, sakila



DIRECTORIES

/film/(uuid).xml



SECURITY

dvd-store-reader/read and dvd-store-writer/update

MarkLogic

Benefits of Our Film Document

- We merged 5 tables into a single document
  - Physical model mirrors the conceptual model
- Viewing and updating doesn't require any joins
- More importantly, searching doesn't either
  - Can easily get word query hits on film text
  - Can easily build facets on language and category

MarkLogic

Lookup Data Is Different in MarkLogic

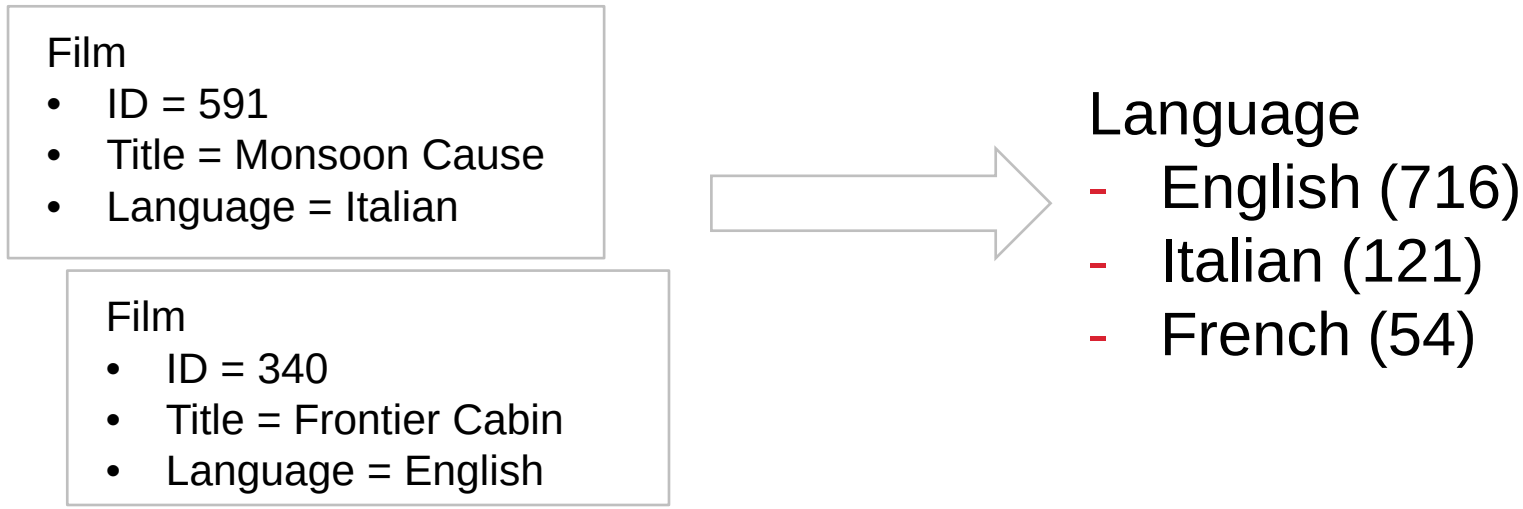
- Common to have lookup tables in relational databases
- How often is this data updated vs how often is it searched?
- Updates are optimized in favor of searches

film_id	title	language_id
133	Chamber Italian	2
285	English Bulworth	1

language_id	name
1	English
2	Italian

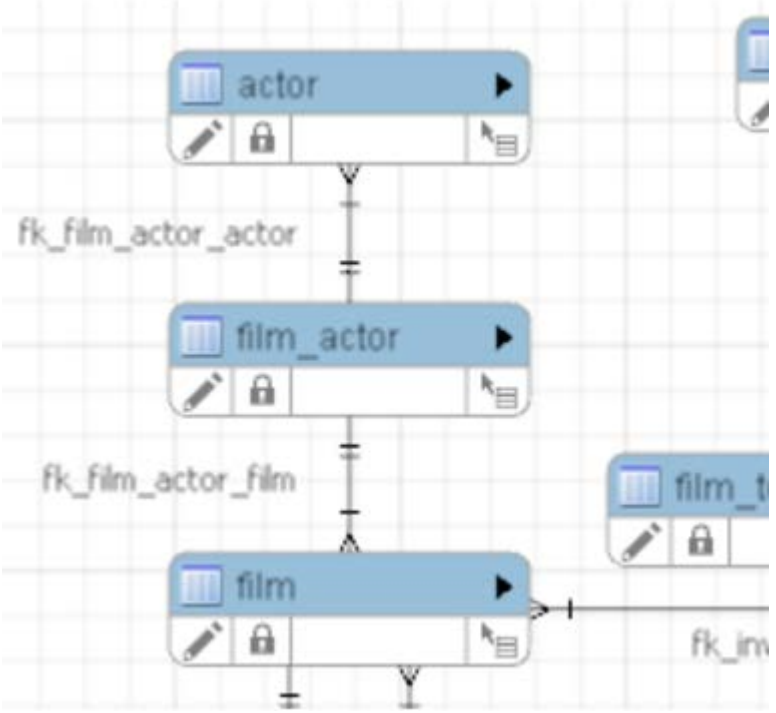
Consider Denormalizing Lookup Data

- Common practice with MarkLogic is to denormalize lookup data
- Favor optimizing search over update
- Much easier to support word queries and facets



Revisiting Actors

- We loaded actors “as-is”
- But what about that many to many?



MarkLogic

Analyzing How Actors Will Be Queried

- We of course want to support searching “for” actors
  - Find me all actor documents with the word “Uma” in them
- But don’t we also want to search for films “by” actor names?
  - Find me all film documents with the word “Uma” in them
  - Or find me all film documents with a cast member with a first name of “Uma”

MarkLogic

Supporting the Query

- We could use semantics to associate documents together:

```
<sem:triple>
  <sem:subject>/film/100.xml</sem:subject>
  <sem:predicate>hasCastMember</sem:predicate>
  <sem:object>/actor/123.xml</sem:object>
</sem:triple>
```

- But we won't get word query hits that way

MarkLogic

Sometimes, It's Best to Denormalize

```
<actors>
   <actor><first-name>Uma</><last-name>Thurman</></>
   <actor><first-name>John</><last-name>Travolta</></>
 </actors>
```

- Now we can support efficient word and element queries
- Can still use semantics for other types of questions
  - Example – linking actors based on films

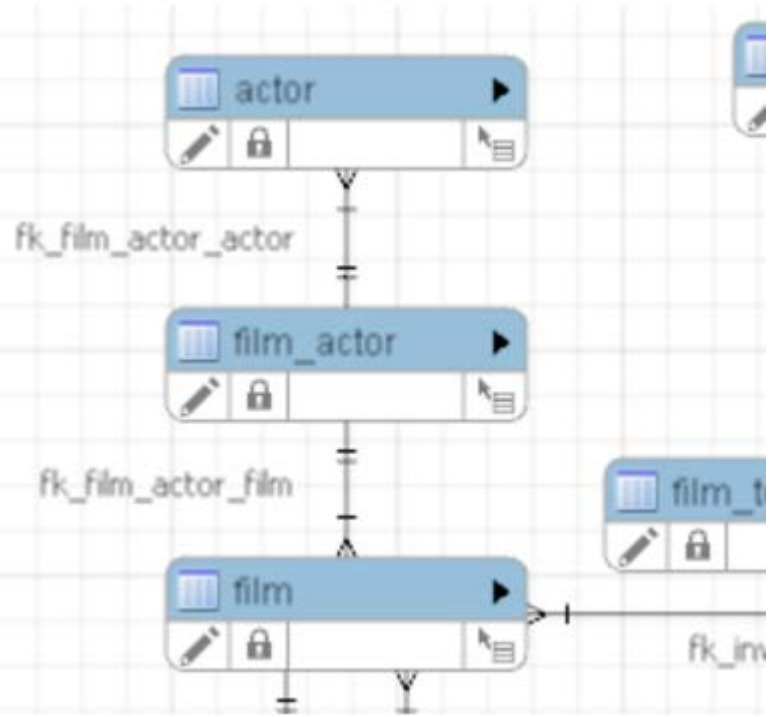
MarkLogic

Thinking Through Denormalizing

- Consider the questions your users want to ask
- Consider how often the data that answers those questions will be updated
- Consider the amount of documents affected when the data is updated
- Consider how efficient a join would be
- Get recommendations
  - MarkLogic Consulting Services
  - StackOverflow

Migrate Films With Actors

```
SELECT film.*, film_text.description as filmText,
  category.name as category, actor.actor_id as "actor/id",
  actor.first_name as "actor/firstName",
  actor.last_name as "actor/lastName"

FROM film LEFT JOIN film_category
  ON film.film_id = film_category.film_id

LEFT JOIN category
  ON film_category.category_id = category.category_id

LEFT JOIN film_actor ON film.film_id = film_actor.film_id

LEFT JOIN actor ON film_actor.actor_id = actor.actor_id

LEFT JOIN film_text ON film.film_id = film_text.film_id

ORDER BY film.film_id
```



MarkLogic

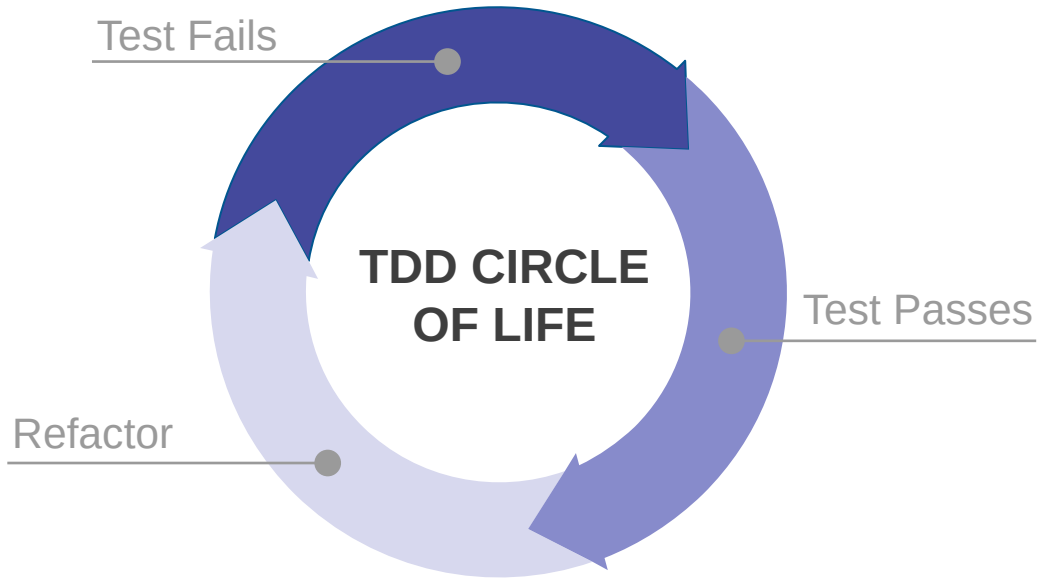
Summary of Migrations

- Actors: load as-is, one row per document
- Films: merge multiple tables and rows into one document; denormalize lookup data
- Films and actors: denormalize some actor data onto film documents

- Your migrations will most likely involve all of these techniques

Migrations in the Real World

- Focus on migration as soon as possible
- Favor Agile development
  - MarkLogic simplifies loading data
  - Migrate some data, test, iterate



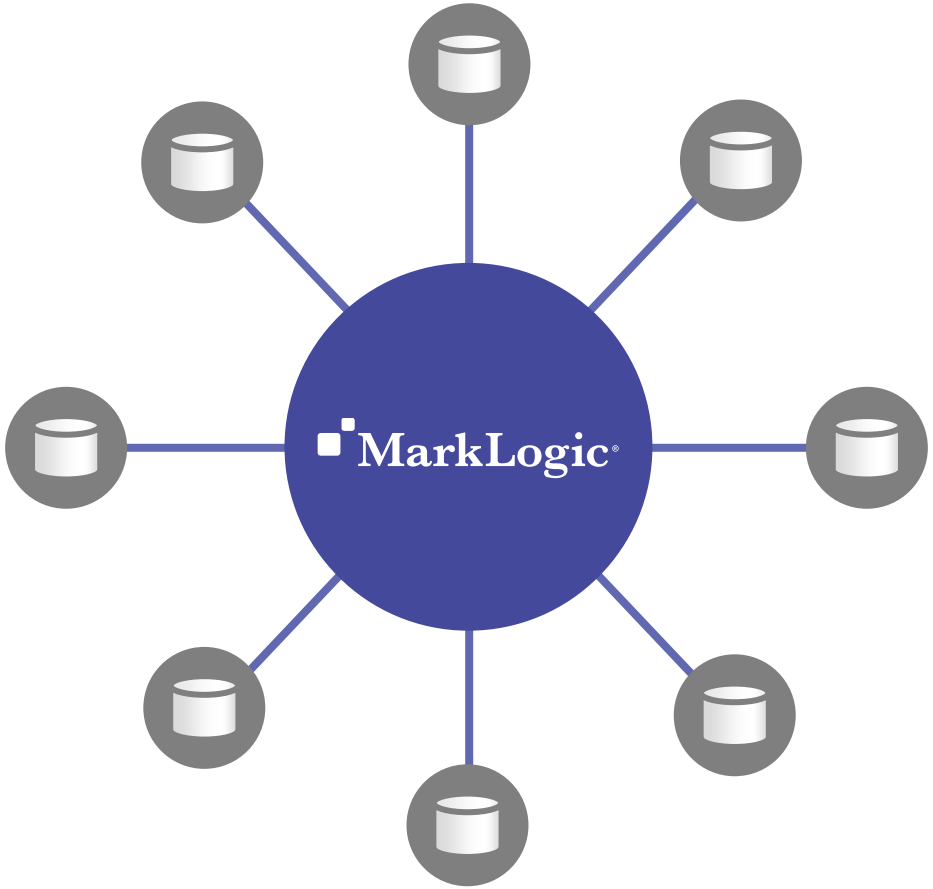
MarkLogic

Export

- Problem
  - BI tools are geared towards relational databases
  - How do we integrate those tools with MarkLogic?
- Solution
  - Use a MarkLogic ODBC server to expose documents as rows and support SQL
- Demo

Summary

- Goal: unified, actionable 360º view of data
- Must copy data from tables to documents
  - Analyze and identify entities
  - Design a data model for those entities
    - URIs, collections, security
  - Iterate, test, iterate, test
- Realize the benefits of documents in an operational, transactional Enterprise NoSQL database



Q&A