Progress® MarkLogic®

Documents

Key Concepts in this Unit

- Supported document:
  - Formats
  - Reference
  - Organization
- Recommended document structure.
- Bulk load/ingest.
- MVCC and Modifying data.
- Bulk update.

Concept: Document

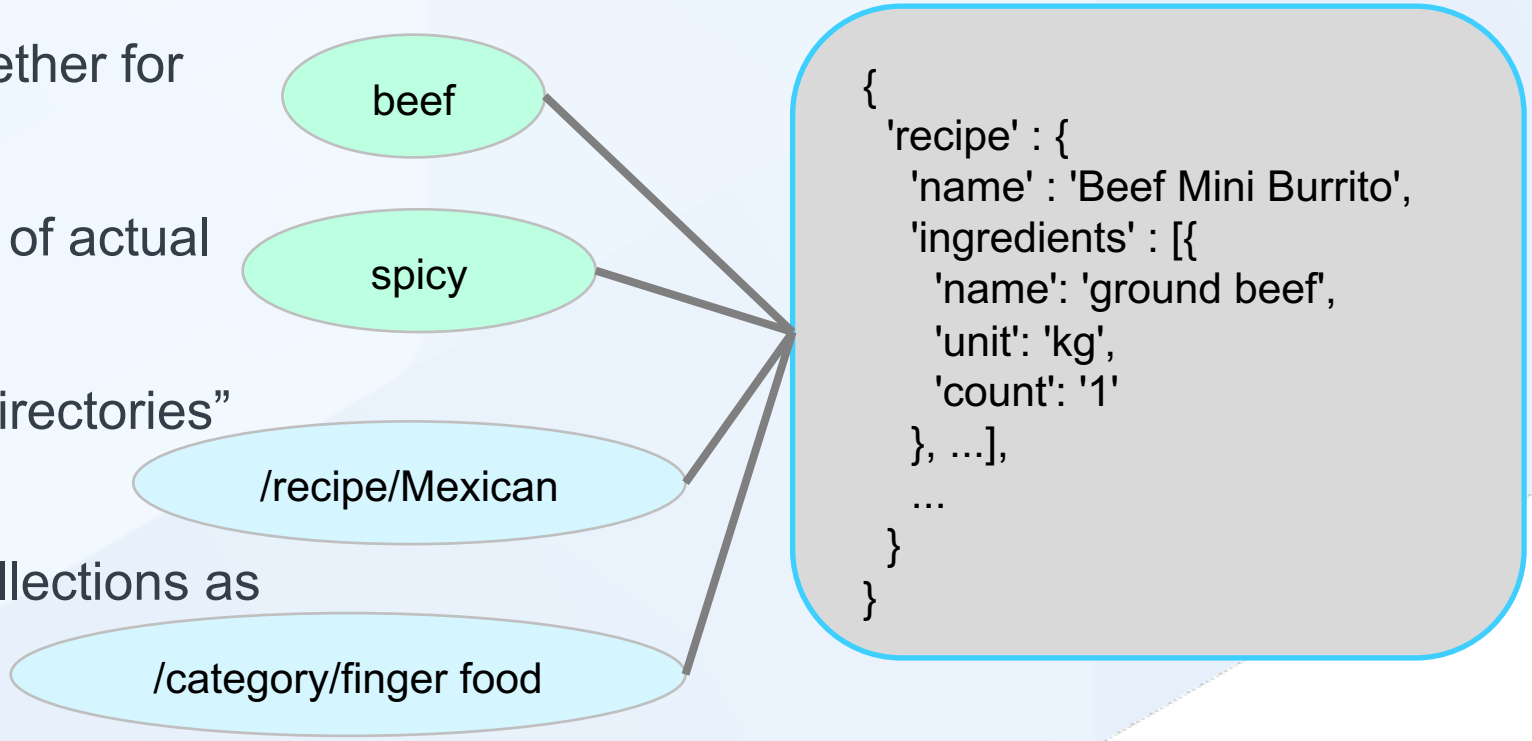
- MarkLogic supports different types of document content:
  - Structured: XMLDocument, JSONDocument
  - Unstructured: TextDocument
  - Graph: RDF Triples represented in XML or JSON.
  - Binaries: BinaryDocument
- Documents can be loaded “as is”
  - No defined schema or structure required before loading.
  - Indexes and Templates may be used to enforce structure and datatype after loading.

Concept: Uniform Resource Identifier (URI)

- A document is assigned a URI when loaded.
- A URI:
  - Is a **string** that uniquely identifies a document stored in MarkLogic, for example, `**/directory/filename.json**` or `**/employee/142.json**'.
  - Performs the similar function as a Primary Key in a RDBMS, like `**Row_ID**`.
- A "directory" is used to organize documents:
  - A document may only belong to one directory.
  - No actual directory structure created inside a forest.

Concept: Collections

- Strings that are used to “tag” documents.
- Used for grouping documents together for search or modification operations.
- The Collection values are not part of actual content, for example, meta-data.
- Collections may also opt to use “directories” as part of the text.
- A document can have as many collections as desired.



Concept: Envelope Pattern

- Allows for information compartmentalization.
- The Envelope is made up of sections:
  - Headers – Capture additional metadata.
  - Triples – Store Embedded Triples that can store relationships.
  - Instance – Stores “Entity” information used by data services.

```
{
  'envelope' : {
    'headers' : {
      'update-history' : [{
        'date' : '2022-01-23',
        'username': 'dan'
      }]
    },
    'triples' : [],
    'instance': {
      'EntityName' : {
        'id' : '1278821',
        'type' : 'record',
        ...
      }
    }
}
}
```

Concept: Triples

- Triples are composed of 3 parts:
  - Subject
  - Predicate
  - Object
- Can be “embedded” as part of a document or “managed” by MarkLogic.

Embedded Triple

```
{
  'envelope' : {
    'triples' : [{
      'triple' : {
        'subject' : 'http://my.ns/id-1234',
        'predicate' : 'http://my.ns#property',
        'object' : 'value'
      }
    }...
```

Managed Triple

```
<sem:triples xmlns:sem="http://marklogic.com/semantics">
  <sem:triple>
    <sem:subject>http://my.ns/id-1234</sem:subject>
    <sem:predicate>http://my.ns#property</sem:predicate>
    <sem:object datatype="http://www.w3.org/2001/XMLSchema#string" xml:lang="en">value</sem:object>
  </sem:triple>
</sem:triples>
```

Relational Records converted to Documents

- Data stored in RDBMS tables is commonly Normalized.
  - Multiple records in multiple, related tables.
- Related records are often denormalized as they're loaded into MarkLogic and converted into individual documents.

Order table	
order_id	**order_date**
10072	2017-01-14

Order Item table			
Item_id	**order_id**	**product**	**quantity**
23122312	**10072**	SpeedPro Ultimate	1
23857292	**10072**	Mens Racer Helmet	1

URI → /orders/10072.json

```
{
  "orderNum": "10072",
  "orderDate": "2017-01-14",
  "items": [{
    "item": {
      "product": "SpeedPro Ultimate",
      "price": 999,
      "quantity": 1
    }
  }, {
    "item": {
      "product": "Mens Racer Helmet",
      "price": 95,
      "quantity": 1
    }
  } ...
```

Tool: MarkLogic Content Pump – MLCP

- A Java-based tool that supports:
  - Load of multiple documents in a directory and its sub-directories.
  - Conversion of each row of delimited text files into documents.
  - Conversion of RDF triples into equivalent XML files.
  - Generate custom URIs and content transformation.
  - Throttle of threads and batch size.
- Can be used to extract documents from a MarkLogic database in one cluster to a MarkLogic database to another, for example, Dev to Test/UAT.

MLCP Execution

- Command line or as a Custom ml-gradle Task:
  - Command Line

```
$ home/cent/mlcp/bin/mlcp.sh import -mode local -host localhost -port 8100 -username admin -password admin -
output_collections “star-wars,characters” -input_file_path /home/cent/Desktop/fundamentals/data/star-
wars/characters/ -output_uri_replace ".*characters,'/characters'"
```

  - Custom ml-gradle Task - Defined in the build.gradle file.

```
build.gradle ×
build.gradle
49
50  /**
51   * Load the characters folder
52   */
53  task importCharacters(type: com.marklogic.gradle.task.MlcpTask) {
54      classpath = configurations.mlcp
55      command = "IMPORT"
56    port = mlAppConfig.restPort
57      output_collections = "star-wars,characters"
58    input_file_path = "/home/cent/Desktop/fundamentals/data/star-wars/characters/"
59      output_uri_replace = ".*characters,'/characters'"
60  }
61
62  /**
63   * Load the images folder
PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE
[cent@centos7-vm star-wars]$ ./gradlew importCharacters
```

Other Supported Data Load Tools

- In addition to MLCP, data can be loaded into a MarkLogic database with other tools:
  - Java Data Movement SDK
  - NodeJS MarkLogic Client
  - Apache NiFi Processors
  - MuleSoft Connector
  - Pega Connector
  - Hadoop Connector
  - Kafka Connector, etc.

Progress MarkLogic

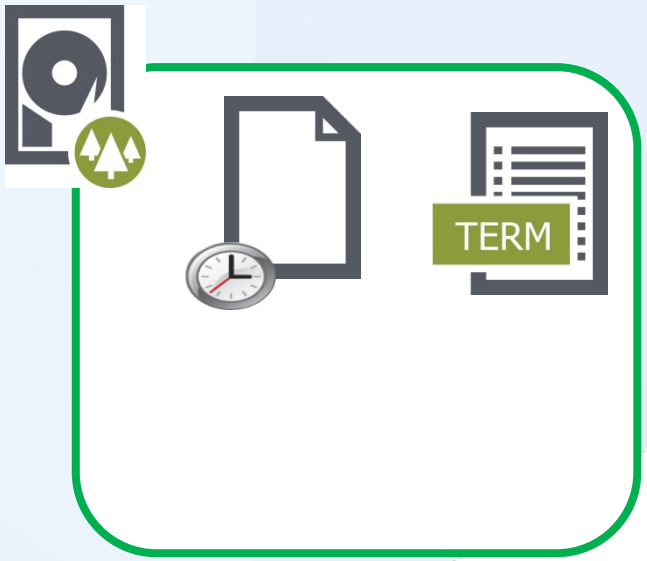
Labs:
Load data
MLCP – Creating ml-Gradle Custom Tasks

Progress® MarkLogic®

Document management

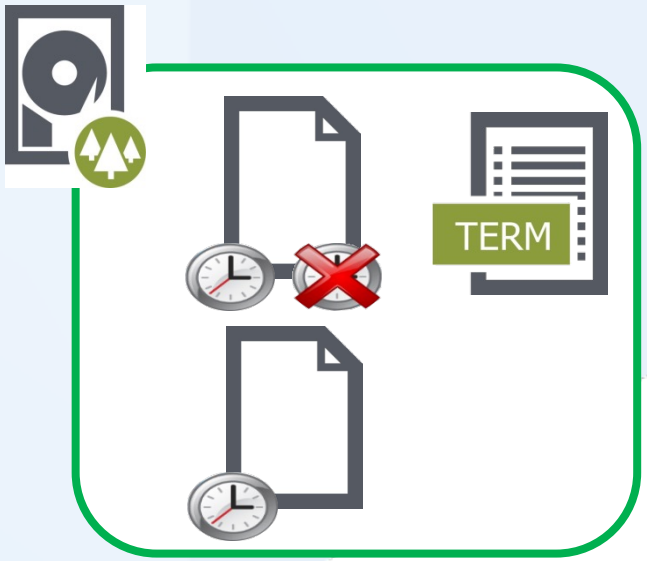
Multi-Version Concurrency Control – MVCC

- When a document is first loaded into the database, MarkLogic sets the document “start timestamp”.
- For more information on concurrency, please look into docs or take the data services course.



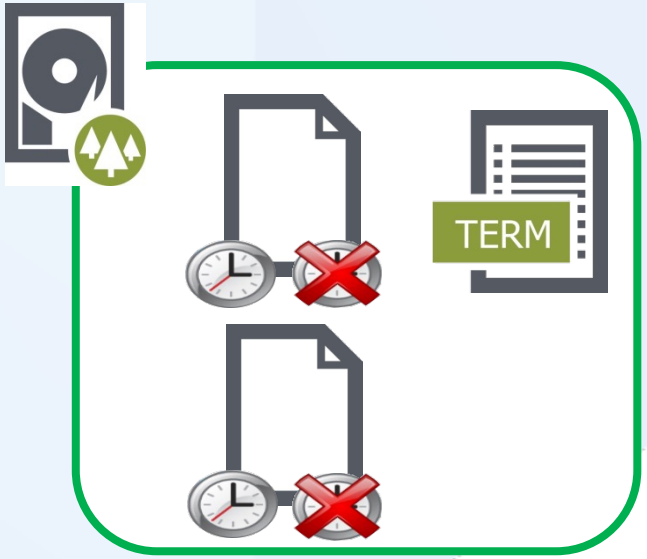
Updates

- Using `xdmp` functions to delete or modify documents with the same URI will generate deleted fragments - **There is no “update-in-place”.**
- The following actions count as an update:
  - Loading new content with the same URI.
  - Changing document permissions.
  - Adding/removing collections.
  - Replacing/adding/removing a property/element.
  - Any change to document meta-data (quality, property, meta-data).
- Each update creates a new version of that document.
  - The “end timestamp” is set.



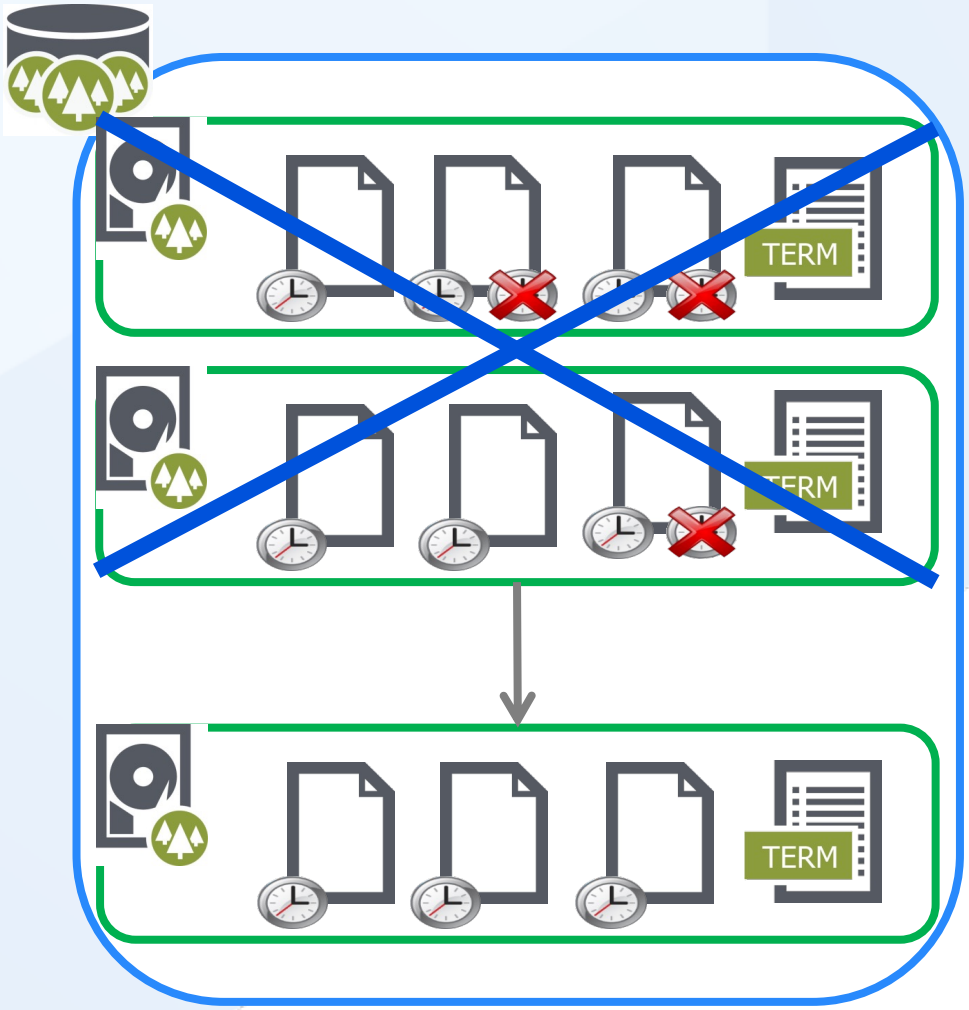
Deletes

- Deleting a document sets the “end timestamp” of a document.
- Documents with “end timestamps” are called “deleted fragments”.
- Physical deletion does not happen immediately.
  - Deleted fragments can accumulate over time, this does not significantly affect your query speed.
- MarkLogic supports point-in-time query as long as MarkLogic has not executed the `**Merge**` process.



Merges

- MarkLogic will drop deleted fragments during “merges”.
- This allows for recovery of resources:
  - Disk
  - RAM



Document Creation Libraries

- XML Data Management Platform (xdmp):
  - **xdmp.documentLoad** - Function used to load documents from the filesystem into the database.
  - **xdmp.documentInsert** – Function used to write in-memory documents into the database, for example, a document created within a code block.
- Uni- and ‘Bi-temporal` library:
  - `temporal` - Documents that are managed as a series of versioned documents in a protected Collection.
  - **temporal.documentLoad** and **temporal.documentInsert** – Similar functions used when implementing uni- or bi-temporal document.
- REST API
  - Allows for client-side application capable of invoking HTTP REST API to load content.

Multi-Version and `Temporal` Functions

- Temporal is used when there is a requirement to maintain snapshots of a transaction across various time dimensions.
  - Example: Financial and Insurance industries use bitemporal data to track changes to contracts, policies and events to adhere to strict regulation and compliance requirements.
- When using Temporal Functions, MarkLogic will generate a version of the document with new system time (and optional valid time).
  - These versions are retained upon merge.
  - More information on time-related constraints can be found in the [Temporal Developer's Guide](#).

☐	koolorder.1637501933324391017 4.json	object	(no properties)	kool, koolorder.json
☐	koolorder.722662528974205989.json	object	(no properties)	kool, koolorder.json
☐	koolorder.821364416548035268.json	object	(no properties)	kool, koolorder.json
☐	koolorder.json	object	(no properties)	kool, koolorder.json, latest

Built-in Mechanism to Update Documents

- `xdmp` and `temporal` libraries:
  - `nodeReplace` and other similar functions allow for update of just the document content or structure without affecting meta-data.
  - `documentAddCollections`, `documentAddPermissions`, etc. may be used to modify document meta-data without affecting actual content.
- REST API:
  - PUT and POST allows for the update of both content and meta-data (quality, permission, collection, etc.)
  - PATCH allows for update of just the document content or meta-data separately.
    - Equivalent of functions like nodeReplace, documentAddCollections, documentAddPermissions, etc.

Tool: Content Reprocessing in Bulk – CORB

- A Java-based tool:
  - Uses two modules/files:
    - Custom URI selector module that defines the documents to be processed.
    - Custom transformation module to process documents identified by the URI selector module.
  - Pre- and post- batch task hooks.
  - Throttle of threads and batch size
- Invoked via command line or Custom Gradle Task.

Tool: MarkLogic Data Hub

- A framework that consolidates the common tasks and configuration in creating a centralized operational Data Hub.
- Employs best practices like:
  - Use of the envelope pattern
  - Use of entity services
- Provides various pre-built steps that address common curation tasks:
  - Mapping of existing content to the target structure.
  - Smart Mastering - Matching and Merging of duplicate documents.

Progress MarkLogic

Lab: Accessing and Updating data

Progress® MarkLogic®

Security Basics

Privileges

- Actions that create or modify data, or actions that accesses data in a special way require corresponding privileges.
- Privileges cannot be assigned to users directly.

xdmp.documentInsert

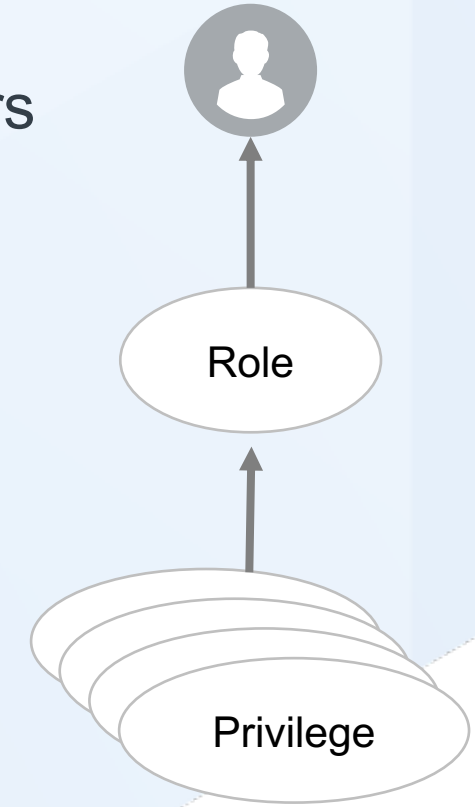
XQuery xdmp:document-insert

Required Privileges

If a new document is inserted, the `unprotected-uri` privilege (only if the URI is not protected), the `any-uri` privilege, or an appropriate URI privilege is also needed. If adding an unprotected collection to a document, the `unprotected-collections` privilege is needed; if adding a protected collection, the user must have either permissions to update the collection or the `any-collection` privilege.

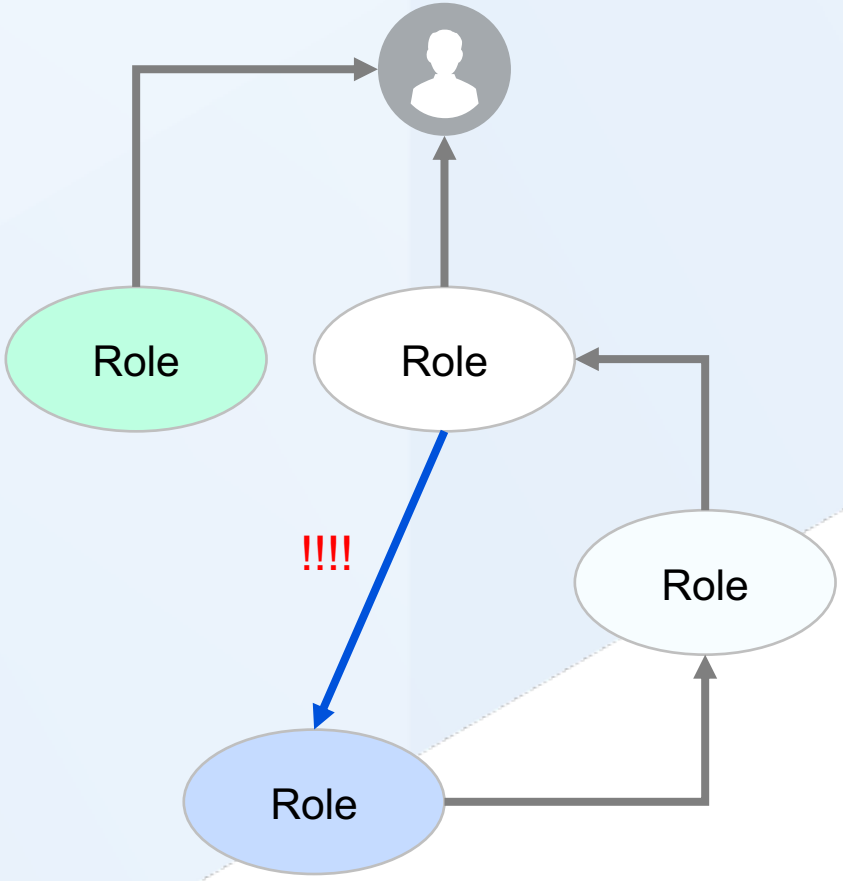
Roles

- **Roles** play the central part in controlling access to documents in MarkLogic database.
- Each role is a collection of **privileges** that allow users access to functions to perform their tasks.
- Users are assigned roles.



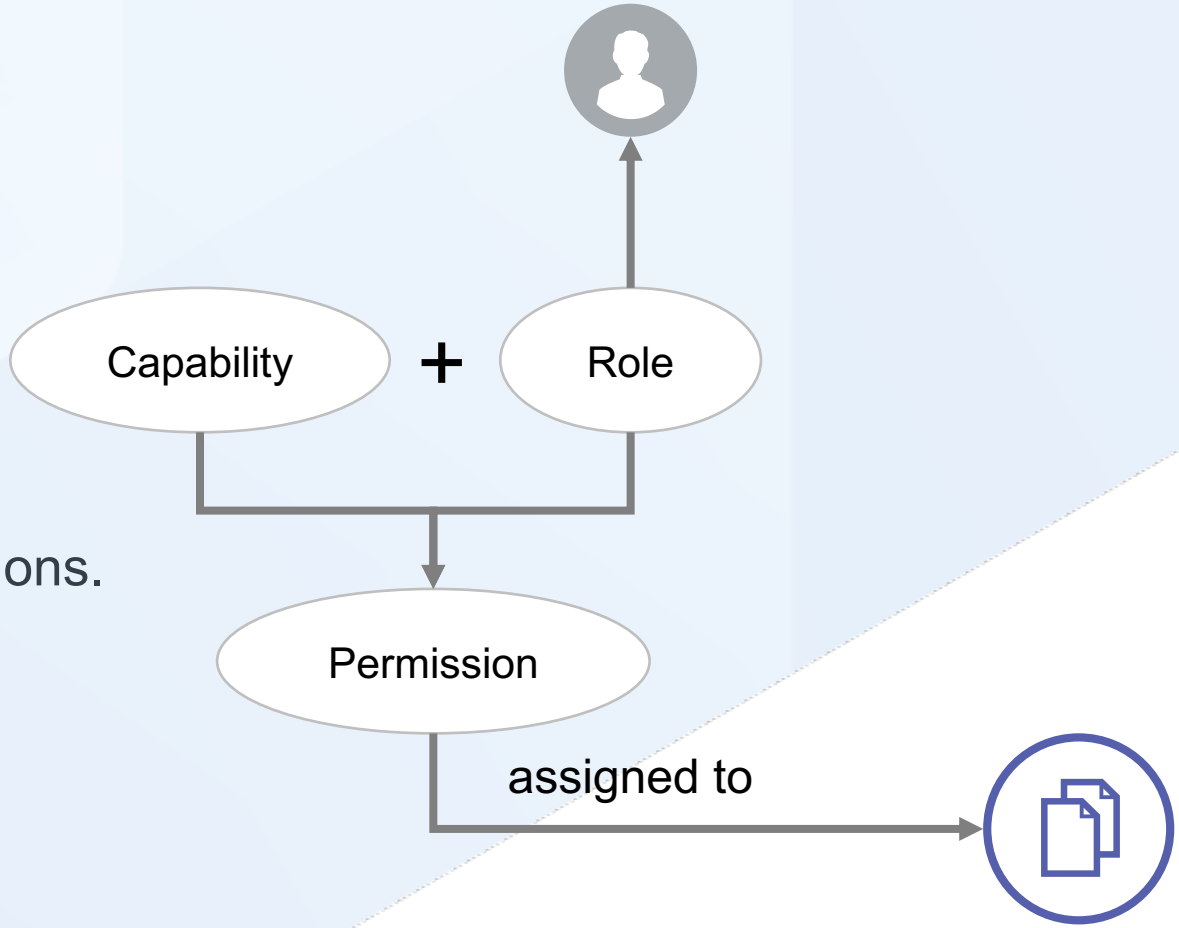
Roles

- A user may be assigned multiple roles, a role may be assigned to multiple users.
- A role may inherit other roles to be able to perform any action the other role is allowed to.
    - Be careful of circular role inheritance.



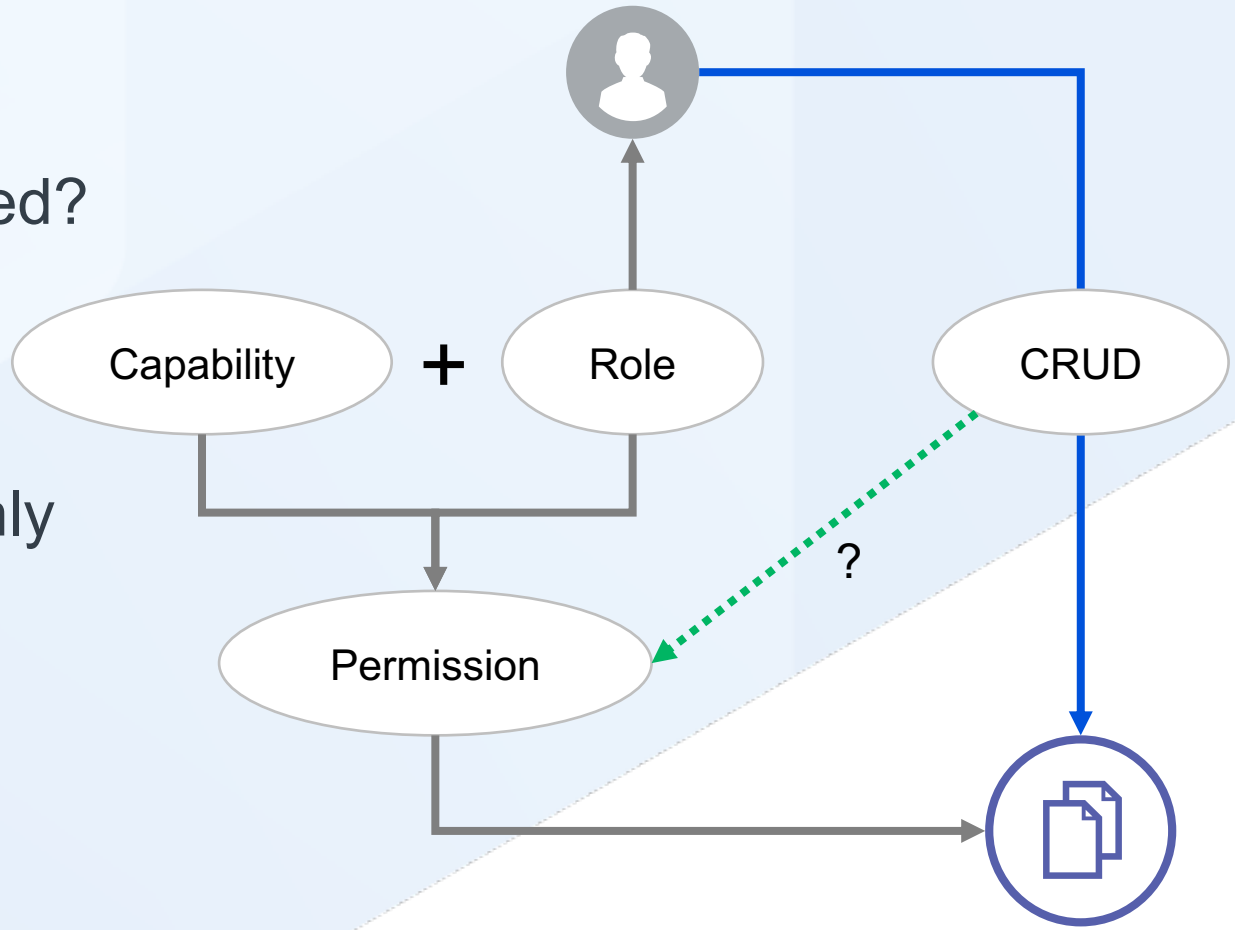
Role Based Access Control (RBAC)

- Document Permission = role + capability
- Capability include:
  - Insert
  - Read
  - Update
  - Node-update
  - Execute (module-specific)
- A document may be assigned multiple permissions.
- Permissions may be assigned during insert (recommended) or after – as an update.



Role Based Access Control (RBAC)

- Each action of a user will check the user's assigned roles against all configured permissions of a document:
  - Does the user have the role assigned?
  - Is the role paired with the right capability?
- Documents without permissions can only be read and/or modified by an `admin` level user.



Progress MarkLogic

Labs:
CORB2 – using ml-Gradle Custom Tasks
Upload Files Using WebDav (optional)

Recap

- MarkLogic Server supports structured, unstructured and binary content.
- Graphs are structured documents of a certain shape/structure.
- Documents can be organized using directories and/or collections.
- Use envelope pattern to compartmentalize your documents.
- MarkLogic Server creates a new version of a document for each insert/update.
- Use tools like MLCP for efficient bulk ingest.
- Use tools like CoRB2 for efficient bulk update.