MarkLogic

RDBMSを超えて：MarkLogicでリレーショナルデータを活用する

マークロジック株式会社
シニアソリューションアーキテクト
能仁 信亮

MarkLogic

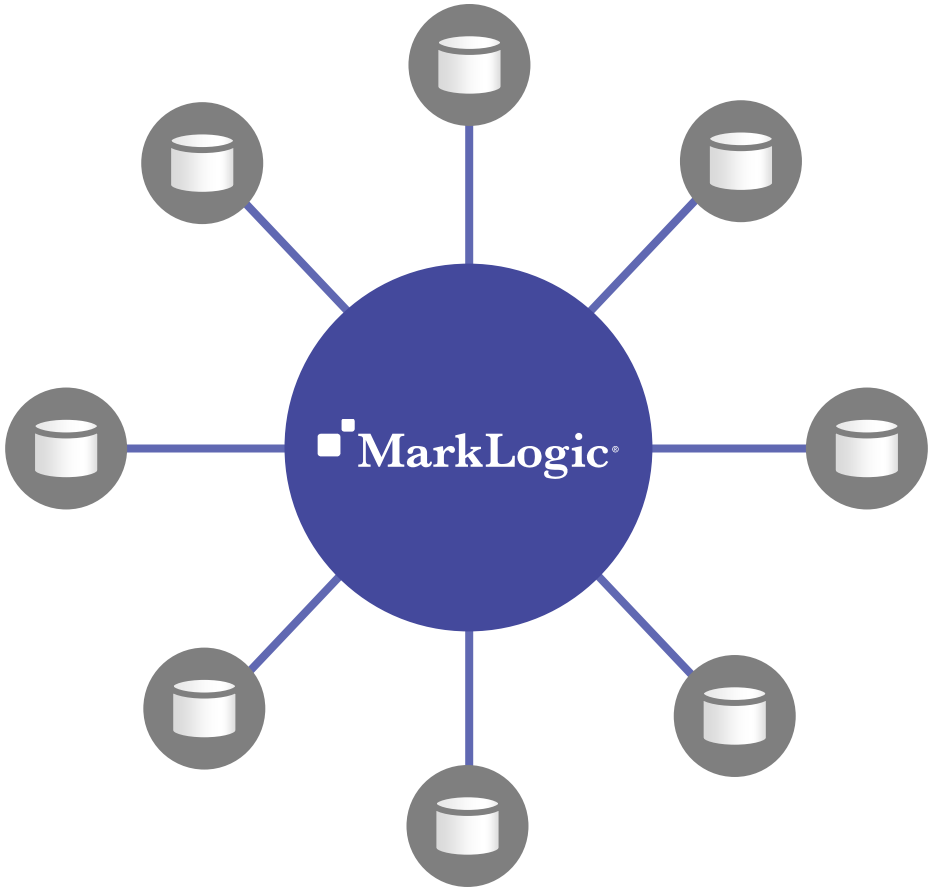
アジェンダ

- 自己紹介
- 取り組むべき課題の再確認: リレーショナル・データを MarkLogic に取り込む
- デモ : 実装例をもとに考え方やTipsをご紹介

MarkLogic

自己紹介

- 能仁 信亮 (のうにん しんりょう)
- ソリューション・アーキテクト
- MarkLogic入社前は、Hadoop, NoSQLとRDBMSを組み合わせたシステム構築の提案、導入支援を行っていました
- Twitter : @snonin



望ましいソリューション

データをより上手く、早く、少ないコストで統合できるデータベース

MarkLogic

MarkLogicという選択

オペレーショナルでトランザクショナルな 基幹業務対応エンタープライズNoSQLデータベース

簡単にデータを取り込める

柔軟なデータモデル

- データをそのまま取込む (ETL不要)
- 構造化、非構造化データの混在
- データとメタデータを一緒に保管
- データやデータ構造の変更に対応

簡単にデータを出せる

ユニバーサルインデックスで何でも検索

- インデックスを一度付けることで、何度でもクエリ可能
- 非常に高速でリアルタイム
- JSON、XML、テキスト、地理空間情報、セマンティックトリプルをすべてひとつのデータベースでクエリ可能

100% 信頼できる

基幹業務に対応できるエンタープライズ仕様

- データの信頼性とトランザクション (100% ACID 準拠)
- 標準仕様の自動フェイルオーバー、レプリケーション、バックアップ/リカバリー機能
- 基幹業務に対応したエンタープライズグレードのセキュリティとコモンクライテリア認証

MarkLogic

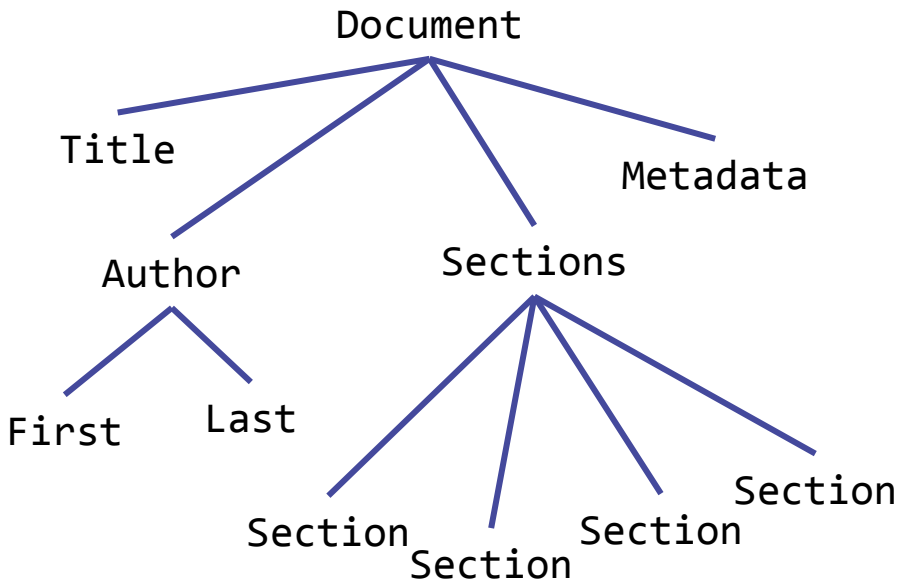
データを簡単に取り込める

- 構造化データと非構造化データ – 当然可能です!
- 変化し続けるデータとデータ構造に対応する – もちろん対応できます!
- データをそのまま(as-is)取り込む – これについては、後ほどお話します

MarkLogic

テーブル とドキュメント

- リレーショナル = 2次元のテーブル
- ドキュメント = ネスト構造・階層構造
- テーブルをそのまま(as-is)ロード可能
  1行が1ドキュメントになる
  - しかし、ドキュメントの利点は失われている



MarkLogic

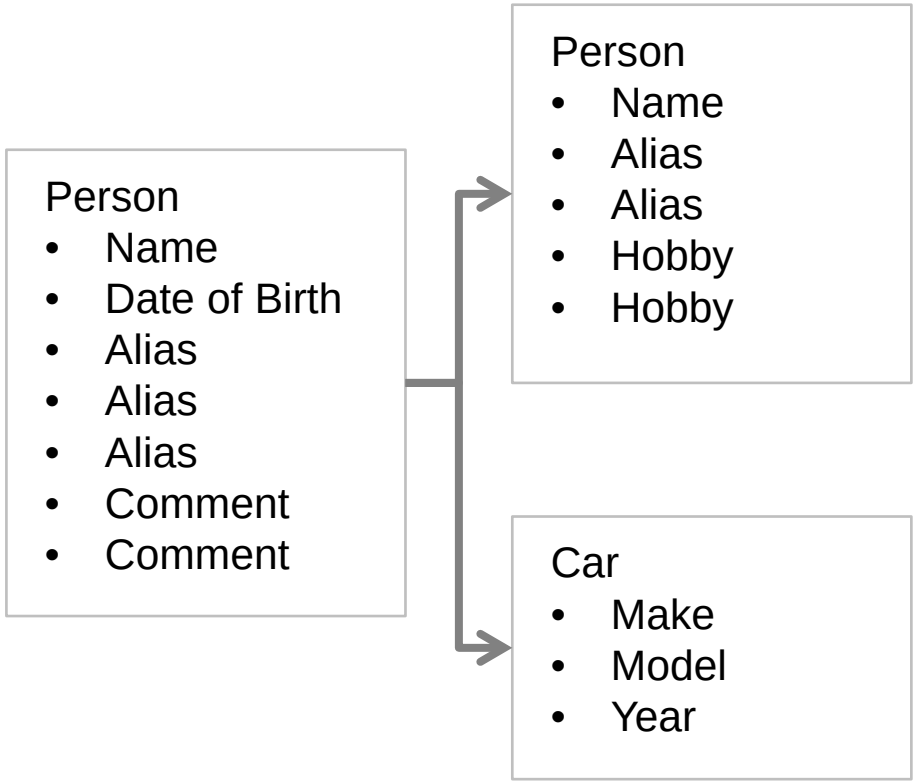
なぜドキュメントなのか?

- 業務上自然なデータの分類とRDBMSのテーブルは必ずしも一致するわけではない
- RDBMSでは、概念モデルのエンティティを1:多のリレーションをもつ個別のテーブルに分割しなければならない場合がある
  - 有効な場合もあるが、そうではないこともある
  - これ以外の選択肢がないことが問題

MarkLogic

ドキュメントは概念モデルを反映させることができる

- 業務モデルのなかで、階層をもったエンティティをモデリング
- 物理モデルでも、そのモデルをそのまま利用できる
- リレーションも、利用可能
- モデルを作成する際に選択可能



MarkLogic

MarkLogicのドキュメントはさらに柔軟

- 単一のテーブルに制限されずデータベース全体に対してクエリを発行可能
- ドキュメントを複数のCollectionに紐付けて分類可能
- 単一のトランザクションで、多数のドキュメントをinsert/update/delete可能
- 様々なデータセットを同じ形式にモデリングすることなく、データを取り込むことが可能
  - データ統合において、キーとなる特徴

MarkLogic

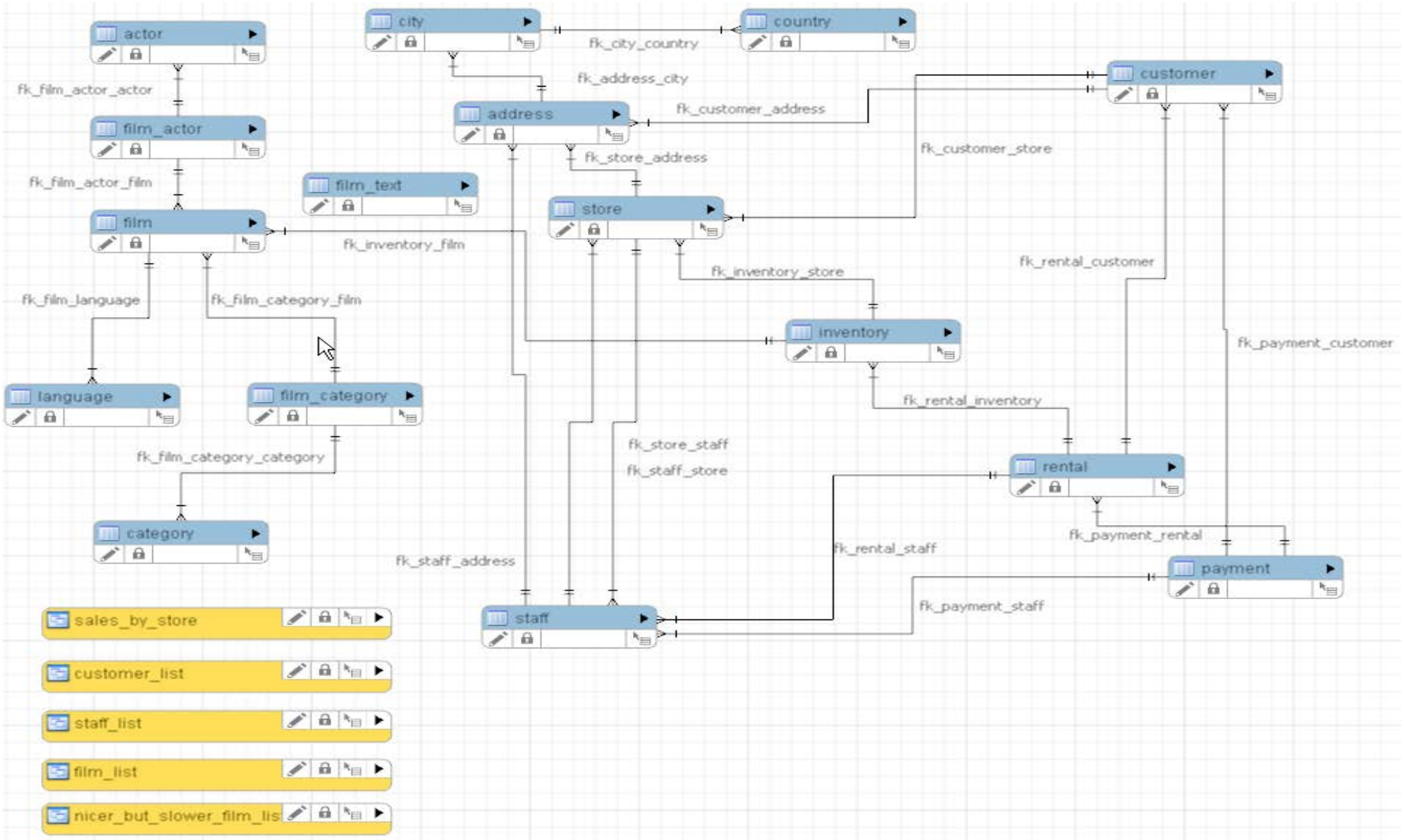
取り組むべき課題

- 統合され活用可能なデータの360度ビューを手に入れたい
- データ・サイロを統合する必要がある
- データ・サイロの大半は、RDBMSからなる
- MarkLogic はドキュメントを保持する
- ここまでMarkLogic とドキュメントの利点を見てきた
- でも、実際にどのようにRDBMSのテーブルをドキュメントに変換すればよいのだろうか?

MarkLogic

サンプル・データ

- MySQL Sakila データセット
  - https://dev.mysql.com/doc/sakila/en/
  - DVD レンタルショップ を想定したデータセット
  - MySQL特有の機能や新機能をテストするために利用されている
- テーブルからドキュメントへの変換をテストするのに最適



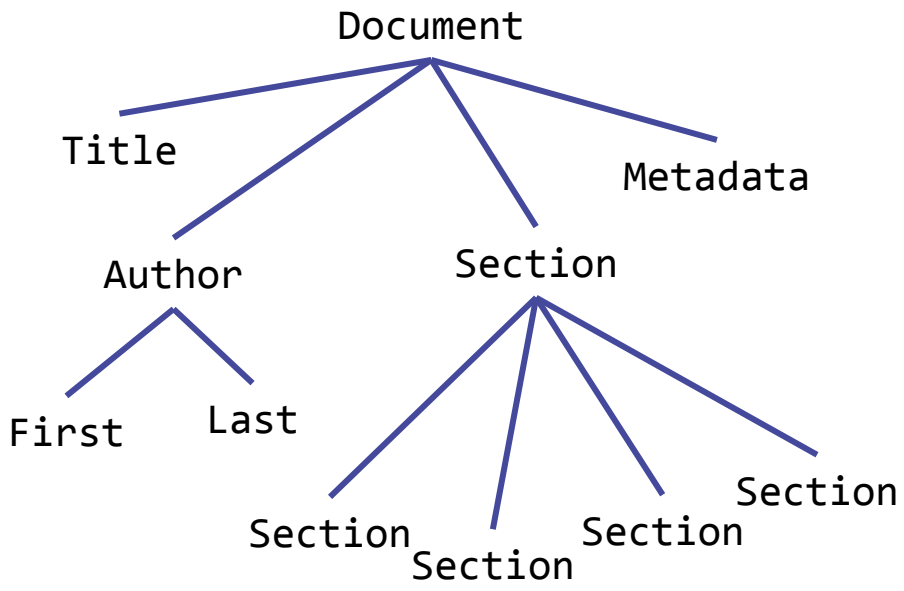
MarkLogic

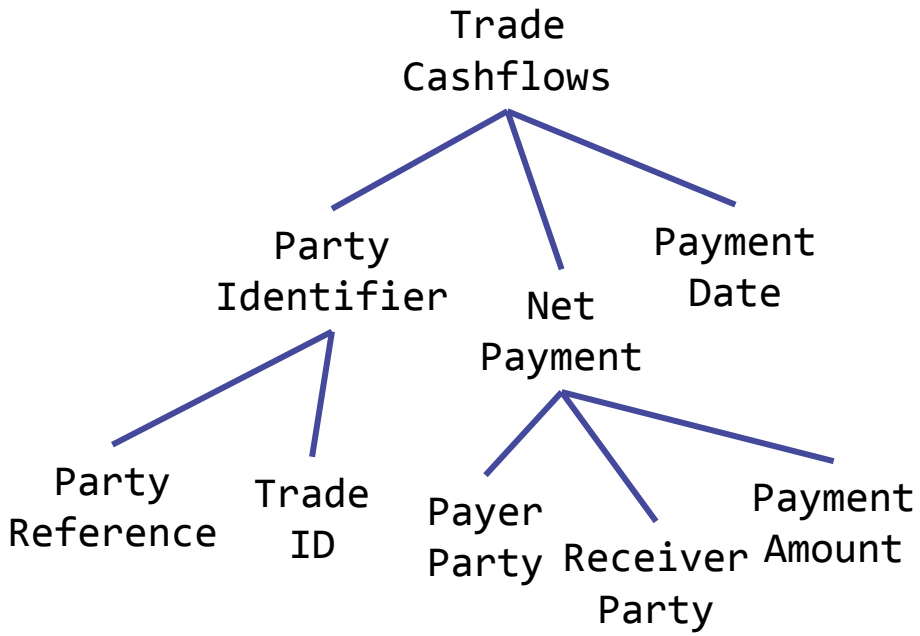
テーブルからドキュメントへ

- MarkLogicのデータモデリングの選択肢を理解する必要がある
- エンティティを特定する必要がある
  - システムで利用される、主要な名詞はなんだろうか?
  - 典型的にはそれらが、ドキュメントの種別になる

MarkLogic データモデリング 入門

- MarkLogic はドキュメント指向のデータベース
  - 階層型のデータモデルにより、任意の構造のデータをサポート
  - 圧縮された木構造を保持





MarkLogic

MarkLogic データモデリング 入門

- ドキュメントは URI をもつ
  - 任意の文字列を利用可能。一意でさえあればよい
  - 主キーに似ているが、主キーがテーブルをスコープとしているのに対して、データベース全体をスコープとする

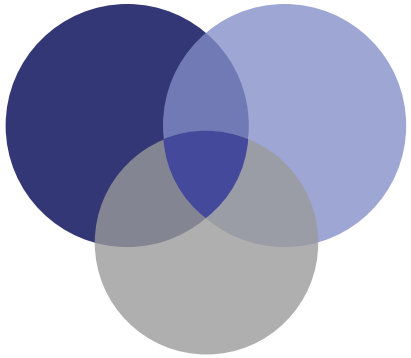
MarkLogic

MarkLogic データモデリング 入門

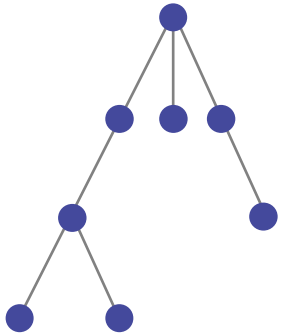
- どのようなURIを設定すべきか
  - UUID (RFC 4122)を利用：sem:uuid-string()
    - 名前空間に関する問題を簡単に回避できる
      例) IDが“14”のデータが複数のDBからくる
  - ソースシステムの情報等を組み合わせて一意のURIを生成する
    例) /film/<ソースシステムコード>/<ソースシステムの主キー>.xml
    - ソースシステムからの更新がURIをベースに行えるため簡単
    - URI生成の規則を厳密に管理する必要がある
- URIはパスのように見えることが多いが、かならずしもパスである必要はない
  - 例：“/film/123.xml” もしくは “123.xml” もしくは単に”123”でもよい

MarkLogic

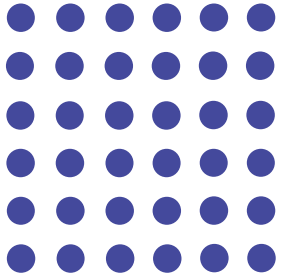
MarkLogic データモデリング 入門



COLLECTIONS
集合をベースに
n:m の関係性



DIRECTORIES
排他的かつ階層的で、ファイルシステムに類似。URIをベースとする



SECURITY
ロールとドキュメントレベルのパーミッション

MarkLogic

エンティティを特定する

- ユーザーが会話でとりあげたり、問合せを行う主要な名詞はなんだろうか?
- DVD レンタルショップでは:
  - 顧客(customer)の視点では: 特定の役者(actor)がでている映画(film)を探したい
  - 顧客(customer)の視点では: 特定の映画(film)を取り扱っている店舗(store)を探したい
  - スタッフ(staff member)の視点では: 期限を超過したレンタル(rental)を見つけたい
- 名詞– 映画(film), 役者(actor), 店舗(store), レンタル(rental), 顧客(customers), スタッフ(staff member)
- これらがドキュメントのコレクションになる

MarkLogic

では、データ移行をはじめていきましょう

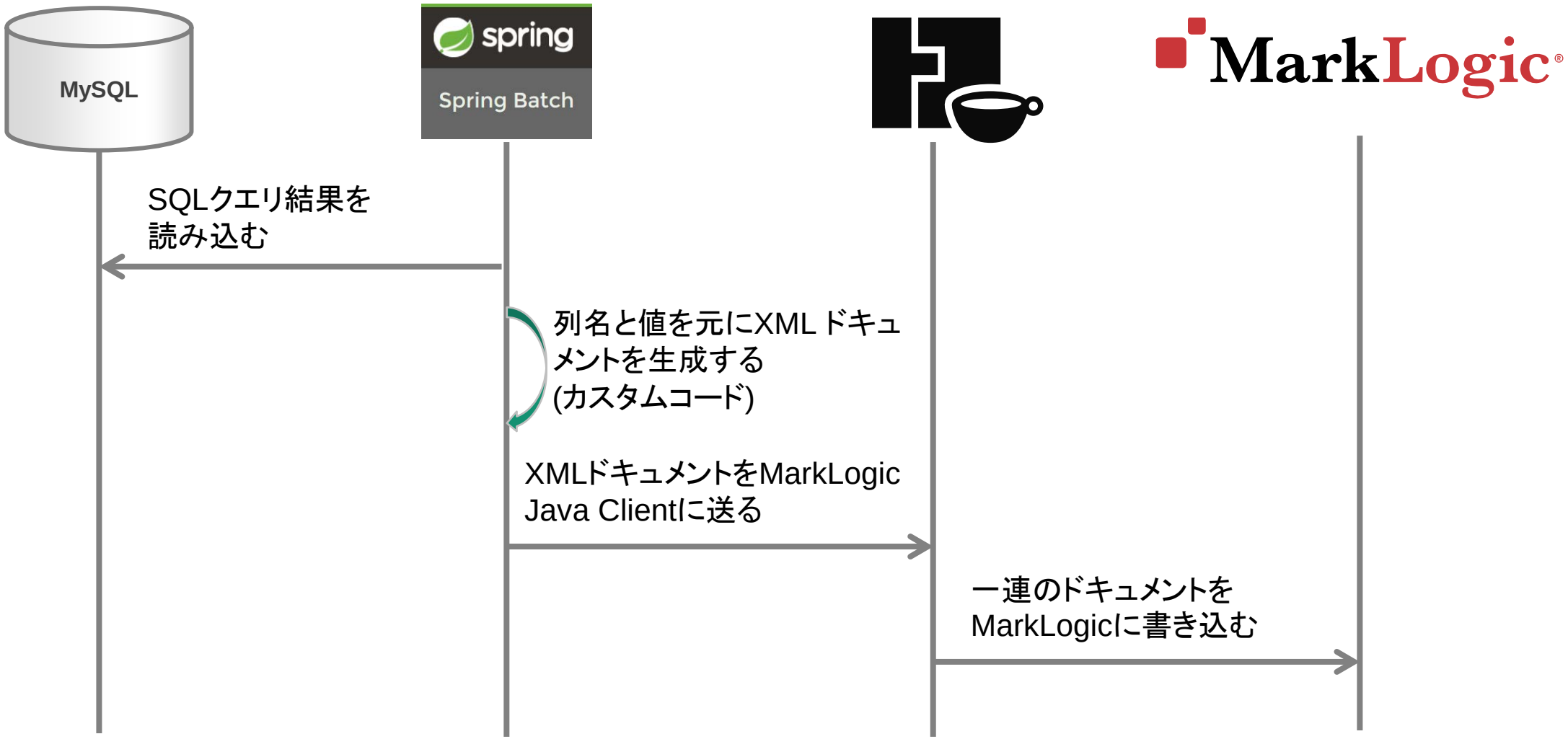
- デモは slush generator を利用して作成
  - https://github.com/rjrudin/slush-marklogic-spring-boot
  - Angular / Spring Boot / MarkLogic
- 中間層のSpring Boot でSpring Batch を利用してデータを移行
- まずはactorsのデータからはじめて、filmsを次に移行します

MarkLogic

一括移行のためのSpring Batch

- データのバッチ移行に役立つ、様々な商用やオープンソースのツールが存在
- このデモでは、Spring Batchを利用
- Javaなので、MarkLogicのツール群と簡単に連係可能
  - MarkLogic Java API
  - Content Pump

デモ アーキテクチャ



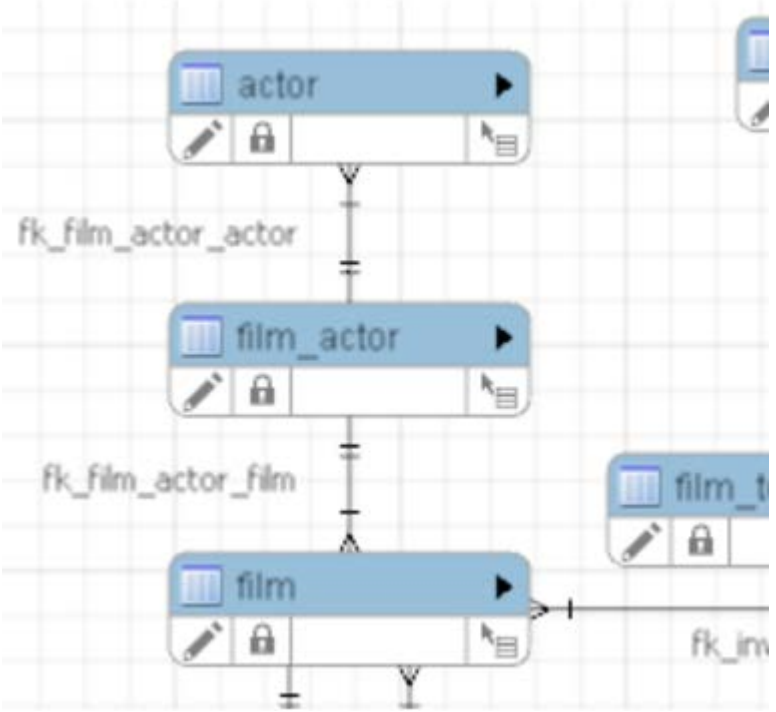
MarkLogic

差分更新でも、いままでのテクニックを再利用可能

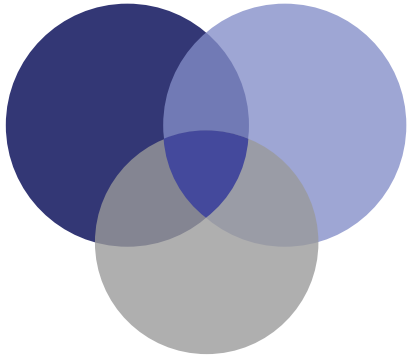
- RDBMSから、差分でデータを抽出する方式の例
  - 最終更新日時 を格納する列を用意し、その列を利用
    - 条件で前回の抽出日付以降のものを抽出
  - Snapshotの利用 (メインフレームなどで有効)
    - ある時刻での全量データ(Snapshot)を抽出し、前回のSnapshotと比較。差分のみロード
  - Change Data Captureの利用
    - RDBMSのトランザクション・ログから差分データを抽出
      (IBM InfoSphere CDC, Oracle GoldenGate…)
  - RDBMSのトリガーを利用して差分データを抽出

役者(actor)を分析する

- actorはエンティティ
  - “actor”ドキュメントを作成する
- SQLクエリ:
  - SELECT * FROM actor

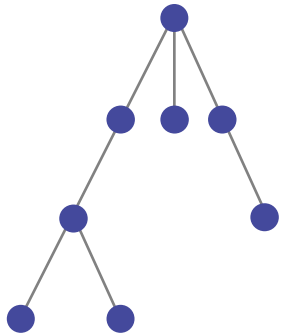


Actor をモデリングする



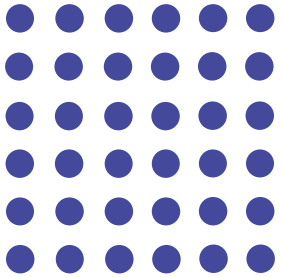
COLLECTIONS

“actor”, “sakila”



DIRECTORIES

“/actor/(uuid).xml”



SECURITY

“dvd-store-reader/read”

“dvd-store-writer/update”

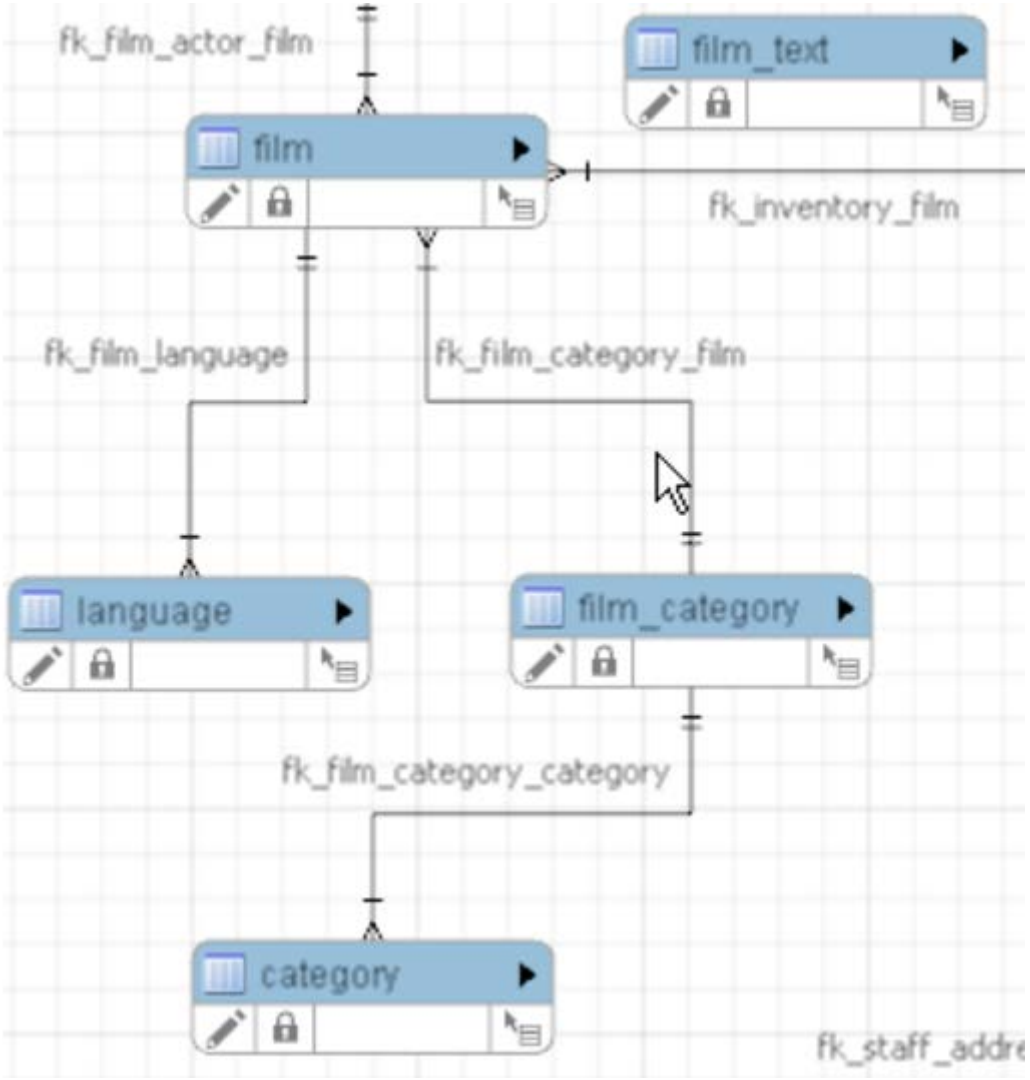
MarkLogic

Actor ドキュメントに関して

- 行の各列を、要素名に
  - シンプルなアプローチ。多くの場合、よいスタート地点になる
- 分析を行って、さらに工夫ができるかを考える
- URI と collection は将来の “actor”データセットを考える際に有用
  - 例 – IMDB から役者のデータを取り込み、collectionとして “actor” と“imdb”を設定
  - 完全に異なる構造を持たせることも可能

映画(film)を分析する

- filmはエンティティ
- “film”ドキュメントを作成する
- テーブル – film, film_text, language, film_category, category
  - 概念的には、これらをひとつにまとめたい

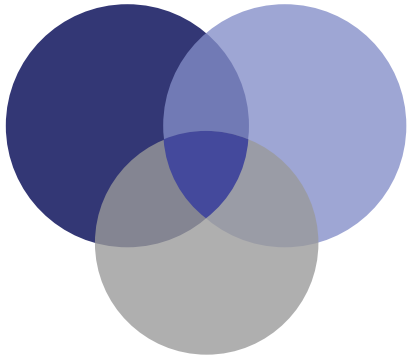


MarkLogic

film を移行する

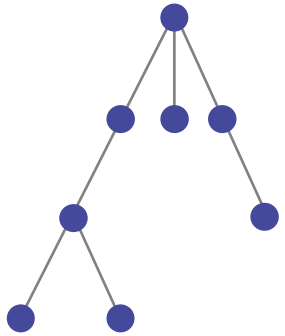
```
SELECT film.*, film_text.description as filmText, category.name as category, …
FROM film
LEFT JOIN film_category ON film.film_id = film_category.film_id
LEFT JOIN category ON film_category.category_id = category.category_id
LEFT JOIN film_text ON film.film_id = film_text.film_id
LEFT JOIN language ON film.language_id = language.language_id
ORDER BY film.film_id
```

filmをモデリングする



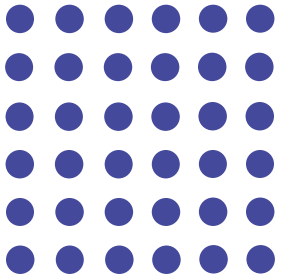
COLLECTIONS

“film”, “sakila”



DIRECTORIES

“/film/(uuid).xml”



SECURITY

“dvd-store-reader/read”

“dvd-store-writer/update”

MarkLogic

作成したfilmドキュメントの利点

- 5つのテーブルを1つのドキュメントに統合
  - 物理モデルが概念モデルを反映している
- 参照や更新時にjoinを必要としない
- さらに重要なのは、サーチの際に:
  - film textでワードクエリーが簡単に実行できる
  - rating や category でファセットを簡単に設定できる

MarkLogic

MarkLogicにおける参照データの取り扱い

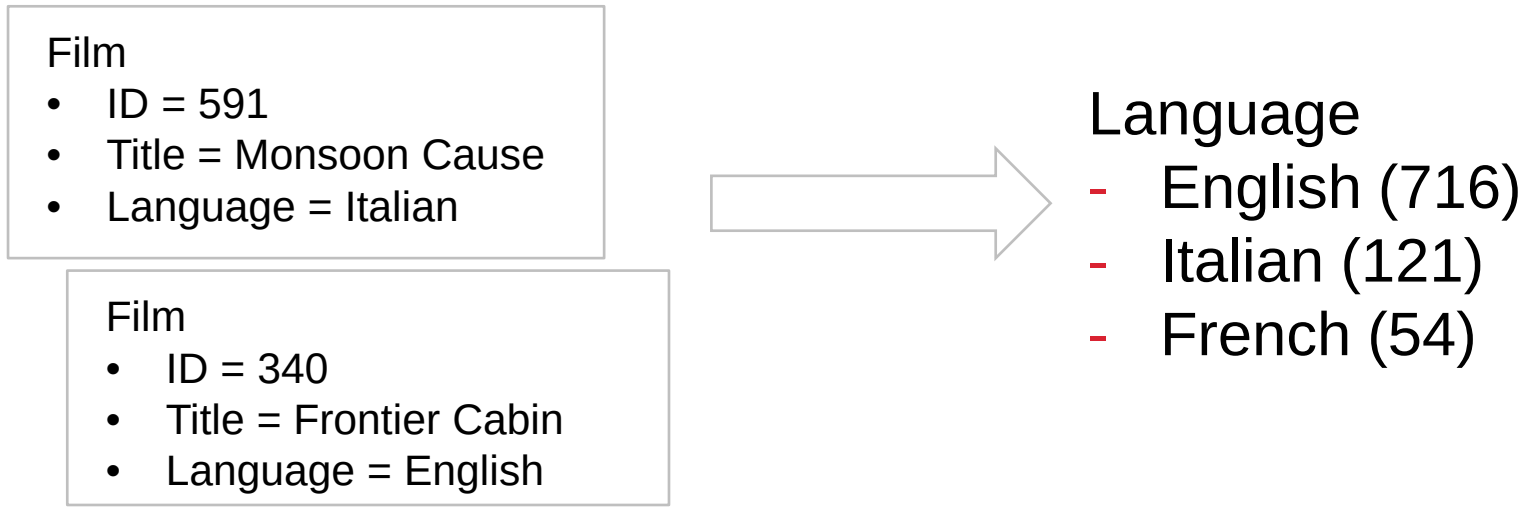
- リレーショナルデータベースでは、参照テーブル(lookup table)を利用することが一般的
- 検索処理よりも、更新処理に最適化されている
- データの更新の頻度 vs データの検索の頻度は?

film_id	title	language_id
133	Chamber Italian	2
285	English Bulworth	1

language_id	name
1	English
2	Italian

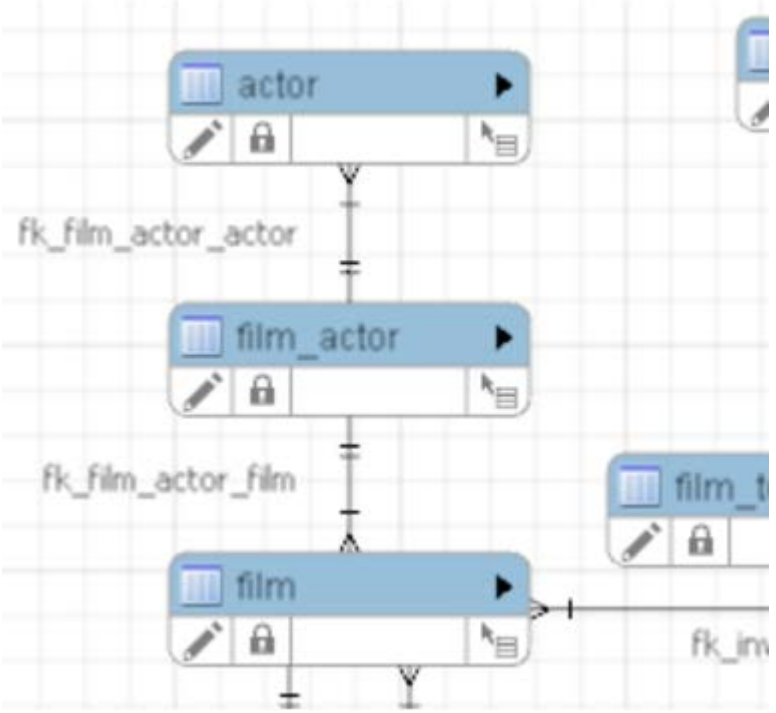
参照データを非正規化する

- MarkLogicによる実装では、参照データを非正規化する手法がよく行われる
- 更新処理よりも、検索処理を優先
- ワードクエリや、ファセットなどの実装が、かなり簡単になる



Actorを見直してみる

- Actorを「そのまま (as-is)」ロードしていた
- しかし、多対多の関連性はどうだろうか?



MarkLogic

Actor がどのように検索されるかを分析する

- もちろん、役者(actor)の検索はサポートしたい
  - “Uma”という単語が含まれるactorドキュメントを探したい
- しかし、映画(film)を役者(actor)の名前でも探したい
  - “Uma”という単語が含まれるfilmドキュメントを探したい
  - もしくは、名前が“Uma”である出演者がいるfilmドキュメントを探したい

MarkLogic

actor データの一部を非正規化する

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

- 効率的なワードクエリ、エレメントクエリをサポートできる
- その他の、問合せに対しては、セマンティックを使うこともできる
  - オスカー主演女優賞を受賞した出演者がいる映画を探したい

MarkLogic

非正規化の考慮事項

- 利用者が、知りたい質問を考える
- その質問に対する答えを与えるデータが更新される頻度を検討する
- データが更新された際に影響するドキュメントの量を検討する
- Joinがどの程度効率的かを検討する
- SPARQLを使って質問に答えるために、ドキュメントに対してトリプルによって付加情報与えることを検討する
- 推奨される方法を聞く
  - MarkLogic Professional Service
  - stackoverflow

filmとactorを移行する

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

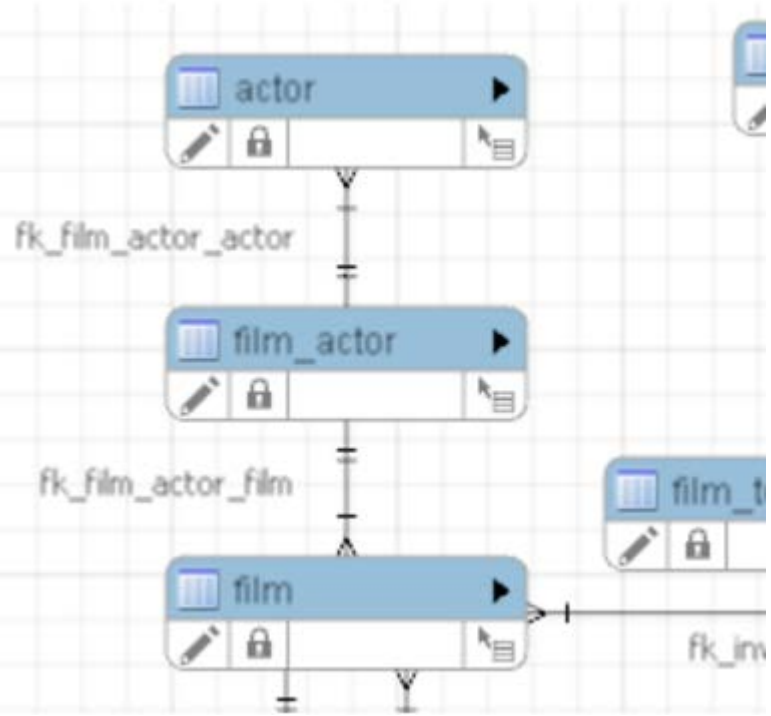
LEFT JOIN category ON film_category.category_id = category.category_id

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

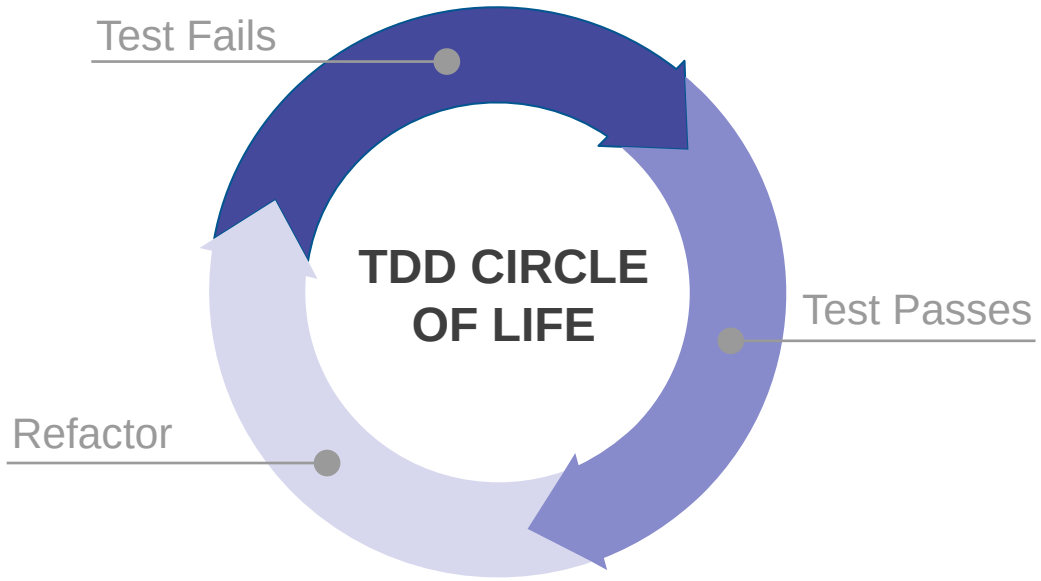
データ移行のまとめ

- actor: そのまま(as-is)ロードする。1行が1ドキュメント
- film: 複数のテーブルや行を1ドキュメントにマージする。参照データを非正規化する
- film と actor: actorデータの一部を非正規化してfilmドキュメントに含める

- 移行の際には、これらすべてのテクニックが必要となる場合が多い

実プロジェクトでの移行

- まずは可能な限り、移行に集中する
- アジャイル開発が好ましい
  - MarkLogic はデータのロードをシンプルにする
  - 一部データを移行しテスト。これを繰り返す



MarkLogic

Export

- 問題
  - BIツールは、RDBMSに最適化されている
  - それらのツールをどのようにMarkLogicと組み合わせるか?
- 解決策
  - MarkLogic ODBC サーバーを利用してドキュメントを行として公開し、SQLをサポートする

MarkLogic

まとめ

- 目的: 統合され活用可能なデータの360度ビュー
- データをテーブルからドキュメントへコピーする必要がある
  - エンティティの分析と特定
  - それらのエンティティのデータモデルの設計
    - URI, collection, セキュリティ
  - 反復、テスト、反復、テスト
- オペレーショナルかつトランザクショナルなEnterprise NoSQL Databaseでのドキュメントの利点を認識