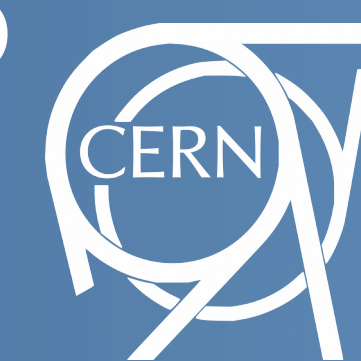


# Design principles of the Metadata Querying Language (MQL) implemented in the ATLAS Metadata Interface (AMI) ecosystem



This poster describes the design principles of the Metadata Querying Language (MQL) implemented in AMI, a metadata-oriented domain-specific language allowing to query databases without knowing the relation between tables. With this simplified yet generic grammar, MQL permits writing complex queries much more simply than Structured Query Language (SQL).

## What is MQL?

The Metadata Querying Language (MQL) is a domain specific language for executing queries on a RDBMS closely to spoken language.

**Physicist:** « I want to list the names of the datasets that have files with size > 0 »

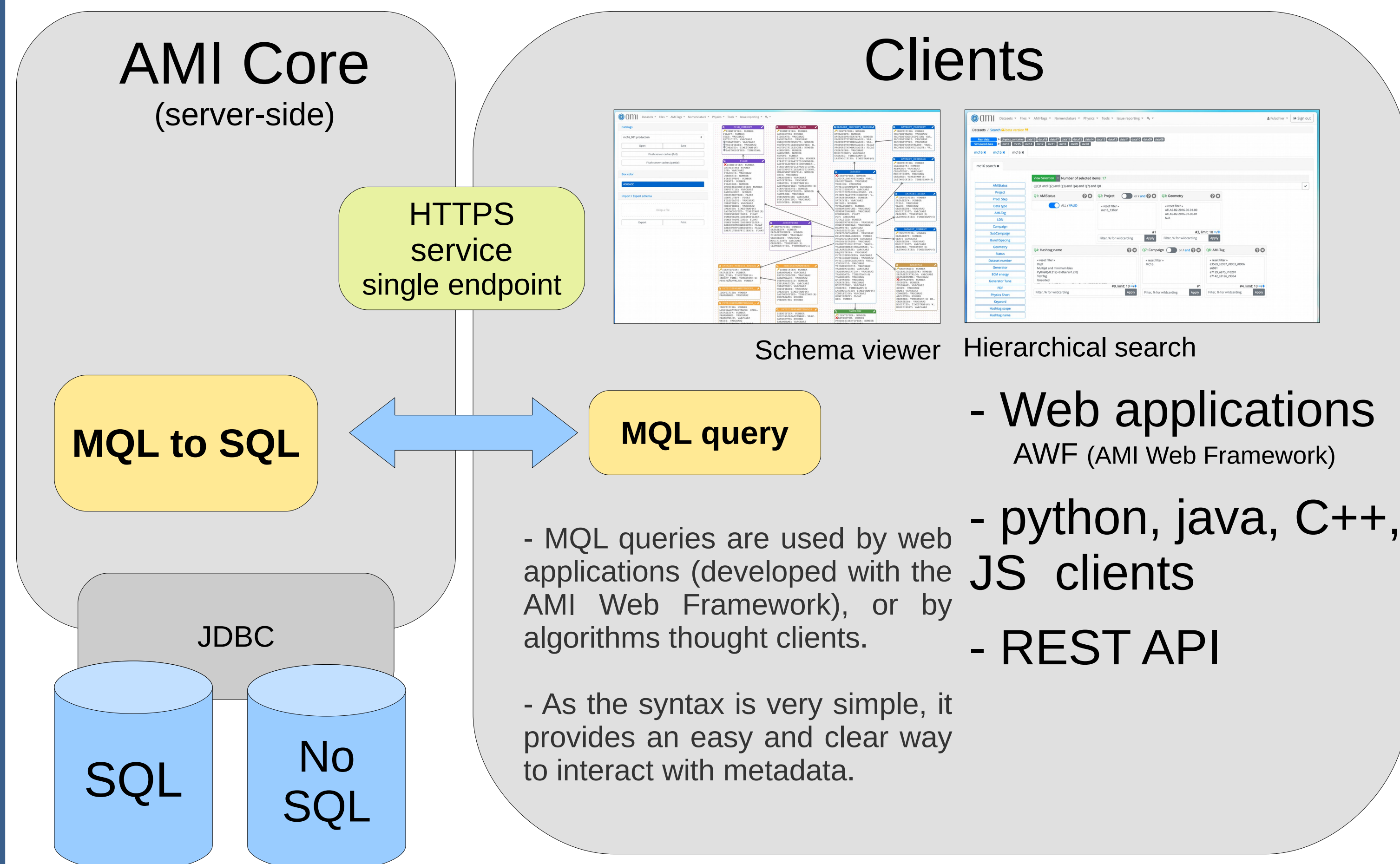
**MQL:** « It's easy ! »

```
SELECT dataset.name WHERE file.size > 0 »
```

**SQL:** « It depends on your DB schema... »

```
SELECT dataset.name FROM dataset, file WHERE dataset.Id = file.FK AND file.size > 0 »
```

## AMI clients



## MQL to SQL

### MQL

- No **FROM** clause
- No **join** in **WHERE** clause

```
SELECT Dataset.Name
WHERE
[
  FileParam.Name = 'Xsection'
  AND
  FileParam.Value > x
]
AND
[
  FileParam.Name = 'Luminosity'
  AND
  FileParam.Value < y
]
```

### SQL

- **FROM** clauses
- Joins in **WHERE** clause
- Isolations in **IN** clauses

```
SELECT DATASET.NAME
FROM DATASET, FILE
WHERE
DATASET.ID = FILE.DATASETFK
AND
FILE.ID IN
(
  SELECT FILE.ID
  FROM FILE, FILEPARAM
  WHERE
  FILE.ID = FILEPARAM.FILEFK
  AND
  FILEPARAM.NAME = 'Xsection'
  AND
  FILEPARAM.VALUE > x
)
AND
FILE.ID IN
(
  SELECT FILE.ID
  FROM FILE, FILEPARAM
  WHERE
  FILE.ID = FILEPARAM.FILEFK
  AND
  FILEPARAM.NAME = 'Luminosity'
  AND
  FILEPARAM.VALUE < y
)
```

### AMI Framework (JAVA)

- MQL queries:
- SELECT
  - INSERT
  - UPDATE
  - DELETE

#### PARSING

MQL query is parsed with ANTLR

#### TREE TRAVERSAL

#### FIELD RESOLUTION

**ISOLATION & JOIN**

```
[ Xsection > x ] AND [ Luminosity < y ]
```

[ ]: isolated expr.  
( ): simple expr.

SQL is executed on the datasource using the AMI primitives (users and roles, transaction engine, connection pooling, JDBC Interface)

Simple relations, bridges, cycles  
The AMI algorithms can handle complex database structures

#### REFLEXION

databases, tables, fields, relations and indices extracted and put in cache

#### DATABASES

Dataset

ID

ID

File

FK

FileParam  
Name / Value  
(e.g. x-section, luminosity)